UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Istenič

# Razvoj večdotične površine na osnovi barvno-globinske kamere

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Danijel Skočaj

Ljubljana, 2013

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

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: 01895/2013
Datum: 01.02.2013

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

Kandidat: **KLEMEN ISTENIČ**

Naslov: **RAZVOJ VEČDOTIČNE POVRŠINE NA OSNOVI BARVNO-GLOBINSKE KAMERE**

**MULTI-TOUCH SURFACE BASED ON RGBD CAMERA**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomski nalogi izdelajte uporabniku prijazno in cenovno sprejemljivo večdotično površino. Sistem naj temelji na projektorju, ki na poljubno ravno površino projicira sliko, ter na barvno-globinski kameri Kinect, ki zajema 3-dimenzionalno informacijo. Sistem naj detektira konice prstov rok ter jim sledi v prostoru in zaznava dotike prstov s površino. Zaznane podatke naj preko protokola TUIO posreduje ciljni aplikaciji in tako omogoči uporabniku večdotični nadzor le-te. Izdelan sistem primerno ovrednotite ter ocenite njegovo natančnost in odzivni čas. Predstavite tudi praktične primere uporabe in ocenite uporabniško izkušnjo.

Mentor:

doc. dr. Danijel Skočaj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

# IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Klemen Istenič,

z vpisno številko 63060226,

sem avtor diplomskega dela z naslovom:

Razvoj večdotične površine na osnovi barvno-globinske kamere

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja
- 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 16.6.2013 Podpis avtorja:

*Zahvaljujem se mentorju doc. dr. Danijelu Skočaju za vso pomoč in številne nasvete pri izdelavi tega diplomskega dela. Prav tako se zahvaljujem as. mag. Luki Čehovinu, ki mi je s predlogi in napotki izdatno pomagal tekom razvoja sistema.*

*Hvala podjetju Dolmen d.o.o. (Piki, Ireni in Igorju) za vso podporo pri izdelavi diplomskega dela in gostoljubje v njihovih poslovnih prostorih.*

*Posebna zahvala gre mojim staršem, ki so me tekom celotnega študija vedno podpirali, tako moralno kot finančno. Zahvala gre tudi teti Lidiji, bratu Timoteju in vsem prijateljem, ki so me med študijem spodbujali in podpirali.*

# Kazalo

# Seznam uporabljenih kratic in simbolov

**2D** (*angl.: Two Dimensional*) – dvo-dimenzionalen

**3D** (*angl.: Three Dimensional*) – tri-dimenzionalen

**GPE** (*angl.: Graphics Processing Unit*) – grafična procesna enota

**HTML** (*angl.: Hypertext Markup Language*) – označevalni jezik za oblikovanje večpredstavnostnih vsebin

**IR** (*angl.: Infrared light*) – infrardeča svetloba

**KF** (*angl.: Kalman filter*) – Kalmanov filter

**NUI** (*angl.: Natural User Interface*) – naravni uporabniški vmesnik

**RANSAC** (*angl.: RANdom SAmple Consensus*) – soglasje naključnih vzorcev

**SDK** (*angl.: Software development kit*) – programski paket za razvoj

**TUIO** (*angl.: Tangible User Interface Objects*)

**OSC** (*angl.: OpenSound Control*)

**UDP** (*angl.: User Datagram Protocol*)

# Povzetek

Interaktivne površine postajajo vse bolj priljubljene, saj je njihova uporaba zelo preprosta ter uporabnik za njihovo upravljanje ne potrebuje posebnih pripomočkov. Z uporabo ustreznih tehnologij lahko skoraj vsaki površini dodamo funkcionalnost večdotičnosti.

Danes številni muzeji uporabljajo interaktivne površine za prikaz informacij na uporabniku zanimive načine, vedno več šol in podjetij pa uporablja interaktivne table, ker te povečujejo zanimanje slušateljev. Interaktivne mize, stene in tla so vedno bolj zanimivi tudi za uporabo v oglaševalske namene, saj vedno vzbudijo veliko zanimanje med ljudmi.

Poleg bolj uveljavljenih dvo-dimenzionalnih interaktivnih površin se počasi uveljavljajo tudi tri-dimenzionalne. S prihodom nizko cenovnih barvno-globinskih kamer, te postajajo vse bolj dosegljive tudi širši javnosti. Tri-dimenzionalne interaktivne površine omogočajo uporabniku interakcijo ne le z dotiki površine, ampak tudi s spreminjanjem oddaljenosti prstov od površine.

V diplomskem delu je opisan razvoj takega sistema za tri-dimenzionalno interakcijo uporabnika z računalnikom. Sistem omogoča zaznavo in sledenje konicam uporabnikovih prstov. Površine z dodano funkcionalnostjo večdotičnosti so uporabniku prijazne ter cenovno dostopne. Tri-dimenzionalne informacije o sceni sistem zajame s pomočjo barvno-globinske kamere Kinect. Podatke o zaznanih prstih posreduje ciljni aplikaciji preko protokola TUIO. Sistem ovrednotimo na podlagi natančnosti zaznave, odzivnega časa ter uporabniške izkušnje. Razvit sistem se je med ocenjevanjem izkazal za zelo natančnega ter dovolj odzivnega za uporabo v vsakdanjih aplikacijah.

## Ključne besede:

Kinect, barvno-globinska kamera, interakcija človek-računalnik, naravni uporabniški vmesnik, tri-dimenzionalna interakcija, večdotična površina

# Abstract

The popularity of interactive surfaces is increasing due to their user-friendliness and the lack of need to use special controllers. With appropriate technologies we can add multi-touch functionality to almost any surface.

Today numerous museums use interactive surfaces to display the information on user-interesting ways. Number of schools and companies using interactive tables is increasing, due to the positive effects the usage has on the enhancement of the interest of participants. Interactive tables, walls and floors are usually useful for advertising purposes as well as they always attract crowds.

In addition to more established two-dimensional interactive surfaces, three-dimensional are slowly emerging as well. With the release of low-cost RGBD cameras they are becoming increasingly available to general public. They allow the users to interact with the system not only by touching the surface but also by adjusting the distance of the fingers to the surface.

This diploma thesis describes the development of a system for 3D human-computer interaction. The system is capable of detecting and tracking user's fingertips and thus creating user-friendly and affordable multi-touch surface. Three-dimensional information of the observed scene is captured by RBGD camera Kinect. Information about the detected fingertips is transmitted to the target application over TUIO protocol. System is evaluated based on the accuracy of detections, response time, and user experience. During the assessment it proved to be very accurate and responsive enough to be used in everyday applications.

**Key words:**
Kinect, RGDB camera, human-computer interaction (HCI), natural user interface (NUI), 3D interaction, multi-touch surface

# Poglavje 1

# Uvod

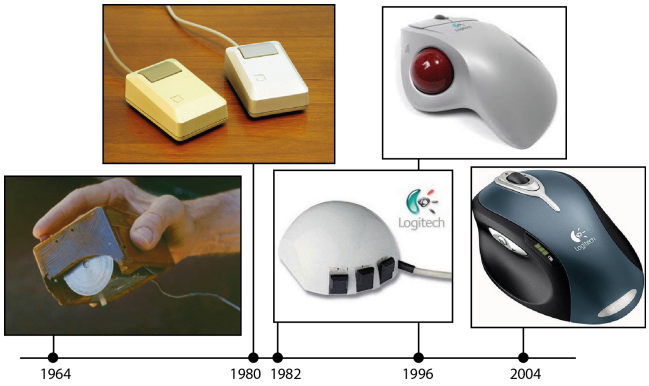
## 1.1 Motivacija

Računalništvo je v zadnjih desetletjih napredovalo s svetlobno hitrostjo. Računalniki so iz velikanskih strojev, ki so obsegali celotne sobe, postali prenosljivi ter dostopni praktično vsakemu posamezniku. Njihova uporabnost se je, iz računanja preprostih matematičnih izračunov, razširila na praktično vsako področje našega vsakdana. Svet je postal povsem odvisen od računalnikov.

Ne glede na ves napredek je način interakcije človeka z računalnikom ostal praktično nespremenjen od 80. let 20. stoletja, ko je računalniška miška postala nepogrešljiv del vsakega računalnika. Njena oblika se je skozi čas spreminjala (slika 1.1), postala je brezžična ter optična, njen osnovni princip delovanja pa je kljub vsemu ostal nespremenjen. Kljub številnim pomanjkljivostim, kot so na primer nezmožnost predvidevanja lokacije kurzorja na zaslonu s pogledom na miško, konstantno preslikovanje horizontalne ravnine, na kateri uporabnik upravlja z miško na vertikalno ravnino zaslona ter povečane možnosti poškodbe ob vsakodnevni uporabi zaradi ponavljajočih se gibov (angl. repetitive strain injury), raziskovalcem do danes ni uspelo razviti bolj priljubljenega načina interakcije.

Poleg omenjenih težav ima uporaba miške še eno posebej veliko omejitev - uporabniku omogoča le eno točko interakcije s sistemom. Za boljšo predstavo, kaj takšna omejitev pomeni, lahko za primer vzamemo povsem enostavno vsakodnevno dejanje, kot je zavezovanje čevlja. Redko kdo si je sposoben zavezati čevelj z eno roko, kaj šele z uporabo le enega prsta.

V zadnjih letih se je razvilo več tehnologij, ki omogočajo uporabniku več sočasnih točk interakcije s sistemom in hkrati odpravljajo večino prej omenjenih težav. Najbolj pogosta implementacija je uporaba zaslona na dotik. S




Slika 1.1: Razvoj računalniške miške

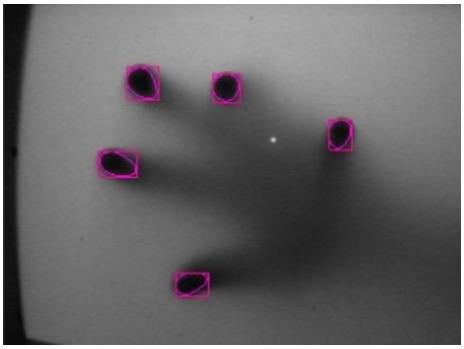
pocenitvijo izdelave zaslonov na dotik se ti sedaj uporabljajo v praktično vseh telefonih in tabličnih računalnikih, v zadnjem letu pa vse bolj pogosto tudi na prenosnih računalnikih (slika 1.2). Njihova glavna pomanjkljivost je še vedno izjemno visoka cena za zaslone večjih dimenzij, tako da je njihova komercialna uporaba omejena na manjše dimenzije.

Slika 1.2: Prenosni računalnik Dell Studio 17 z zaslonom na dotik (vzeto iz [24])

V nasprotju z njimi je izdelava večdotične površine, z uporabo IR kamer in oddajnikov, projektorja ter algoritmov računalniškega vida, pri večjih dimenzijah relativno enostavna in cenovno ugodna. Na slikah zajetih z IR kamerami, lahko določimo mesta uporabnikovih dotikov površine, ker se projicirana IR svetloba odbija od uporabnikovih prstov. Ta področja nato zaznamo z uporabo algoritmov računalniškega vida (slika 1.3). Število tako zaznanim mest je omejeno le z zmožnostjo računalnika na katerem se algoritmi izvajajo. Opa-

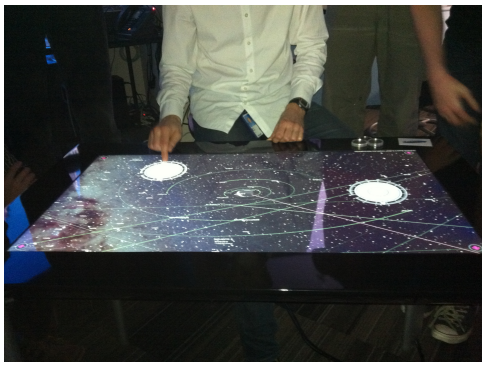
zovana površina pa je omejena le s skupnim vidnim poljem vseh uporabljenih IR kamer.

Najbolj znan primer take uporabe je interaktivna miza Samsung SUR40 (slika 1.4) s tehnologijo Microsoft PixelSense [29]. Ker je sistem v prodaji le skupaj z drago strojno opremo, je rešitev zelo draga in posledično ne preveč razširjena med običajnimi uporabniki.

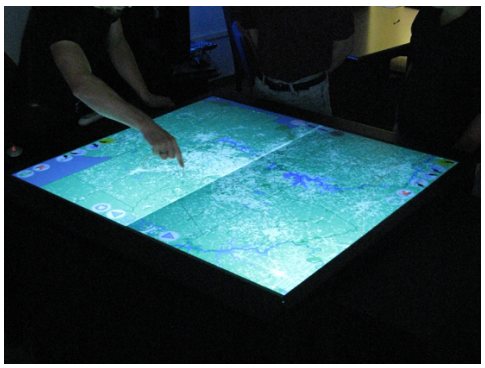


Slika 1.3: Slika zajeta z IR kamero skupaj z zaznanimi uporabnikovimi dotiki (vzeto iz [22]

Podoben princip zaznave uporablja tudi odprtokodno rešitev CCV (Community Core Vision) [22] raziskovalne skupnosti NUI (Natural User Interface Group) [33]. Ta omogoča uporabniku izdelavo lastne interaktivne mize (slika 1.5) za le delček cene interaktivne mize Samsung SUR40.



Slika 1.4: Samsung SUR40 s tehnologijo Microsoft PixelSense



Slika 1.5: Interaktivna miza izdelana z CCV

Čeprav predstavljene rešitve omogočajo večdotično interakcijo uporabnika,

je ta še vedno omejena na 2-dimenzionalno (2D) ravnino. Če ponovno vzamemo primer zavezovanja čevlja, je to primerljivo z zavezovanjem, kjer prsti ne smejo zapustiti ravni podlage. Čeprav lažje kot z enim samim prstom, še vedno predstavlja nepredstavljivo zapleteno nalogo v primerjavi z načinom, ki ga uporabljamo v vsakdanjem življenju.

Predstavitev nizko cenovne barvno-globinske kamere Kinect je tako na področju interakcije človeka z računalnikom, kot celotnega računalništva, pomenila pravo revolucijo. Podpredsednik ter glavni analitik podjetja Forrester James L. McQuivey je po predstavitvi kamero Kinect opisal z besedami: *"Kinect bo za naslednje desetletje to, kar je bil operacijski sistem za osemdeseta, računalniška miška za devetdeseta in kar je internet od nekdaj. Je stvar, ki bo spremenila vse"* [28].

Pred tem je bila 3-dimenzionalna (3D) interakcija uporabnika s sistemom možna le z uporabo posebej za to namenskih objektov (Nintendo Wii kontroler 1.6, posebna rokavica 1.7 itd.) ali pa z uporabo sistema več barvnih kamer, ki pa je zahteval zahtevno in natančno kalibracijo. Tako je bila taka interakcija v večini na voljo le posameznim navdušencem ter raziskovalcem.


Slika 1.6: Nintendo Wii kontroler (vzeto iz [32])


Slika 1.7: Posebna rokavica za 3D interakcijo (vzeto iz [37])

Enostaven dostop do 3D informacije o sceni, ki ga kamera Kinect zajema ter izjemno nizka cena, sta v zadnjih letih omogočili bliskovit razvoj 3D interakcije uporabnika z računalnikom. Raziskovalci so z dodatno informacijo o tretji dimenziji dobili možnost robustnejše in natančnejše zaznave oseb, objektov in celo posameznih uporabnikovih prstov. Danes že kar nekaj podjetij (Ubi Interactive, GestSure Technologies, Manctl itd.) uporablja kamero Kinect kot del svojega produkta. Ogromen potencial pa je zaznal tudi sam Microsoft, ki je leta 2012 organiziral prvi pospeševalnik za podjetja z najboljšimi idejami o uporabi kamere Kinect [36].

## 1.2 Sorodna dela

Veliko zanimanje za njihovo uporabo je tudi na področju interaktivnih površin. Z uporabo globinske kamere je enostavno dodati funkcionalnost večdotičnosti praktično vsaki površini, brez potrebe po opremljanju površine z dodatno opremo. A. Wilson je v [20] predstavil smernice za razvoj 2D večdotičnih površin, z opazovanjem majhnega prostora nad površino. Obstoj dovolj velikega števila povezanih elementov v tem pasu pomeni uporabnikov dotik. Preprostost predstavljenega pristopa je botrovala velikemu številu implementacij entuziastov in raziskovalcev. Ena izmed takih implementacij je lepo predstavljena v [6]. S. Murugappan, et al. so v [15] poleg prostora neposredno nad površino za zaznava uporabili tudi prostor nad njim, kjer so zaznavali uporabnikove roke. Na ta način so enostavno zaznali lažne dotike, nastale kot posledica šuma pri zajemu slike. Poleg tega so dodatne informacije omogočale združevanje dotikov posamezne roke in rok glede na uporabnika. Pokazali so še, da je z novimi informacijami mogoče tudi zaznati ali je roka leva ali desna.

Uporaba globinskih kamer omogoča dodajanje večdotične funkcionalnosti tudi ne ravnim površinam. Z uporabo prenosne globinske kamere in projektorja, sistem OmniTouch [3] omogoča zaznavo dotika na poljubni površini med vsakdanjimi opravili. Posebej je zanimiv, ker omogoča zaznavo v realnem času tudi na površinah, katerih oblika se vseskozi spreminja (npr. uporabnikova dlan). Poleg 2D so raziskovalci globinske kamere začeli uporabljati tudi za 3D interaktivne površine. Hilliges et al. so v [5] z opazovanjem prostora nad površino omogočili uporabniku preprosto interakcijo z navideznimi predmeti prikazanimi na površini. V [1] pa so šli še korak dlje, saj so omogočili preslikavo resničnega objekta v navidezni svet in uporabniku upravljanje z njim. Preslikavo resničnih objektov v različne aplikacije in igrice so uporabili tudi v [8, 7].

Razvoj 3D interaktivnih površin, kjer zaznavamo in sledimo konicam uporabnikovih prstov na in nad opazovano površino je trenutno izredno aktualna raziskovalna tema. Podjetje Ubi Interactive je razvilo sistem [39], ki omogoča 3D interakcijo na poljubni ravni površini. Korak dlje pa so naredili F. Klompmaker, et al. s sistemom dSensingNI [12], ki omogoča zaznavo na poljubni obliki površine. Poleg sledenja uporabnikovim prstom lahko sistem sledi tudi objektom na površini.

3D interaktivne površine so uporabne na praktično vseh področjih, kjer uporabnik upravlja z računalnikom. Uporabnik lahko s premikanjem prstov v prostoru nad opazovano površino upravlja z aplikacijo drugače kot s premikanjem na površini. S približevanjem ter oddaljevanjem prstov od površine

bi lahko upravljal s količino podrobnosti, ki so prikazane ali pa povečeval in zmanjševal zemljevid oziroma sliko. Informacijo o oddaljenosti bi lahko uporabili tudi v številnih grafičnih programih, kjer bi na tak način lahko na primer določili debelino črt ali velikost orodja. Zanimiva bi bila tudi uporaba v igricah, kjer bi hitrost približevanja in oddaljevanja lahko pomenila silo, s katero uporabnik izvede določeno potezo v igri. Veliko dodano vrednost bi tak sistem lahko prinesel tudi programom za učenje (npr. klavirja). S sledenjem prstov bi lahko ugotovili napake, ki jih uporabnik naredi med igranjem. Ker za upravljanje ne potrebujemo nobenega kontrolerja, je uporabna tudi v okoljih, ki zaradi svoje specifičnosti, zahtevajo sterilnost (npr. operacijske sobe).

## 1.3 Glavni cilji

Cilj diplomskega dela je razvoj sistema, ki bo omogočal zaznavo in sledenje konicam uporabnikovih prstov na opazovani površini in prostoru nad njo. Z uporabo 3D informacij o sceni, zajetimi z barvno-globinsko kamero Kinect ter z obdelavo le-teh z ustreznimi metodami računalniškega vida, predpostavljamo, da bo razvit sistem omogočal dovolj veliko natančnost zaznave in s tem udobno uporabo večdotične površine. Med razvojem sistema bomo pozornost namenili tudi optimizaciji količine obdelanih podatkov v posamezni globinski sliki, saj je naša želja omogočiti delovanje sistema na računalnikih brez podpore GPE. Podatke o zaznanih konicah uporabnikovih prstov sistem preko protokola TUIO posreduje ciljni aplikaciji. Tako bo uporaba razvitega sistema omogočila dodajanje večdotične funkcionalnosti poljubni ravni površini.

V okviru diplomskega dela bomo tudi ovrednotili delovanje in uporabniško izkušnjo. Ocenili bomo natančnost zaznave ter odzivni čas sistema. Pri tem bomo posebej določili čas, potreben za samo zaznavo konic prstov ter čas, ki ga kamera Kinect potrebuje za zajem in posredovanje informacij. Uporabniško izkušnjo bomo preverili z uporabo gonilnika UniSoftHID in programa Multi-touch Vista [30] ter preprostega programa za prikaz lokacijo zaznanih prstov v 3D prostoru. Program Multi-touch Vista omogoča nadzor nad operacijskim sistemom Microsoft Windows 7 ter vseh na njem delujočih aplikacij. Poleg tega bomo z uporabo programskega paketa Microsoft Touch Pack for Windows 7 [40], preverili uporabniško izkušnjo tudi pri aplikacijah, razvitih posebej za uporabo z interaktivnimi mizami.

## 1.4 Zgradba diplomskega dela

Preostanek diplomskega dela je razdeljen na 6 poglavij. V drugem poglavju bomo predstavili strojne in programske komponente razvitega sistema ter omejitve, ki jih le-te prinašajo. Poglavje 3 vsebuje opise procesov inicializacije in zaznave ter algoritma za detekcijo konic prstov rok. V poglavju 4 sta podana kratek opis protokola TUIO, ki ga uporabljamo za pošiljanje podatkov ciljni aplikaciji, ter opis možnih ciljnih aplikacij. Rezultati ovrednotenja ter primeri uporabe pa so prikazani v poglavjih 5 in 6. Na koncu sledi še zaključek diplomskega dela s predlogi nadaljnjih izboljšav.

# Poglavje 2

# Opis sistema

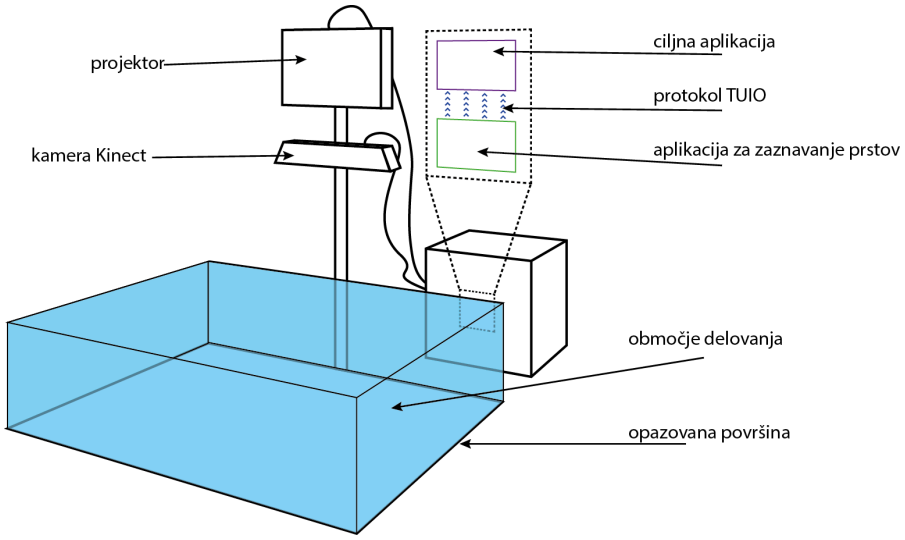
## 2.1 Zgradba sistema

Razvit sistem omogoča dodajanje 3D večdotične funkcionalnosti poljubni ravni površini. Sestavljajta ga strojna in programska oprema, katerih zgradba je prikazana na sliki 2.1. Za osnovno delovanje sistema sta potrebni le 3D kamera Kinect in povprečen računalnik, ki za delovanje v realnem času ne potrebuje podpore GPE. Za dobro uporabniško izkušnjo poleg prej naštetih potrebujemo še projektor. Ta na opazovano površino projicira sliko ciljne aplikacije, ki jo uporabnik preko kretenj (premiki prstov, dotiki površin itd.) upravlja. Na sliki 2.2 je prikazan sistem med uporabo.

Sistem je sposoben slediti poljubnemu številu uporabnikovih prstov v realnem času. Območje uporabe (opazovana površina in prostor nad njo) sta omejena z zmožnostmi kamere Kinect oziroma s prostorskimi omejitvami, ki jih določi uporabnik.V nadaljevanju bomo opisali posamezne komponente ter njihov način delovanja.

## 2.2 Barvno-globinska kamera Kinect

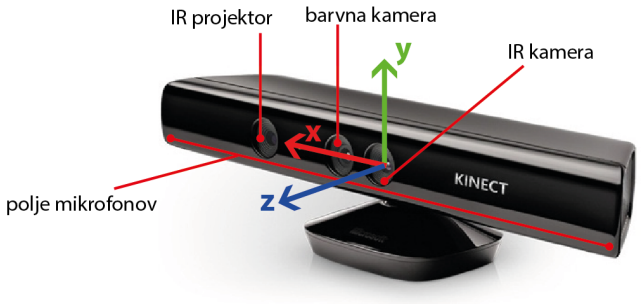
Kinect, znan tudi pod imenom "Project Natal", je naprava v osnovi namenjena brez-kontrolnem upravljanju igralne konzole Xbox 360. Med številnimi senzorji (slika 2.3), ki omogočajo zajem zvoka ter barvne in globinske slike, je najbolj znana globinska kamera. Njeno delovanje temelji na tehnologiji 3D zaznavanja podjetja Prime Sense. S pravilno kalibracijo barvne in IR kamere Kinect omogoča zajem barvnega oblaka 3D točk s hitrostjo 30 slik na sekundo. Čeprav je bil sprva namenjen le zabavi, so uporabniki od prve predstavitve leta




Slika 2.1: Shematski prikaz razvitega sistema

Slika 2.2: Razvit sistem

2009 prepoznali njegove številne možnosti uporabe. Danes se poleg zabave med drugim uporablja tudi na področjih robotike, naravnih uporabniških vmesnikov (NUI), prepoznavanja kretenj in navidezne resničnosti. Prav zaradi široke uporabnosti so sedaj na voljo številna orodja za povezavo z računalnikom, med katerimi najbolj izstopajo Microsoftov Kinect SDK, OpenKinect-ov libfreenect in OpenNI.




Slika 2.3: Kinect

### 2.2.1 Strojna oprema

Kinect je vsebuje široko zbirko senzorjev. Sestavljajo ga 3D senzor, ki je sestavljen iz IR kamere in IR projektorja, barvne kamere, zbirke mikrofonov in 3-osnega (XYZ) pospeškomera, katerih lokacijo na kameri Kinect prikazuje slika 2.3.

- 3D senzor: sestavljen iz IR projektorja, ki skupaj z IR kamero omogoča zajem 3D podatkov o prizoru, v kakršnih koli svetlobnih pogojih. Projektor projicira statičen vzorec točk na okolico, ki jo IR kamera posname. S primerjavo referenčnega in posnetega vzorca Kinect določi oddaljenost posamezne točke. Omogoča zajem globinske slike dimenzij $640 \times 480$ sl. el. pri hitrosti 30 slik na sekundo z 11-bitno ločljivostjo ($2^{11} = 2048$ različnih stopenj);
- barvna kamera: omogoča zajem 32 bitnih slik dimenzij $640 \times 480$ sl. el. pri hitrosti 30 slik na sekundo ali slik dimenzij $1280 \times 1024$ sl. el. pri hitrosti 15 slik na sekundo;
- polje mikrofonov: vgrajeni so štirje mikrofoni. Vsak kanal se procesira s 16 biti in frekvenco vzorčenja 16 kHz. Omogočajo prepoznavo zvoka, lokalizacijo zvočnega vira in odstranjevanje odmeva.

Kinect ima horizontalni vidni kot 57°in vertikalni 43°. Območje delovanja je omejeno med $0,4$ in 6 m, medtem ko se pri uporabi z igralno konzolo Xbox 360 zmanjša na območje med $1,2$ in $3,5$ m.

### 2.2.2 Določanje globine

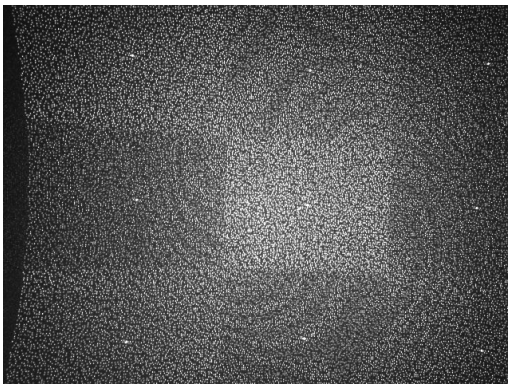
3D senzor sestavljata IR kamera in IR projektor (označena na sliki 2.3), ki sta povezana v stereo način na razdalji $b = 7,5cm$. IR projektor s pomočjo leče razprši žarek, ki povzroči konstanten vzorec točk (slika 2.4). Sestavljen je iz $3 \times 3$ mreže polj, vsake velikosti $211 \times 165$ točk. Vsaka podmnožica točk je neodvisna od preostanka točk, hkrati pa v vzorcu enolično določljiva.

Za določanje oddaljenosti posamezne točke ima kamera Kinect shranjen referenčni vzorec. To je vzorec, projiciran na ravnino, ki je oddaljena $Z_r$. Uporablja se za primerjavo s projiciranim vzorcem scene, zajetim preko IR kamere. Zamik v poziciji točk (v slikovnih elementih) predstavlja disperzijo. Določi jo na podlagi optimalnega ujemanja lokalnih vzorcev (vzorec velikosti $9 \times 9$ sl. el. okoli opazovane točke) z natančnostjo $1/8$ slikovnega elementa.

Pri pretvorbi dobljenih disperzij v 3D koordinatni sistem je potrebno najprej odpraviti učinke distorzije. Herrera et. al. so [4] opisali postopek določitve ter enačbo (2.1) za njeno odpravo.

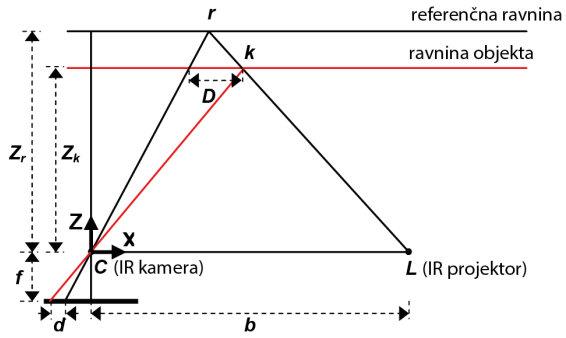
$$d_p = d + D_\delta (u, v) \cdot exp (\alpha_0 - \alpha_1 d), \tag{2.1}$$

kjer je $d_p$ popravljena vrednost disperzije, $d$ vrednost disperzije, dobljena z globinske slike kamere Kinect, $D_\delta$ vzorec prostorske distorzije in $\alpha_0$ ter $\alpha_1$ koeficienta spreminjanja velikosti napake disperzije.



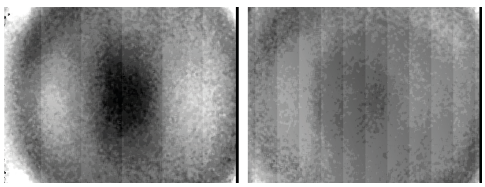
Slika 2.4: IR vzorec, ki ga projecira IR projektor [35]

Globinski koordinatni sistem kamere Kinect definiramo kot desno orientiran ortogonalni koordinatni sistem z izhodiščem v perspektivnem centru IR kamere. Os Z je pravokotna na ravnino slike v smeri objekta k, os X pa je pozitivna v smeri IR projektorja (sliki 2.3 in 2.5).

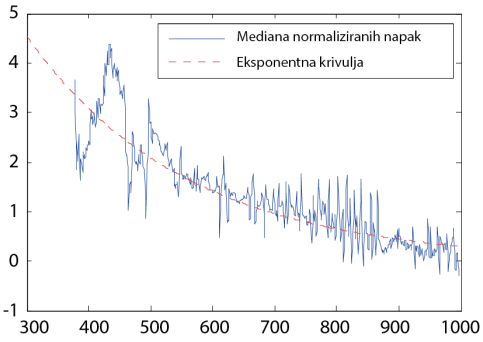



Slika 2.5: Povezava med disperzijo in razdaljo objekta [11]

Brez odprave distorzije imajo dobljene koordinate mersko napako s statičnem vzorcem, ki se z oddaljenostjo zmanjšuje. Odvisnost velikosti napake z oddaljenostjo prikazuje slika 2.7, vzorec napake pa slika 2.6.



Slika 2.6: Napake izmerjenih disperzij na razdalji $0,56m$ (levo) in $1,2m$ (desno) [4]




Slika 2.7: Velikost napake pri različnih oddaljenostih [4]

Pretvorbo popravljenih vrednosti disperzij naredimo na podlagi podobnih trikotnikov (slika 2.5), kjer veljata enačbi:

$$\frac{D}{b} = \frac{Z_r - Z_k}{Z_r} \tag{2.2}$$

in

$$\frac{d}{f} = \frac{D}{Z_k}, \tag{2.3}$$

kjer $Z_k$ predstavlja oddaljenost točke, $Z_r$ oddaljenost referenčne ravnine in $d$ ter $D$, ki predstavljata določeni dispariteti v slikovnih elementih in enotah koordinatnega sistema.

Z združitvijo enačb (2.2) in (2.3), dobimo enačbo za oddaljenost objekta od izhodišča koordinatnega sistema kamere Kinect:

$$Z_k = \frac{Z_r}{1 + \frac{Z_r}{fb}d}. \tag{2.4}$$

Kinect, zaradi omejene pasovne širine, ne omogoča pridobitve dejansko izmerjenih disparitet, ampak le normalizirane 11-bitne vrednosti. Uradno Prime Sense ni nikoli objavil načina normalizacije, vendar naj bi bil ta linearen. Zato v enačbo (2.4) vnesemo enačbo za linearno normalizacijo disperzij $(m \cdot d_n + n)$ in s tem dobimo enačbo za pretvorbo normalizirane disparitete v **koordinato z** točke globinskega koordinatnega sistema kamere Kinect:

$$\begin{aligned} Z_k &= \frac{1}{\left(\frac{m}{fb}\right) d_n + \left(\frac{1}{Z_r} + \frac{n}{fb}\right)} \\ &= \frac{1}{c_1 d_n + c_0}, \end{aligned} \tag{2.5}$$

kjer $c_0$ in $c_1$ predstavljata koeficienta, ki ju pridobimo s kalibracijo kamere Kinect.

Preostali koordinati točke **(x,y)** dobimo z uporabo modela kamere s točkasto odprtino (angl. pinhole camera model), kjer točke preslikamo v 3D prostor s pomočjo perspektivne transformacije.

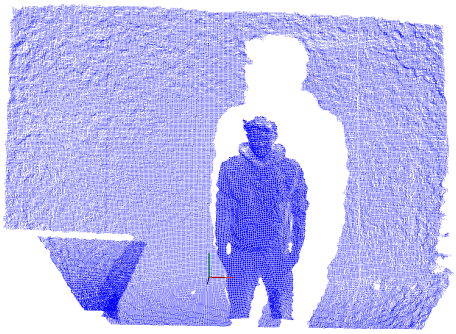
$$\begin{aligned} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}, \\ x' &= x/z, \\ y' &= y/z, \end{aligned} \tag{2.6}$$

kjer je (u,v) pozicija točke v globinski sliki, točka ($f_x$, $f_y$) goriščna razdalja in ($c_x$,$c_y$) pozicija perspektivnega centra IR kamere.

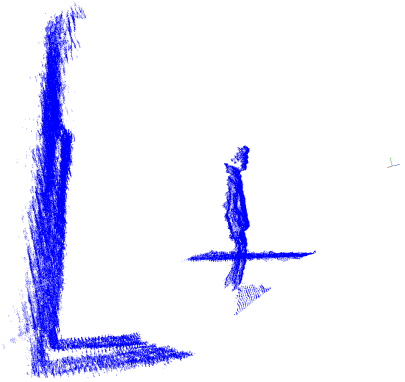
Sliki 2.8 in 2.9 prikazujeta barvno in globinsko sliko zajeti s kamero Kinect. Z uporabo globinske slike in prej opisanega postopka pretvorbe točk dobimo oblak 3D točk prikazan na sliki 2.10. Slika 2.11 prikazuje isti oblak točk s stranskega pogleda.

Slika 2.8: Barvna slika scene

Slika 2.9: Globinska slika scene



Slika 2.10: Oblak 3D točk scene



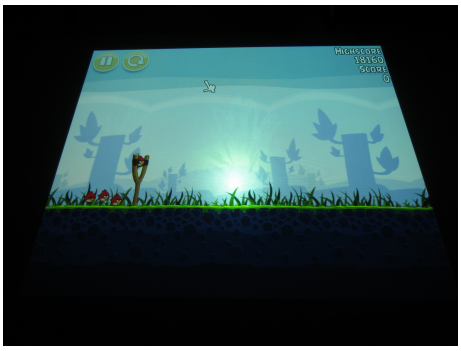
Slika 2.11: Stranski pogled na oblak 3D točk scene

## 2.3 Projektor

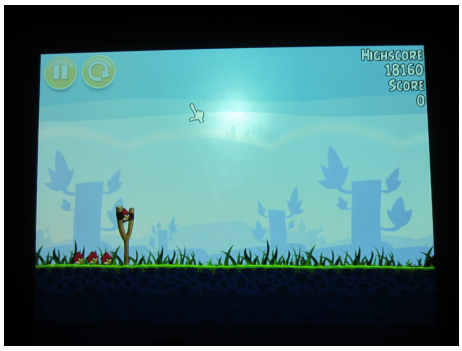
S pomočjo projektorja projiciramo sliko ciljne aplikacije na opazovano površino. Za delovanje sistema projektor ni nujno potreben, vendar neposredno prikazovanje posledic uporabnikovih dejanj omogoča boljšo uporabniško izkušnjo. S sistemom lahko uporabljamo kateri koli projektor, ki ga računalnik, na katerem se izvaja ciljna aplikacija, podpira. Pri izbiri projektorja so za nas najbolj pomembne lastnosti **velikost** in **ločljivost projicirane slike** ter možnost **trapeznega popravka slike oz. premika leče**. Pomembnost posamezne lastnosti je odvisna od namestitve projektorja glede na opazovano površino.

Končna velikost interaktivne površine bo manjša izmed dimenzij projicirane slike in maksimalne možne dimenzije opazovane površine s kamero Kinect. Zaradi prostorske omejitve je projektor velikokrat nameščen relativno blizu opazovane površine, tako da je lahko projicirana slika manjša od pričakovane.

Tako je pred končno izbiro projektorja priporočeno preveriti velikost projicirane slike na enaki razdalji, kot ga bomo kasneje namestili. Poleg tega je projektor v večini primerov nameščen pod nepravim kotom glede na opazovano površino. Slika ima v takem primeru, namesto pričakovane pravokotne (slika 2.13), trapezno obliko (slika 2.12). Veliko projektorjev omogoča odpravo takega popačenja z možnostjo premika leče.



Slika 2.12: Popačena projekcija

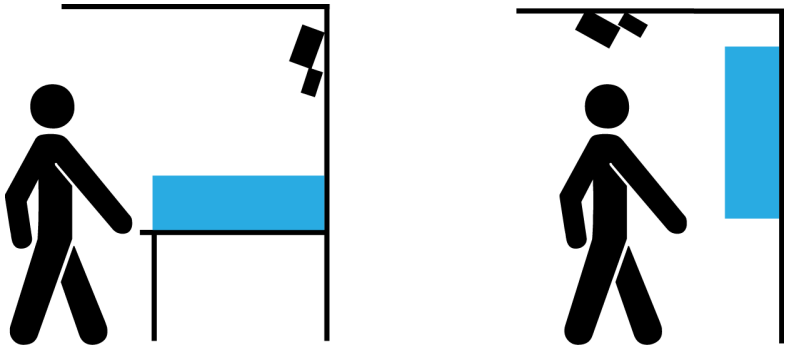



Slika 2.13: Popravljena projekcija

## 2.4 Površina

Površina, kateri želimo dodati funkcionalnost 3D večdotičnosti, je lahko poljubna ravna površina. Delovanje sistema je neodvisno od njene orientacije in naklona. Sistem lahko uporabimo tako na vodoravnih (mizah) kot na horizontalnih ravnih površinah (tablah) (slika 2.14). Za uporabo sistema na neravnih površinah bi bilo potrebno dodatno definirati le funkciji, ki bi preslikali poljubno obliko površine na ravnino in obratno.

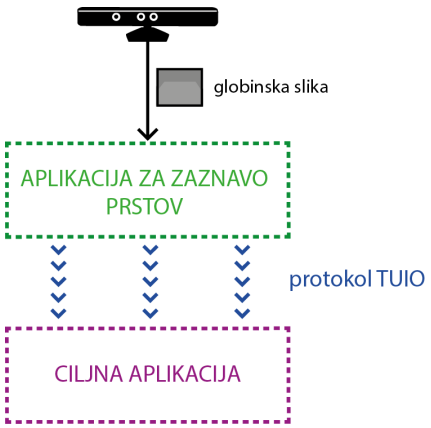
Površina je lahko poljubne barve ter vseh vrst materialov z izjemo zelo odbojnih in prosojnih. Kamera Kinect meri oddaljenost objektov z IR kamero. Odbojni in prosojni materiali projecirano IR svetlobo prekomerno oziroma prešibko odbijajo, zato kamera Kinect ni sposobna pravilno določiti oddaljenosti.



Slika 2.14: Različne orientacije opazovanih površin

## 2.5 Programske komponente

Programski del razvitega sistema je sestavljen iz aplikacije za zaznavanje prstov in ciljne aplikacije (slika 2.15). Aplikacija za zaznavo prstov skrbi za zaznavanje in sledenje uporabnikovih prstov, s pomočjo podatkov pridobljenih s kamere Kinect. Postopek je podrobneje opisan v poglavju 3. Lokacije zaznanih prstov posredujemo ciljni aplikaciji s pomočjo protokola TUIO (poglavje 4.1). Ciljna aplikacija, ki jo prikazujemo na opazovani površini s pomočjo projektorja, je lahko bodisi posamezna aplikacija, ogrodje aplikacij ali poseben gonilnik. Podrobneje jih bomo predstavili v poglavju 4.2. Končni cilj je omogočiti uporabniku ustrezno interakcijo s ciljno aplikacijo.



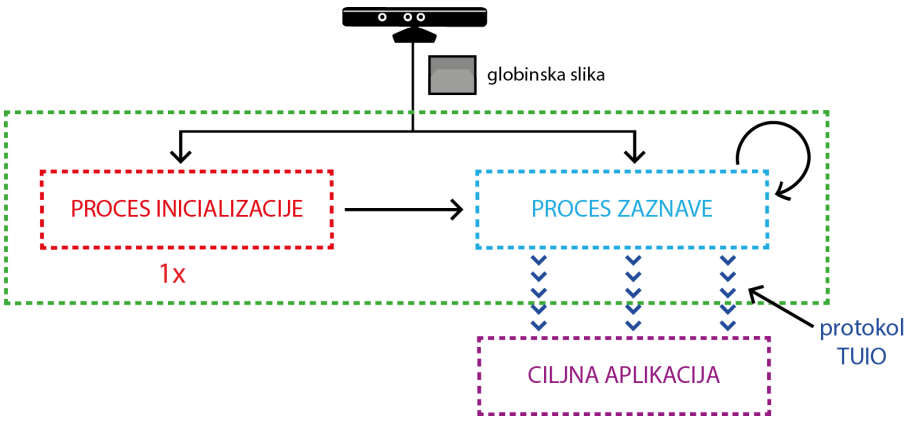

Slika 2.15: Shema programskih komponent

# Poglavje 3

# Zaznavanje prstov

V tem poglavju bomo opisali delovanje aplikacije za zaznavanje prstov. Njen cilj je natančna detekcija in robustno sledenje uporabnikovim prstom ter posredovanje njihovih lokacij ciljni aplikaciji. Delovanje lahko razdelimo na dva glavna procesa: **a.)inicializacija** in **b.)zaznavanje**.

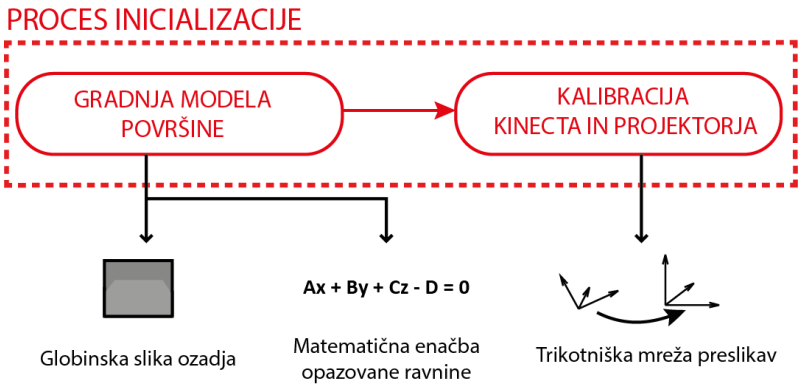
V *procesu inicializacije* je naš cilj določitev ustreznega modela ozadja ter pravilna kalibracija kamere Kinect s projektorjem. Proces opravimo le enkrat, na začetku delovanja. V nasprotju z njim se *proces zaznave* neprestano izvaja. Z uporabo globinske slike ter podatkov, pridobljenih v procesu inicializacije, zaznamo, preslikamo in posredujemo lokacije prstov. Zgradbo aplikacije za zaznavo prstov prikazuje slika 3.1.




Slika 3.1: Shema procesov aplikacije za zaznavo prstov

## 3.1 Inicializacija sistema

Pogoj za pravilno in natančno delovanje celotnega sistema je primerna inicializacija. Proces zajema izgradnjo modela površine, ki ji želimo dodati funkcionalnost večdotičnosti ter kalibracijo kamere Kinect s projektorjem (slika 3.2).




Slika 3.2: Koraki procesa inicializacije

### 3.1.1 Gradnja modela površine

Model površine je osnova za delovanja celotnega sistema. Opisuje površino, ki ji želimo dodati funkcionalnost večdotičnosti. Uporabljamo ga tako za detekcijo uporabnikovih rok (odstranjevanje ozadja), kot za preslikavo 3D pozicije prstov na opazovano površino.

Model je sestavljen iz:

- globinske slike modela ozadja;
- matematične enačbe ravnine opazovane površine in
- mej opazovane površine.

Naš sistem, za razliko od nam znanih primerljivih sistemov, poleg globinske slike uporablja tudi matematično enačbo opazovane površine ter meje s katerimi jo lahko omejimo. To nam omogoča natančen izračun oddaljenosti zaznanih uporabnikovih prstov ter odpravlja nepotrebno obdelavo podatkov, ki so izven nam zanimive površine.

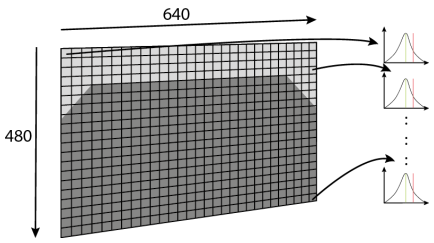
### Globinska slika modela ozadja

Globinska slika modela ozadja (slika 3.3) je referenčni podatek o sceni, ki ga sistem obdeluje. Z njegovo uporabo lahko za vsak slikovni element prejete nove globinske slike klasificiramo kot *ozadje* ali *ospredje*. S tem zmanjšamo število elementov, ki jih moramo obdelati v nadaljnjih korakih. Za pravilno delovanje skozi daljše časovno obdobje je potrebno model sproti tudi posodabljati, da se le ta uspešno prilagaja spreminjajočim se pogojem (osvetlitev, manjše spremembe v prizoru itd.).

V [21] je model ozadja zgrajen na podlagi izračunane predvidene oddaljenosti opazovane površine v posameznem slikovnem elementu globinske slike. A. Wilson je v članku [20] izpostavil problem takega pristopa, saj tako izračunane vrednosti ne odražajo dejansko zajetih vrednosti s kamero Kinect. Zajete vrednosti odstopajo od predvidenih zaradi distorzije leče ter šuma v meritvah. Predstavljena izboljšava kot model ozadja predvideva globinsko sliko prazne površine, ki jo zajamemo pred začetkom zaznavanja. V praksi se tak pristop izkaže za boljšega kot prvi, vendar ne odpravlja problema šuma meritev, ki nastane kot posledica spreminjajočih se okoljskih razmer na katere je kamera Kinect občutljiva [2].

V želji po odpravi tega problema smo v našem sistemu implementirali določitev globinske slike modela ozadja v več korakih. Pred začetkom zaznavanja zajamemo večje število globinskih slik (npr. 100) na podlagi katerih nato določimo mejne vrednosti v posameznem slikovnem elementu. S tem kar se da natančno opišemo dejansko sceno, ki jo opazujemo. Nato pa med zaznavanjem osvežujemo vrednosti posameznih slikovnih elementov in s tem zagotovimo prilagajanje modela ozadja na spreminjajoče se vrednosti meritev zaradi sprememb v okolju.

Slika 3.3: Globinska slika modela ozadja




Slika 3.4: Model ozadja, sestavljen iz posameznih Gaussovih modelov

Model ozadja sestavimo iz neodvisnih modelov, za vsakega izmed slikovnih elementov posebej (slika 3.4). Vsak tak model temelji na Gaussovi porazdelitvi zadnjih n vrednosti. V izogib potrebi po hranjenju in vsakokratnem računanju povprečne vrednosti ($\mu$) ter variance ($\sigma^2$) to opravljamo inkrementalno:

$$\mu_{t+1} = \alpha F_t + (1 - \alpha)\,\mu_t, \tag{3.1}$$

$$\sigma^2_{t+1} = \alpha\,(F_t - \mu_t)^2 + (1 - \alpha)\,\sigma^2_t, \tag{3.2}$$

kjer je $t$ časovni korak in $\alpha$ predstavlja utež, ki uravnava razmerje med stabilnostjo in hitrostjo prilagajanja modela. Njeno vrednost smo izbrali empirično (0,05).

Posodabljanje modela predstavlja ključen korak pri zagotavljanju uspešnega prilagajanja sistema na spreminjajoče se pogoje. Kot je opisano v članku [16], model vsakega elementa posodobimo v skladu z enačbama (3.1) in (3.2). V primeru ko uporabnikova roka dlje časa miruje, se s tako formulacijo posodobitve le-ta skozi čas zlije v model ozadja. Tak način delovanja za nas ni sprejemljiv, saj je roka, četudi dlje časa miruje, vseskozi naš opazovani objekt. Primerna prilagoditev procesa posodabljanja je opisana v članku [13], katerega formulacijo smo pri implementaciji uporabili tudi mi.

$$\mu_{t+1} = M\mu_t + (1 - M)\,(\alpha F_t + (1 - \alpha)\,\mu_t)\,, \tag{3.3}$$

kjer je:

$$M = \begin{cases} 0; & \text{če je element del ozadja} \\ 1; & \text{če je element del ospredja} \end{cases}$$

### Matematična enačba ravnine

Poleg informacij o porazdelitvi disperzij (globinski model ozadja), potrebujemo tudi matematični model površine, ki omogoča pravilen izračun oddaljenost ter vpliv zaznanih prstov na opazovano površino. Opazovano površino zapišemo kot matematično enačbo ravnine v prostoru:

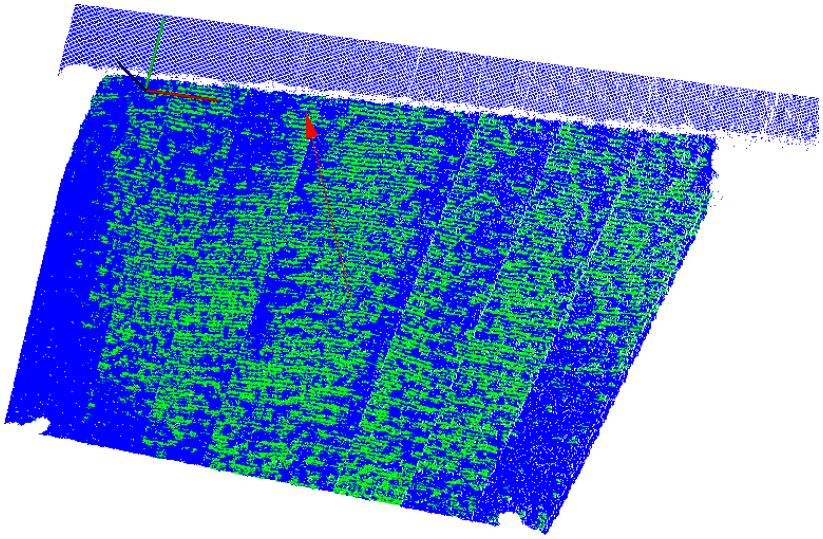
$$Ax + By + Cz - D = 0. \tag{3.4}$$

V procesu ocene parametrov najprej zgradimo globinsko sliko modela ozadja. Za izgradnjo uporabimo večje število globinskih slik (minimalno 100), da zmanjšamo vpliv majhnega nihanja disperzijskih vrednosti. Za vsak slikovni element, z uporabo enačbe (3.5), izračunamo maksimalno vrednost disperzije, ki še ustreza modelu ozadja. Element z uporabo enačb (2.5) in (2.6) preslikamo

v 3D točko ter jo dodamo v oblak 3D točk ozadja (slika 3.5). Nato parametre ravnine (A, B, C, D) ocenimo na podlagi RANSAC (angl. RANdom SAmple Consensus) metode.

$$|d_i - \mu_i| < k\sigma, \tag{3.5}$$

kjer sta k utež in $\sigma$ standardni odklon, dobljen iz Gaussovega modela posameznega slikovnega elementa.



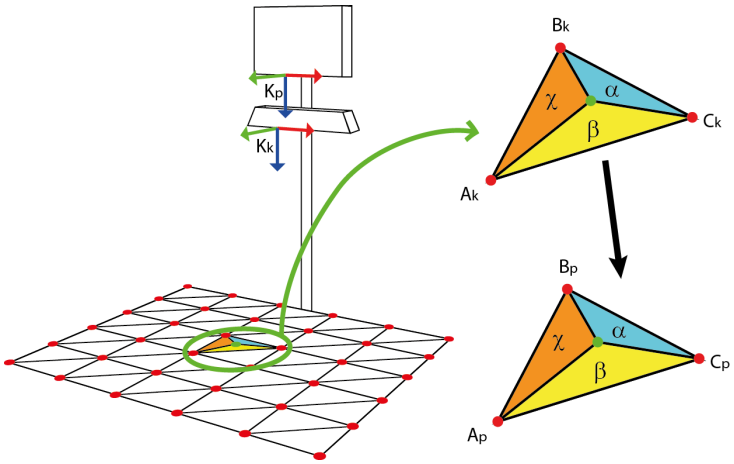
Slika 3.5: Oblak 3D točk scene z zeleno označenimi točkami, ki pripadajo opazovani ravnini

### Meje opazovane površine

V veliki večini primerov bo kamera Kinect zajela sliko širšega območja, kot je le opazovana površina. Da po nepotrebnem ne obdelujemo podatkov, ki so zunaj mej naše opazovane površine, lahko omejimo površino, ki nas pri zaznavanju zanima.

### 3.1.2 Kalibracija sistema

Kalibracija sistema je pomembna za pravilno preslikavo točk med globinskim koordinatnim sistemom kamere Kinect $K_k$ in koordinatnim sistemom ciljne aplikacije (projektorja) $K_p$. Opazovano polje ciljne aplikacije razdelimo na trikotno mrežo, katerih referenčne točke so prikazane na sliki 3.6. Uporabnik v procesu kalibracije s prstom posamezno pritiska na referenčne točke, kar sistem zazna ter njihove lokacije shrani. Točko $P_k$ preslikamo iz koordinatnega sistema




Slika 3.6: Trikotniška mreža in uporaba baricentričnih koordinat za preslikavo

kamere Kinect $K_k$ v koordinatni sistem projektorja $K_p$ s pomočjo referenčnih točk, izmerjenih v postopku kalibracije.

Najprej moramo najti ustrezen trikotnik $A_kB_kC_k$, v katerem leži točka $P_k$. Točka leži v trikotniku $ABC$, če istočasno veljajo vsi trije pogoji (slika 3.7):

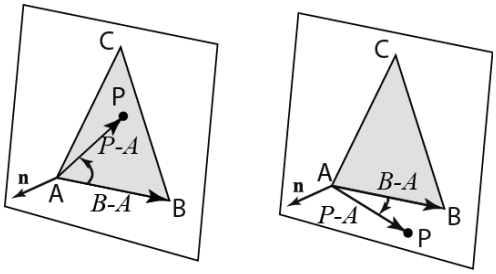
$$[(B_k - A_k) \times (P_k - A_k)] \cdot n \geq 0, \tag{3.6}$$
$$[(C_k - B_k) \times (P_k - B_k)] \cdot n \geq 0, \tag{3.7}$$
$$[(A_k - C_k) \times (P_k - C_k)] \cdot n \geq 0, \tag{3.8}$$

kjer je n normala ravnine trikotnika.

Vsako tako točko lahko zapišemo kot

$$P_k = \alpha A_k + \beta B_k + \gamma C_k, \tag{3.9}$$

kjer so $\alpha$, $\beta$, in $\gamma$ baricentrične koordinate.




Slika 3.7: Točka v trikotniku (levo) in izven trikotnika (desno)

Baricentrične koordinate točke $P_k$ izračunamo:

$$\alpha = \frac{[(C_k - B_k) \times (P_k - B_k)] \cdot n}{[(B_k - A_k) \times (C_k - A_k)] \cdot n}, \tag{3.10}$$

$$\beta = \frac{[(A_k - C_k) \times (P_k - C_k)] \cdot n}{[(B_k - A_k) \times (C_k - A_k)] \cdot n}, \tag{3.11}$$

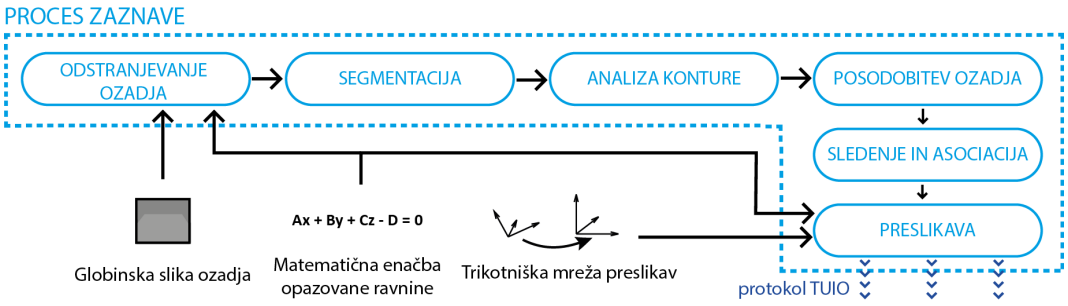
$$\gamma = \frac{[(B_k - A_k) \times (P_k - A_k)] \cdot n}{[(B_k - A_k) \times (C_k - A_k)] \cdot n}. \tag{3.12}$$

Z uporabo izračunanih baricentričnih koordinat $(\alpha, \beta, \gamma)$ in točk trikotnika (v projektorskem koordinatnem sistemu!), preslikamo točko $P_k$ v točko v koordinatnem sistemu projektorja $P_p$.

$$P_p = \alpha A_p + \beta B_p + \gamma C_p \tag{3.13}$$

## 3.2 Proces zaznave

Proces zaznave predstavlja srce razvitega sistema. S pomočjo podatkov, pridobljenih v procesu inicializacije (podpoglavje 3.1), vsako prejeto globinsko sliko primerno obdelamo in kot rezultat obdelave posredujemo lokacije zaznanih prstov ciljni aplikaciji. V procesu zaznave najprej odstranimo ozadje (podpoglavje 3.2.1). Preostali elementi ospredja velikokrat vsebujejo šum, ki ga odpravimo s pomočjo segmentacije ospredja (podpoglavje 3.2.2). Regije, ki vsebujejo premalo število elementov, zavržemo, pri preostalih pa analiziramo njihov obris (podpoglavje 3.2.3), s pomočjo katerega določimo točke, ki ustrezajo konicam uporabnikovih prstov. Detekcije prstov poskušamo smiselno povezati z detekcijami iz prejšnjih globinskih slik v želji po povezavi v smiselne kretnje (podpoglavje 3.2.4). Lokacije prstov nazadnje še preslikamo v ustrezni koordinatni sistem ciljne aplikacije (podpoglavje 3.2.5), ter jih z uporabo protokola TUIO posredujemo ciljni aplikaciji (poglavje 4.1). Shema procesa zaznave je prikazana na sliki 3.8).




Slika 3.8: Shema procesov aplikacije za zaznavo prstov

### 3.2.1 Odstranjevanje ozadja

Odstranjevanje ozadja je tehnika, ki se jo uporablja za detekcijo potencialno zanimivih objektov v prizoru, posnetem s statično kamero. Deluje na principu primerjave trenutno zajete in referenčne slike (oz. model ozadja). Področja, kjer je razlika med istoležnimi elementi dovolj velika, nakazuje možnost prisotnosti zanimivega objekta. Osnova za odstranjevanje je dober model ozadja, ki ozadje kar se da natančno opisuje (tj. prizor brez objektov, ki nas zanimajo). Model je potrebno sproti tudi posodabljati, da se le-ta uspešno prilagaja spreminjajočim se pogojem (osvetlitev, manjše spremembe v prizoru itd.).

V podpoglavju 3.1.1 je opisan model ozadja, ki ga uporabljamo v razvitem sistemu.

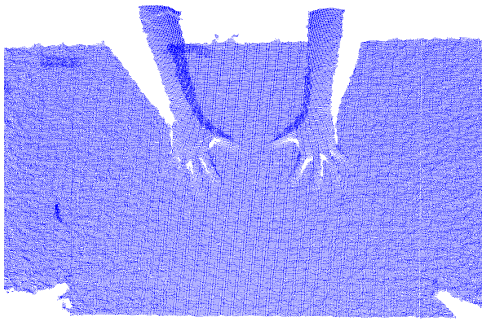
Osnova metode odstranjevanja ozadja je enačba:

$$|B_t - I_t| > Thr, \tag{3.14}$$

kjer je $B_t$ model ozadja, $I_t$ trenutna slika in Thr v naprej določen prag.

M. Piccardi je v članku [16] analiziral prednosti in slabosti različnih metod za odstranjevanje ozadja. Na podlagi analize smo se odločili za uporabo *Running Gaussian Average* metode, katere glavni prednosti sta časovna in prostorska nezahtevnost. V članku je kot njena glavna pomanjkljivost izpostavljena srednja oziroma slaba natančnost, kar pa ne velja za uporabo z globinskimi slikami. Odstranjevanje poteka na podlagi vrednosti disperzij, ki so veliko manj občutljive na okoljske spremembe kot barve na osvetlitev, ki so bile uporabljene za analizo v članku. Tako so rezultati odstranjevanja za nas povsem zadovoljivi.

Slikovne elemente prejete globinske slike klasificiramo na ospredje in ozadje na podlagi enačbe (3.14). Sliki 3.9 in 3.10 prikazujeta oblaka 3D točk pred in po odstranitvi ozadja. Na sliki 3.10 so vidni tudi osameli elementi - šum. Ti so posledica nihanja disperzij in občutljivosti meritev globinske kamere Kinect na okoljske pogoje [2]. Taki elementi so bili ob klasifikaciji s pomočjo enačbe (3.14) le malo nad pragom TH.



Slika 3.9: Oblak 3D točk zajete scene

Slika 3.10: Oblak 3D točk po odstranitvi ozadja

Šum v veliki večini primerov uspešno odpravimo s segmentacijo ospredja. Ospredje razdelimo na povezane regije, pri čemer regije s premajhnim številom elementov označimo kot šum in zavržemo. Slika 3.13 prikazuje oblak 3D točk po odstranitvi šuma.

### 3.2.2 Segmentacija

Segmentacija slike je proces delitve slikovnih elementov na segmente (regije), ki najverjetneje ustrezajo objektom oz. območjem iz realnega sveta. Segmenti so homogeni glede na določen opazovan kriterij (robovi, barva, tekstura, evklidska razdalja ...), istočasno pa se od sosednjih regij dovolj razlikujejo. Pomembno je določiti ustrezen prag delitve med regijami, saj lahko prenizko postavljen prag povzroči zlitje, previsok prag pa nezaželeno drobljenje (fragmentacijo) regij. Segmentacija ne pomeni tudi klasifikacije, tako da so dobljeni segmenti le deli celotne slike (kot deli sestavljanke) brez dodatnega novega znanja o njihovem kontekstu.

Delitev elementov na segmente se v večini primerov opravlja na podlagi

- *zaznavanja diskontinuitet* lastnosti sosednjih elementov (razdelitev slike glede na spremembo v opazovanem kriteriju (segmentacija na osnovi robov);

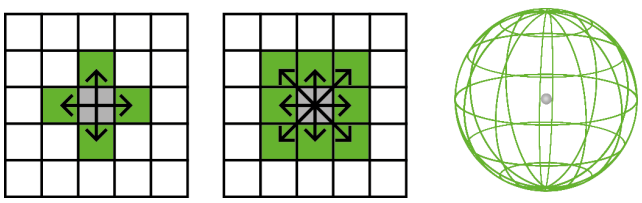
- *zaznavanja podobnosti* med sosednjimi elementi (razdelitev slike glede na podobnost v opazovanem kriteriju - segmentacija na osnovi regij in pragovna segmentacija).

Segmentacija na osnovi regij gradi regije neposredno iz slikovnih elementov, glede na podobnost s sosednjimi elementi. Homogenost posamezne regije je najbolj pomemben segmentacijski kriterij, ki je lahko sestavljen iz enega ali več podkriterijev (barva, prostorska homogenost itd.). Segmentacijske metode na osnovi regij so manj občutljive na pojavljanje šuma kot metode na osnovi robov ali pragovna segmentacija.

Proces rasti regij se začne z izborom začetnega semena in segmentacijskega kriterija. V postopku rasti regije preverimo vse sosednje elemente trenutnega semena in njihovo ustreznost glede na izbran kriterij. V primeru, da so ti dovolj podobni semenu, regijo razširimo na nove elemente. Postopek ponavljamo, dokler je širjenje trenutne regije mogoče. V nasprotnem primeru izberemo novo začetno seme iz slikovnih elementov, ki še ne pripadajo kateri izmed regij. Ko vsi slikovni elementi pripadajo eni izmed regij, se segmentacija zaključi.

S segmentacijo elemente razdelimo na posamezne objekte zanimanja. Za segmentacijo uporabimo samo elemente, ki smo jih po odstranitvi ozadja označili kot ospredje. S tem bistveno pohitrimo delovanje, saj nam elementov ozadja ni potrebno obravnavati. Regije, ki vsebujejo premalo elementov, označimo kot šum in zavržemo. S tem odpravimo veliko večino šuma, ki ostane po postopku odstranjevanja ozadja (slika 3.10).

Sosednji elementi v 2D slikah so največkrat 4- ali 8-strani (slika 3.11), medtem ko so v 3D koordinatnem sistemu sosedi vsi elementi, ki so od opazovanega oddaljeni manj kot določena razdalja. Določanje sosednjih elementov v 3D prostoru je računsko izjemno potratno in bi za delovanje v realnem času zahtevalo uporabo računalnika s podporo GPE. Zaradi naše želje po možnosti uporabe na računalnikih brez take podpore, smo se odločili za uporabo 8-stranih sosednjih elementov. Pri tem je potrebno poudariti, da taka uporaba ne povzroča nobenega dodatnega poslabšanja rezultatov segmentacije, dokler se pri določitvi segmentacijskega kriterija upošteva omejitve 3D kamere. Minimalna razdalja med sosednjima elementoma je določena z ločljivostjo globinske slike ter oddaljenostjo kamere od objekta.

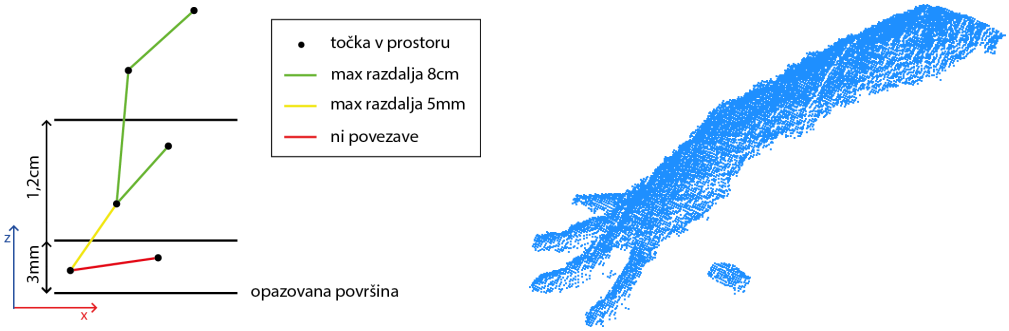


Slika 3.11: 4-strani sosedi (levo), 8-strani sosedi (sredina), mejna razdalja v 3D prostoru(desno)

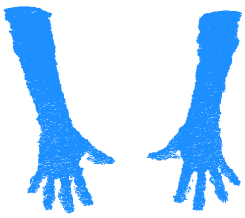
Izbira segmentacijskega kriterija je zelo pomembna, saj ta določa način združevanja elementov v homogene regije. Naš segmentacijski kriterij je sestavljen iz več kriterijev, prikazanih na sliki 3.12(levo), kar nam zagotavlja večjo robustnost sistema.

V želji po čim bolj robustnem sistemu, je razdalja med točkama, ki ležita vsaj 1,2 cm nad površino, lahko do 8 cm. S tem je omogočena pravilna segmentacija delno zakritih ali skoraj navpično postavljenih prstov s pripadajočo roko (slika 3.12 desno). Hkrati pa se točki, ki obe ležita manj kot 3 mm nad površino, ne morata povezati, saj je to v večini primerov le šum v okolici konice prstov.

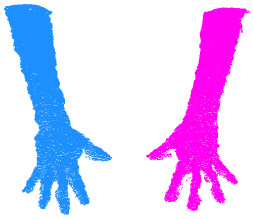
Slika 3.14 prikazuje končni rezultat segmentacije. Množica vsebuje segmentirane regije, za katere predpostavljamo, da so uporabnikove roke. V vsaki izmed regij želimo zaznati vse morebitne uporabnikove prste. Z analizo obrisa posamezne regije določimo točke, ki ležijo na predelih obrisa, podobnih obliki konic prstov. Naš algoritem je sposoben zaznati katero koli kombinacijo prstov z visoko natančnostjo ob majhni računski zahtevnosti.




Slika 3.12: Največje še dovoljene razdalje med točkami pri segmentaciji (levo); pravilna segmentacija, kljub veliki oddaljenosti kazalca (desno)



Slika 3.13: Oblak 3D točk po odstranitvi šuma



Slika 3.14: Končni rezultat segmentacije ospredja (različne barve pomenijo različne regije)

### 3.2.3 Analiza obrisa

Zaznava konic uporabnikovih prstov je zaradi različnih oblik in dimenzij prstov še vedno relativno velik problem, predvsem za zaznavo v realnem času brez ustrezne podpore GPE. V [14] je predlagana zaznava z uporabo več zaporednih testov. Konice uporabnikovih prstov morajo tako biti del dovolj velike regije povezanih elementov, obdane s polnim krogom točk iste regije premera $9mm$, krožnica s premerom $20mm$ in središčem v opazovani točki pa mora regijo prečkati točno dvakrat na razdalji, ki ustreza širini prsta. V [17] je opisana zaznava z analizo oddaljenosti točk obrisa regije od središča uporabnikove dlani, medtem ko [18] predlaga zaznavo z zaporednim krčenjem in širjenjem opazovane regije.

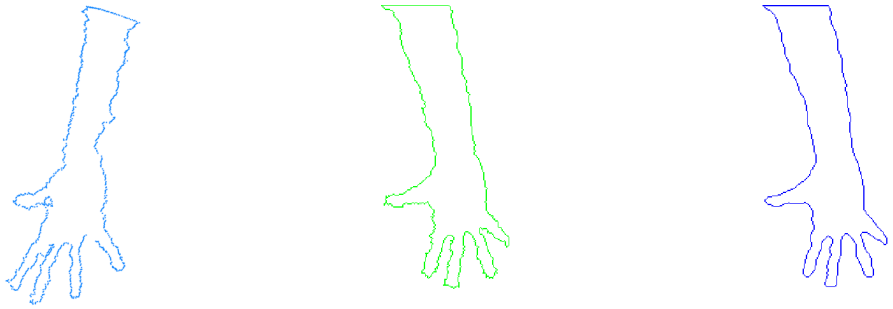
Vsi omenjeni pristopi delujejo dobro, vendar za delovanje v realnem času potrebujejo podporo GPE, ki pa smo se ji mi zavestno odpovedali. Zato

smo v našem sistemu uporabili metodo *k-curvature* [19]. V članku je metoda uporabljena le na barvnih slikah, kar pa za nas ne predstavlja problema, saj je delovanje enako, tako v barvnih slikah kot na 3D točkah. Med implementacijo se je izkazalo, da je v obrisu regije zajete s kamero Kinect (slika 3.15 (levo)), preveč majhnih napak, ki so posledica šuma pri zajemu globinske slike. V izogib napačni analizi, obris preslikamo nazaj v 2D s pomočjo enačbe (2.6), kjer z uporabo koordinat točke $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ izračunamo 2D koordinate točke $(\mathbf{u}, \mathbf{v})$. Preslikan obris (prikazan na sliki 3.15 (sredina)) nato z uporabo povprečnega filtra z dimenzijo maske $7 \times 7$ zgladimo. Nov zglajen obris določimo z uporabo pragovne segmentacije. Končni rezultat je prikazan na sliki 3.15 (desno).

Za točke, ki ležijo na zanimivih delih obrisa, velja:

$$\cos\theta < \phi, \tag{3.15}$$

kjer je kot $\theta$ definiran kot kot med opazovano točko $P_i$ in točkama $P_{i-k}$ ter $P_{i+k}$. Točki sta od $P_i$ oddaljeni za k korakov po obrisu, $\phi$ pa predstavlja v naprej določen prag velikosti kota.



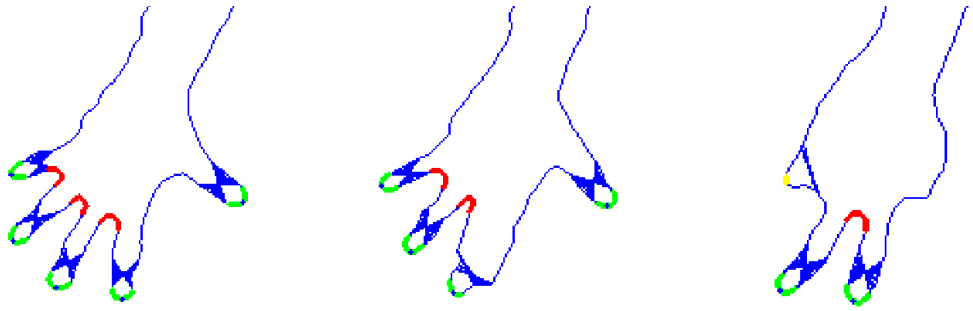
Slika 3.15: Obris v 3D (levo), preslikan obris v 2D (sredina), zglajen obris v 2D (desno)

Kot $\theta$ izračunamo s pomočjo skalarnega produkta vektorja $\langle P_{i-k}, P_i\rangle$ in $\langle P_{i+k}, P_i\rangle$:

$$\cos\theta = \frac{\langle P_{i-k}, P_i\rangle \cdot \langle P_{i+k}, P_i\rangle}{||\langle P_{i-k}, P_i\rangle||\,||\langle P_{i+k}, P_i\rangle||} \tag{3.16}$$

Točke, ki ustrezajo pogoju (3.15), ležijo bodisi na konicah prstov bodisi na krivulji med prstoma. Med njimi lahko razlikujemo s pomočjo predznaka koordinate $z$ vektorskega produkta $\langle P_{i-k}, P_i\rangle \times \langle P_{i+k}, P_i\rangle$. Točke na konicah prstov imajo koordinato $z$ pozitivno. Take točke združujemo v povezane krivulje, glede na zaporednost na obrisu. Premajhna zaporedja točk zavržemo, preostalim pa določimo točko interakcije in jih poskusimo asociirati s predhodnimi detekcijami.

Slika 3.16 prikazuje detekcijo točk na prstih v različnih pozicijah. Točke, ki ustrezajo pogoju (3.15), imajo koordinato $z$ vektorskega produkta $\langle P_{i-k}, P_i\rangle \times \langle P_{i+k}, P_i\rangle$ pozitivno in so povezane v dovolj dolga zaporedja, označimo z zeleno. Točke, ki smo jih zavrnili zaradi prekratkega zaporedja, so označene z rumeno, z rdečo pa so označene točke, katerih koordinata $z$ vektorskega produkta je negativna. Z modro črto sta povezani točki $P_{i-k}$ in $P_{i+k}$ vsake izmed točke $P_i$, ki ustreza pogoju (3.15). Na srednji sliki lahko vidimo, da sistem uspešno prepozna kazalec in sredinec, ki sta tesno skupaj kot en sam prst.



Slika 3.16: Zaznane točke konic prstov v različnih položajih roke

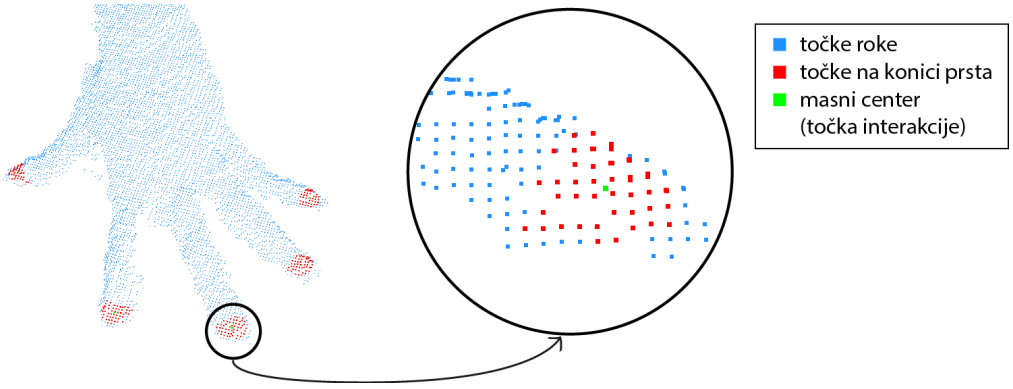
Na podlagi preučitve geometrije roke in prstov ter večkratnih testov pri naši postavitvi sistema (slika 5.2) smo ugotovili, da je primerna razdalja med opazovano točko in k-tima sosedama 16 korakov, kosinus kota pa mora biti večji od praga 0, kar ustreza $\theta = 90°$.

Točko interakcije prsta s sistemom določimo kot masni center točk, ki ležijo na območju omejenem z zaznano krivuljo. Slika 3.17 prikazuje točke na območjih zaznanih krivulj (označene rdeče) in masni center vsakega izmed njih (zelena točka).

### 3.2.4 Sledenje in asociacija

Z uspešno detekcijo vseh uporabnikovih konic prstov pridobimo vse točke interakcije uporabnika s sistemom. Sledenje prstom skozi čas je nujno potrebno, če želimo uporabniku omogočiti interakcijo s ciljnimi aplikacijami s pomočjo kretenj. Tako moramo vsak prst bodisi povezati (asociirati) z detekcijo s predhodne globinske slike bodisi ga označiti kot novega. Poleg tega moramo primerno označiti tudi vse prste, ki jih glede na predhodno stanje nismo več zaznali.

Večino časa bomo želeli istočasno slediti več prstom. Naša metoda je implementirana tako, da za vsakega izmed prstov uporabimo svoj sledilnik, ki




Slika 3.17: Točke interakcije s sistemom kot masni centri točk na območju krivulj

ga hranimo skozi čas. Ob detekciji novega prsta ustvarimo nov sledilnik, ko pa prsta ne zaznamo več, njegov sledilnik preprosto zavržemo. Za zmanjšanje lažnih detekcij uporabimo omejitev, da moramo prst zaznati v vsaj dveh zaporednih časovnih korakih, da ga upoštevamo kot pravilno detekcijo.

### Sledilnik

Sledenje prsta skozi čas je proces določanja njegove lokacije v zaporednih časovnih razmikih. V optimalnih pogojih bi bil algoritem sposoben slediti prstu kjer koli v prostoru v vsakem danem trenutku. Preprosta rešitev je uporaba iskalnega okna, ki obsega določeno okolico okoli zadnje znane pozicije objekta. Tak pristop se izkaže za nezanesljivega, saj je neuspešen, če se prst pojavi izven iskanega okna. To se lahko zgodi zaradi *a*) prehitrega premika; *b*) premajhnega iskalnega okna ali *c*) prenizke zajemne hitrosti sličic kamere (angl. frame rate).

Rešitev bi bila uporaba visoko zmogljivostne kamere, ki bi omogočala zajemanje podatkov pri visokih frekvencah, vendar so take 3D kamere izjemno drage. Dodaten problem predstavlja tudi nezanesljiva določitev lokacije prsta, ki je posledica šuma pri zajemu podatkov.

Težave rešimo z uporabo Kalmanovega filtra (krajše KF) [9]. Omogoča nam predikcijo lokacije prsta v naslednjem časovnem koraku, kar uporabimo kot središče iskalnega okna. Z njegovo uporabo odpravimo tudi posledice šuma meritev. Delovanje KF je bolj podrobno opisano v dodatku A.

Ker sledenje poteka v 3D prostoru (XYZ), katerega dimenzije so med seboj

neodvisne, KF uporabimo za predikcijo lokacije v vsaki izmed dimenzij posebej. S tem zmanjšamo dimenzije matrik, uporabljenih v posameznem filtru, in posledično zmanjšamo skupno računsko zahtevnost izračunov filtra.

Asociacijo parov zaznanih prstov (ob času t) in predhodno zaznanih prstov (ob času t-1) naredimo na podlagi **detekcij** $T^{(d)}$ (lokacija prstov - ob času t) in **predikcij** sledilnikov prstov $T^{(p)}$ - ob času t-1. Uporabimo metodo suboptimalnih najbližjih sosedov z lokalno optimizacijo razdalj ter uporabo validacijskih vrat. Validacijska vrata predstavljajo še največjo sprejemljivo razdaljo med predikcijo in detekcijo, ki omogoča njuno povezavo.

Matriko cen povezav C = $[d_{ij}]$ sestavimo kot:

$$d_{ij} = \left|\left| T_i^{(d)} - T_j^{(p)} \right|\right| \tag{3.17}$$

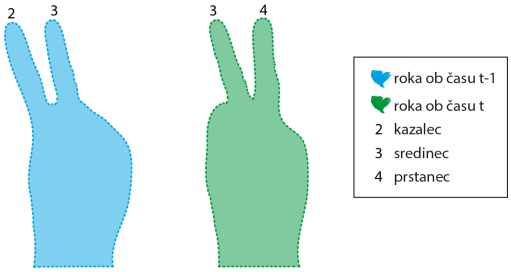
Dokler v matriki cen C obstaja element $d_{ij}$ z nižjo ceno kot je meja validacijskih vrat, izberemo najmanjšo ter povežemo pripadajočo detekcijo in predikcijo. Istočasno iz matrike odstranimo vrstico ter stolpec, ki pripada tej povezavi. Detekcije, ki jih nismo uspeli povezati, označimo kot na novo zaznane prste, ki jim bomo od tega trenutka naprej sledili.

Lokalno optimizacijo razdalj povezav smo uporabili, ker se je ta, v primerjavi z globalno, izkazala za bolj primerno, v primerih ko v istem trenutku en prst izgine ter se drugi pojavi na relativno majhni razdalji. Slika 3.18 prikazuje položaj roke v dveh zaporednih časovnih korakih, kjer v času enega koraka kazalec dvignemo in prstanec spustimo. V želji po čim večjem številu povezav (in s tem čim nižji skupni ceni povezav) prihaja pri globalni optimizaciji do napačnih asociacij detekcij in predikcij, kot je prikazano na sliki 3.19 (levo). Uporaba lokalne optimizacije ni primerna, ko predikcije niso natančne, kar pa v razvitem sistemu ni velik problem. Slika 3.19 (desno) prikazuje pravilno asociacijo z uporabo lokalne optimizacije.

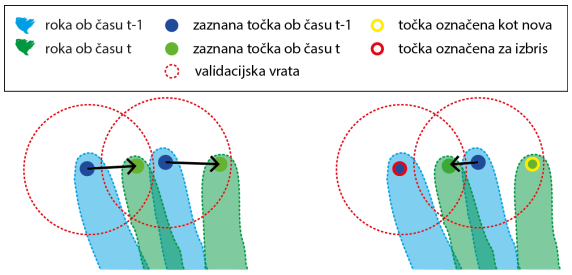
### 3.2.5 Preslikava med koordinatnimi sistemi

Uspešna asociacija lokacij prstov s predhodno zaznanimi prsti nam še ne omogoča izpolnitve končnega cilja - omogočiti večdotično funkcionalnost opazovane površine. Lokacije prstov, ki smo jih zaznali v globinskem koordinatnem sistemu kamere Kinect, nam ne dajo pravih informacij o vplivih prstov na opazovano površino. Prave točke vplivov dobimo s pravokotno projekcijo prstov na površino. Pri tem uporabimo matematične enačbo ravnine (3.4), ki smo jo določili v procesu inicializacije.

Tako dobljene točke so še vedno v globinskem koordinatnem sistemu kamere Kinect, tako da je pred posredovanjem ciljni aplikaciji potrebna še preslikava




Slika 3.18: Roka v dveh časovno zaporednih korakih (modra od času $t-1$ in zelena ob času $t$)




Slika 3.19: Asociacija z globalno optimizacijo (levo) in lokalno optimizacijo (desno)

v ustrezen koordinatni sistem ciljne aplikacije. Preslikavo med koordinatnima sistemoma naredimo z metodo, opisano v podpoglavju 3.1.2.

Točke, preslikane v ustrezen koordinatni sistem, posredujemo ciljni aplikaciji s pomočjo protokola TUIO.

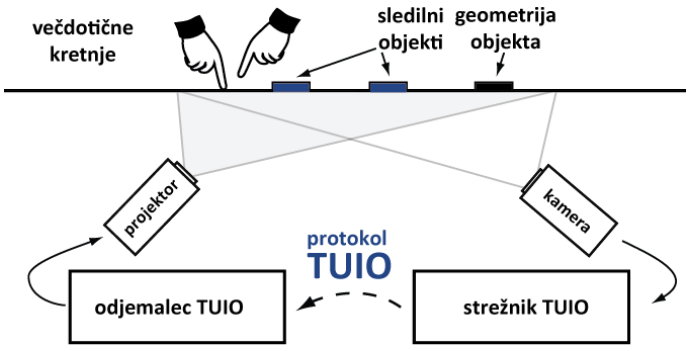
# Poglavje 4

# Programski vmesnik

Kot prikazuje slika 2.15 so programske komponente razdeljene na aplikacijo zaznavanja uporabnikovih prstov in ciljno aplikacijo. Medtem ko smo aplikacijo zaznavanja prstov opisali v poglavju 3, bomo v tem poglavju predstavili možne ciljne aplikacije ter protokol TUIO, ki skrbi za prenos informacij o zaznanih uporabnikovih prstih med obema aplikacijama.

## 4.1 Protokol TUIO

Protokol TUIO je nastal v želji po standardizaciji prenosa podatkov med sledilnikom (strežnikom) in ciljno aplikacijo (odjemalcem). Protokol definira lastnosti zaznanih objektov (pozicija, orientacija itd.) ter postopek medsebojne komunikacije. Za prenos podatkov uporablja protokol OSC (OpenSound Control), kar omogoča uporabo na vseh platformah.




Slika 4.1: Protokol TUIO

Za delovanje uporablja dva glavna tipa sporočil – **SET** in **ALIVE**. Sporočila tipa SET so namenjena sporočanju sprememb lastnosti zaznanih objektov, sporočila ALIVE pa pošiljanju seznama trenutno še aktivnih (zaznanih) objektov. Poleg omenjenih, protokol uporablja še tip sporočil **FSEQ** za identifikacijo vsakega izmed korakov posodobitve. V želji po čim manjših zakasnitvah pri prenosu podatkov, protokol predvideva uporabo transportnega protokola UDP (User Datagram Protocol).

Izgubo ali zakasnitev paketa protokol rešuje s pošiljanjem redundantnih podatkov (dodajanje sporočila ALIVE vsakemu izmed poslanih paketov UDP). Poleg sporočila ALIVE, posamezni paket UDP vsebuje tudi sporočila SET in FSEQ. Uporaba FSEQ omogoča odjemalcu preprosto zaznavo zapoznelega paketa. V izogib napakam, protokol ne predvideva eksplicitnih sporočil ADD in REMOVE. Odjemalec sam, na podlagi ugotovljenih sprememb, med zaporednimi sporočili ALIVE zazna dodajanje in brisanje objektov.

V času pisanja aktualna specifikacija protokola TUIO (verzija 1.1) [10] definira tri različne tipe opisov objektov. Razlikujejo se po lastnostih, ki jih lahko uporabimo za opis:

- **object**: identificiran objekt z znano pozicijo in orientacijo;
- **cursor**: identificiran objekt z znano pozicijo;
- **blob**: neidentificiran objekt z znano pozicijo in orientacijo.

Poleg različnih tipov objektov, protokol definira tudi različne profile, ki omogočajo njihov opis v skladu s številom dimenzij (2D, 3D), v katerem jih opazujemo.

### 4.1.1 2D ali 3D

Zaradi izjemno majhnega števila 3D strežnikov TUIO, je temu primerno tudi majhno število ciljnih aplikacij, ki omogočajo primerno izrabo 3D informacij. V želji po kar se da široki možnosti uporabe, smo implementirali tako uporabo v 2D kot 3D prostoru. Pri uporabi v 2D prostoru gre pravzaprav le za zaznavanje prstov, katerih konice se nahajajo pod določenim pragom nad površino.

### 4.1.2 Implementacija

V razvitem sistemu smo uporabili C++ implementacijo strežnika TUIO s spletne strani [38]. Za opis prstov, smo uporabili tip TUIO CURSOR, saj njihova uporaba omogoča poleg pošiljanja pozicije tudi pošiljanje identitete.

Ta omogoča ciljnim aplikacijam detekcijo uporabnikovih kretenj. Vsak sledilnik prsta vsebuje informacijo o objektu TUIO, ki ga po vsakokratno končani obdelavi globinske slike tudi primerno posodobi oz. označi za izbris, če prsta nismo več zaznali.

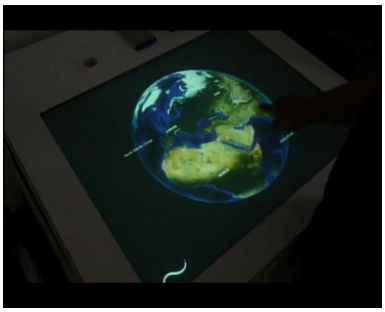
## 4.2 Ciljne aplikacije

Ciljne aplikacije so aplikacije, ki jih želi uporabnik preko razvitega sistema upravljati. Zaznana uporabnikova dejanja so posredovana ciljnim aplikacijam preko protokola TUIO. Ciljna aplikacija deluje kot odjemalec TUIO, sledilnik pa kot strežnik TUIO. Ciljna aplikacija je lahko:

- posamezna aplikacija;
- ogrodje aplikacij;
- gonilnik.

### 4.2.1 Posamezna aplikacija

Obstaja veliko število zelo različnih samostojnih aplikacij, ki so v veliki večini delo amaterskih entuziastov. Med bolj zanimivimi je aplikacija Community Earth [23]. Z uporabo ogrodja NASA World Wind Java SDK [31] omogoča uporabniku prikaz zemeljske površine z različnih perspektiv. S preprostimi kretnjami lahko uporabnik spreminja pogled na Zemljo. Ogleda si jo lahko s perspektive oddaljenega satelita, ali pa se povsem približa in vidi pokrajino v 3D kot da bi bil dejansko tam.

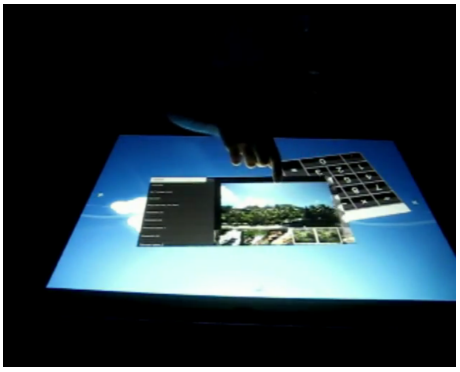


Slika 4.2: Community Earth [23]

### 4.2.2 Ogrodje aplikacij

Poleg posameznih aplikacij obstajajo tudi ogrodja, ki skrbijo za pretvorbo podatkov, pridobljenih iz sledilnikov, v signale in tako omogočijo razvijalcem, da se osredotočijo le na razvijanje aplikacij. Eden takih primerov je ogrodje Glassomium [27]. Glassomium je odprtokodni upravljavec oken s podporo protokola TUIO. Narejen je na podlagi Google Chrome brskalnika. Podpira aplikacije v programskih jezikih HTML5, Javascript, Flash, PHP itd.

Slika 4.3: Primer aplikacije za gledanje slik (vzeto iz [27])



Slika 4.4: Primer uporabe pri več aplikacijah (vzeto iz [27])

### 4.2.3 Gonilnik

Poseben gonilnik, ki emulira delovanje miške, nam omogoča, da lahko uporabnik upravlja z vsako aplikacijo, kot bi jo upravljal z miško. Gonilnik podatke, dobljene preko protokola TUIO, pretvori v miški podobne dogodke. Za operacijski sistem Windows 7 je namenjen gonilnik UniSoftHID, ki skupaj s programom Multi-Touch Vista [30] omogoča osnovno emulacijo miške ter večdotično interakcijo, vendar le v programih, ki to omogočajo.

# Poglavje 5

# Ovrednotenje delovanja sistema

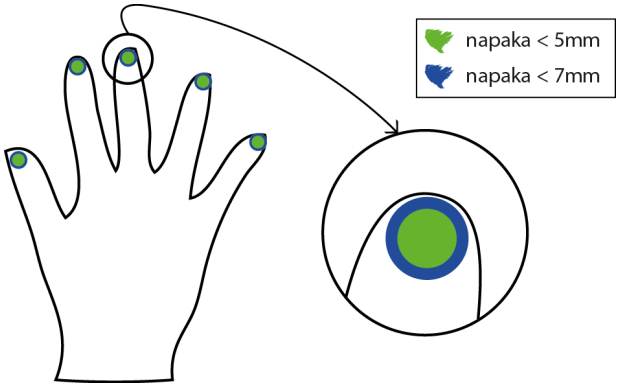
Delovanje razvitega sistema lahko ocenimo na podlagi več kriterijev. Najbolj pomembna sta zagotovo:

- **natančnost zaznave**: Cilj sistema je zaznati sredino nohtne regije prsta, ki je v povprečju širok od $1,5$ do $2cm$. Uporabniku lahko zagotovimo dobro uporabniško izkušnjo le, če bo sistem zaznal mesto dotika v območju uporabnikovega prsta. Zato lahko kot dovolj natančno zaznavo označimo le tako, ki je od dejanskega središča prsta oddaljena za manj kot $0,5cm$. Območja natančnosti so prikazana na sliki 5.1.
- **odzivni čas**: Odzivni čas sistema je čas, ki preteče od dejanske uporabnikove akcije, do zaznave v sistemu. Čas je sestavljen iz časov zajema in obdelave slike ter časa pošiljanja in prikaza posledic uporabnikovega dejanja v ciljni aplikaciji.

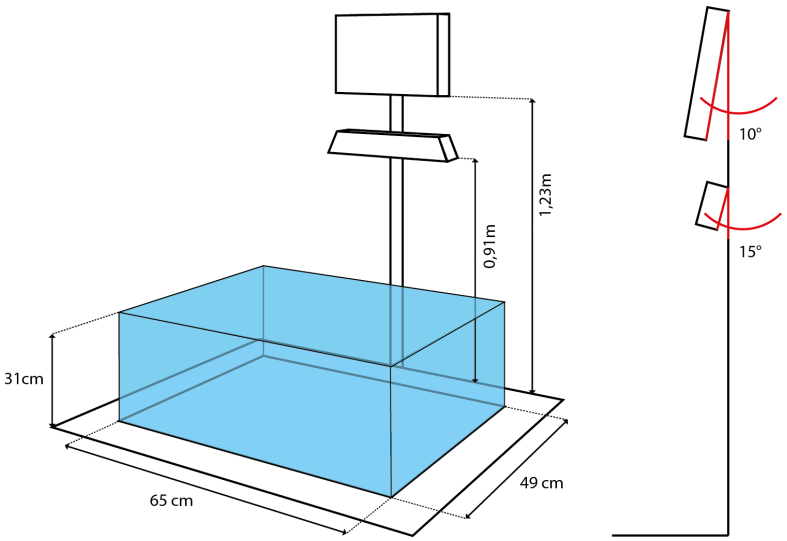
Zakasnitvi zaradi zajema in prikaza posledic sta neodvisni od našega algoritma. Čas zajema in prenosa slike je povsem odvisen od strojnih zmogljivosti kamere Kinect. Čas ni bil nikoli uradno potrjen, vendar je po neuradno dostopnih informacijah [26] približno $90ms$. Čeprav nanju nimamo vpliva, imata pomembno vlogo pri skupnem času zakasnitve sistema in s tem uporabniške izkušnje pri uporabi.

## 5.1 Postavitev

Sistem smo ocenili pri postavitvi, prikazani na sliki 5.2. Poleg kamere Kinect smo pri oceni uporabili še projektor Dell 3400MP DLP in osebni računalnik. Njihove specifikacije so prikazane v tabeli 5.1.




Slika 5.1: Območji natančnosti (zelena - $0,5cm$, modra - $0,7cm$)




Slika 5.2: Postavitev sistema pri evaluaciji

Ker je velikost projicirane slike s projektorja manjša od površine, ki jo lahko opazuje kamera Kinect, smo se odločili omejiti na površino projicirane slike. Ob predpostavki, da projektor ne bi bil ovira, bi lahko opazovali površino velikosti $80cm \times 69cm$.

| **Microsoft Kinect For Xbox 360** | |
|---|---|
| globinska kamera: | $640 \times 480$ sl. el. (30 slik na sekundo) |
| (ločljivost slike / hitrost zajema) | |
| območje delovanja: | od $0,4m$ do $6m$ |
| vidni kot: | horizontalni 53° in vertikalni 43° |

| **Računalnik** | |
|---|---|
| matična plošča: | ASUS P8H67-M PRO |
| procesor: | Intel® Core™ $i5 - 2320@3.00GHz \times 4$ |
| RAM: | $8GB$ |
| OS: | Ubuntu 12.04 (precise) 64-bit |

| **Projektor Dell 3400 MP DLP** | |
|---|---|
| max ločljivost: | $1400px \times 1050px$ |
| velikost slike na podlagi: | $65cm \times 49cm$ |

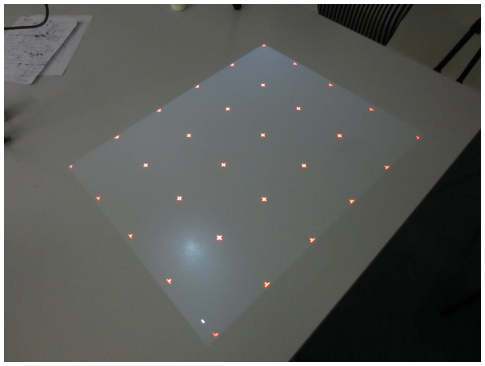
Tabela 5.1: Specifikacije strojnih komponent

## 5.2 Ocenjevalni protokol

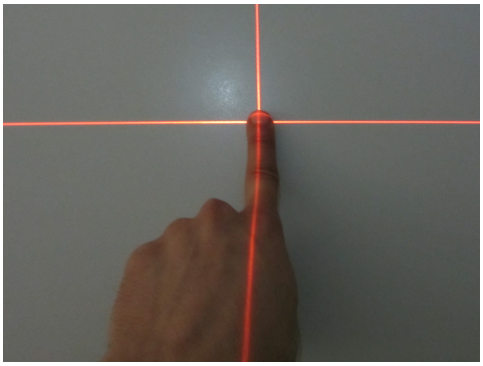
Najprej smo izvedli kalibracijo sistema z mrežo točk velikosti $6 \times 5$, kot je prikazana na sliki 5.3. Postopek kalibracije je bolj podrobno opisan v podpoglavju 3.1.2. Nato so se uporabniku posamezno prikazovale točke, na katere je moral pritisniti. Na sliki 5.4 je prikazan trenutek, ko uporabnik pritisne na prikazano lokacijo točke (stičišče rdečih črt). Napako v zaznavi predstavlja razdalja med lokacijo prikazane točke ter lokacijo zaznave uporabnikovega dotika.

Ocenjevanje natančnosti sistema smo izvedli v dveh delih. Najprej smo natančnost preverili v težišču (baricentru) vsakega izmed trikotnikov, ki sestavljajo mrežo za preslikavo. Baricentrične koordinate teh točk so $(1/3, 1/3, 1/3)$. Take točke smo izbrali, ker predstavljajo povprečno napako v posameznem trikotniku. V drugem delu pa smo natančnost preverili v štirih naključnih točkah vsakega izmed trikotnikov.

Celotno zakasnitev sistema smo izmerili s pomočjo kamere, ki je sposobna zajema 60 slik na sekundo. Empirično smo določili trenutek dejanskega premika uporabnika in trenutka, ko je sistem premik zaznal in prikazal spremembo. Čas obdelave posamezne slike pa smo izmerili programsko in ga



Slika 5.3: Kalibracijske točke na opazovani površini



Slika 5.4: Uporabnik med evaluacijo sistema

določili kot čas od trenutka, ko je sistem prejel globinsko sliko, do trenutka, ko je poslal informacijo preko protokola TUIO.

## 5.3 Rezultati

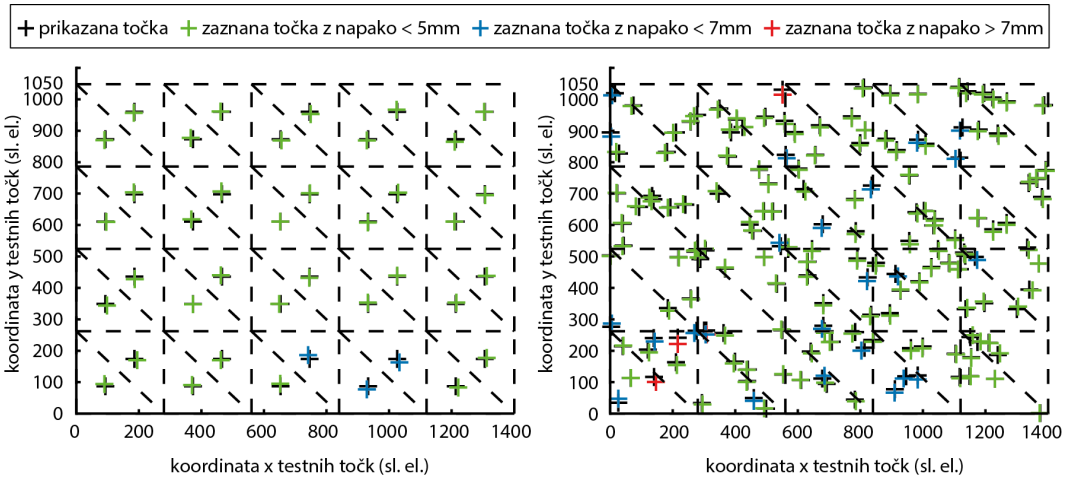
### 5.3.1 Natančnost

Slika 5.5 prikazuje lokacije točk, uporabljenih pri evaluaciji. Levo so prikazane točke, izbrane v prvi seriji točk, medtem ko desna slika prikazuje lokacije točk v drugi seriji. Lokacije prikazanih točk so označene s črnimi križci. Z zelenimi so označene lokacije zaznanih točk, katerih napaka je bila manjša od $0,5cm$, z modro točke z napako, manjšo od $0,7cm$ in z rdečo točke, katerih napaka je bila večja od $0,7$ cm. Črtkane črte označujejo meje trikotnikov, uporabljenih pri preslikavi med koordinatnima sistemoma, njihova oglišča pa so kalibracijske točke.

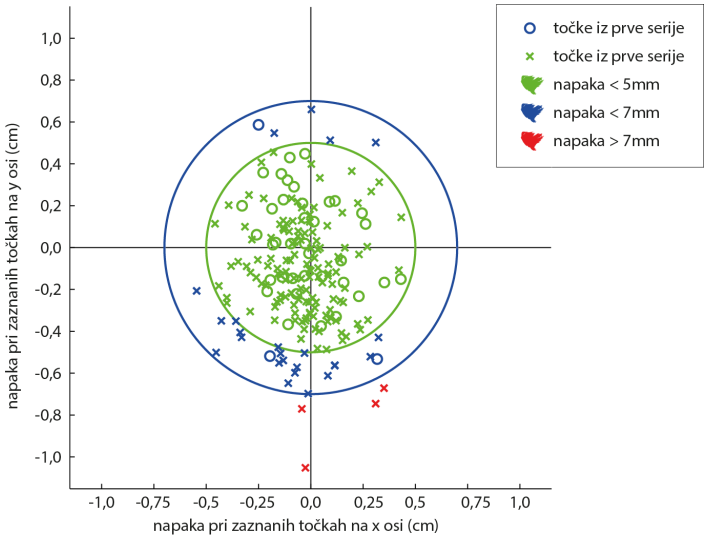
Slika 5.6 prikazuje odmike posameznih zaznanih točk od prikazane točke. Zelen in moder krog prikazujeta meji oddaljenosti $0,5cm$ in $0,7cm$. Na sliki 5.8 so enaki rezultati za boljšo predstavo prikazani tudi na človeškem prstu.

Rezultati testiranja **200 točk** so prikazani v tabeli 5.9. Iz rezultatov je razvidno, da je razvit sistem izjemno zanesljiv, saj je napaka pri zaznavi v $\mathbf{84,5}\%$ točk manjša kot $0,5cm$, v kar $\mathbf{98}\%$ pa manjša kot $0,7cm$. Slika 5.7 (levo) prikazuje število napak po posameznih območjih ter kumulativno na sliki 5.7 (desno). Kot je razvidno s slike 5.8, tudi napaka do $0,7cm$ še vedno omogoča povsem zadovoljivo natančnost za interakcijo s sistemom. Čeprav se na prvi pogled lahko zdi napaka izražena v slikovnih elementih velika, je

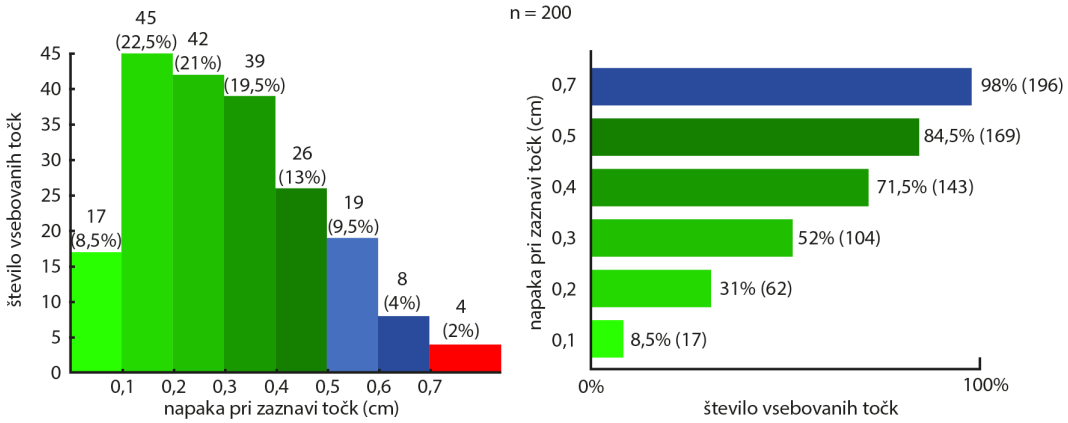
potrebno pri tem upoštevati tudi velikost posameznega slikovnega elementa. Rezultati testiranj so potrdili, da je sistem dovolj natančen za interakcijo s ciljnimi aplikacijami celo pri ločljivosti 1400 × 1050 sl. el. Tako interakcijo smo preizkusili tudi v praksi, ter je opisana v poglavju 6.



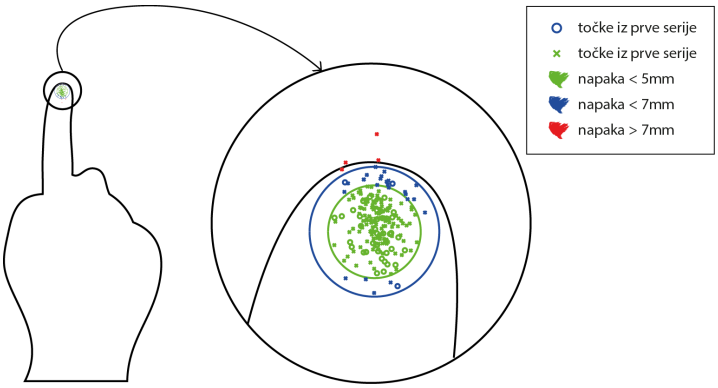

Slika 5.5: Pozicije testnih točk prve (levo) in druge serije (desno)




Slika 5.6: Napake testnih točk




Slika 5.7: Histogram napake zaznanih točke (levo) in kumulativno (desno)




Slika 5.8: Napaka v zaznavi posameznih točk, prikazana na uporabnikovem prstu

| | |
|---|---|
| št. točk | **200** |
| št. točk z napako $< 5mm$ | $\mathbf{169}(\mathbf{84,5}\%)$ |
| št. točk z napako $< 7mm$ | $\mathbf{196}(\mathbf{98}\%)$ |
| povprečna napaka | $\mathbf{3,1mm}$ |
| povprečna napaka na x osi | $\mathbf{1,5mm}$ |
| povprečna napaka na y osi | $\mathbf{2,4mm}$ |
| povprečna napaka (sl. elementi) | $\mathbf{6,4px}$ |
| velikost sl. elementa | $\mathbf{0,46mm \times 0,49mm}$ |

Slika 5.9: Rezultati meritev natančnosti

### 5.3.2 Odzivnost

Rezultati so prikazani v tabeli 5.2. Sistem je za obdelavo posamezne globinske slike od prejetja do pošiljanja v povprečju porabil **6ms** pri enem prstu ter povprečno **10ms** pri uporabi desetih prstov. Kot glavna pomanjkljivost se je izkazala kamera Kinect, ki za zajem in posredovanje slike potrebuje v povprečju od 100 do 120$ms$. Tako je celoten sistem skupaj z zajemom, obdelavo in prikazom potreboval v povprečju **120ms** pri uporabi enega prsta, ter povprečno **125ms** pri uporabi desetih prstov. Čeprav lahko zamik sprva deluje velik, je za normalno uporabo povsem nemoteč. Zamik pride do majhnega izraza le pri uporabi aplikacij, ki za delo potrebujejo resnično takojšen odziv. To so v večini igrice, kjer je odzivni čas igralca pomemben. Kot omenjeno, obdelava slike porabi le **8**% celotnega časa, tako da bi sistem z boljšo globinsko kamero lahko deloval še veliko hitreje.

| | |
|---|---|
| povp. čas zajema | od 100$ms$ do 120$ms$ |
| povp. čas obdelave | 6$ms$ (en prst), 10$ms$ (deset prstov) |
| povp. čas prikaza | 1$ms$ |
| skupni povp. čas | od 120$ms$ do 135$ms$ |

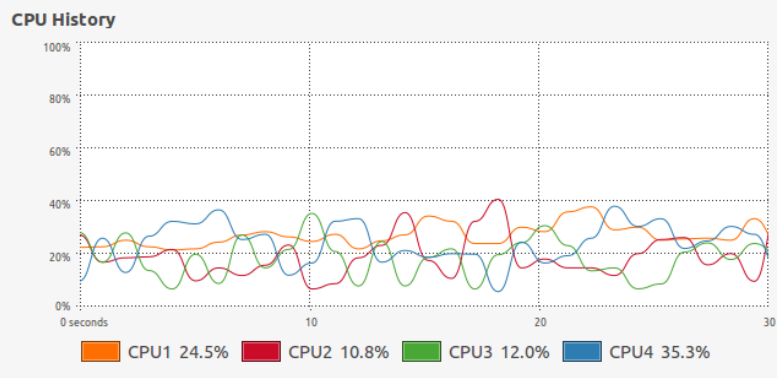
Tabela 5.2: Odzivnost sistema

### 5.3.3 Procesorska zahtevnost

Za boljšo predstavo o zahtevnosti zaznave, smo med procesom zaznave merili tudi zasedenost procesorja. Na sliki 5.10 je prikazana statistika zasedenosti procesorja na vsakem izmed štirih jeder za obdobje 30 sekund, medtem ko smo sledili desetim prstom na dveh uporabnikovih rokah. Povprečna zasedenost vsakega izmed jeder je bila **20**%. Glede na rezultat lahko zaključimo, da bi se lahko na istem računalniku izvajala tudi ciljna aplikacija. V primeru uporabe premalo zmogljivega računalnika, ki ne bi bil sposoben istočasnega izvajanja obeh aplikacij, bi ciljno aplikacijo lahko izvajali tudi na drugem računalniku, vendar le pod pogojem da sta računalnika med seboj povezana.

## 5.4 Omejitve

Razvit sistem ima določene omejitve, ki so posledica omejitev uporabljene strojne opreme ter algoritmov, uporabljenih pri zaznavi prstov. Omejitve sistema




Slika 5.10: Zasedenost procesorja

so:

- kamera Kinect mora biti ves čas delovanja statična (programska omejitev);
- največja dovoljena oddaljenost kamere Kinect od opazovane površine je **95cm** (strojna in programska omejitev);
- največja dimenzija opazovane površine je **80cm** $\times$ **69cm** (strojna omejitev);
- najmanjša možna zakasnitev sistema je približno $90ms$ (strojna omejitev);
- opazovana površina mora biti ravna (programska omejitev);
- nedelovanje na površinah iz odbojnih ali prosojnih materialov (strojna omejitev);
- nedelovanje na površinah, obsijanih s sončno svetlobo (strojna omejitev).

### 5.4.1 Natančnost

Pri uporabi kamere Kinect se je potrebno zavedati, da je to nizko cenovna globinska kamera. Da je kamera lahko na voljo po tako nizki ceni, je proizvajalec moral narediti nekaj kompromisov pri kakovosti, ki se odražajo na njeni natančnosti in funkcionalnosti. Natančnost kamere Kinect se z oddaljenostjo

opazovanih objektov zmanjšuje (glej podpoglavje 2.2.2) ter ne omogoča merjenja oddaljenosti na razdalji, manjši kot $40cm$. S praktičnim preizkusom smo ugotovili, da je zadovoljiva natančnost kamere Kinect, za naš algoritem zaznave, mogoča pri oddaljenosti največ $95cm$ od opazovane površine. Največjo dovoljeno oddaljenost bi lahko povečali z drugačnim načinom odstranjevanja ozadja, vendar bi za to potrebovali računalnik s podporo GPE, čemur pa smo se želeli izogniti. Pri oddaljenosti $95cm$ je največja mogoča dimenzija opazovane površine $80cm \times 69cm$.

### 5.4.2 Odzivnost

Kamera Kinect za zajem ter posredovanje globinske slike računalniku, po neuradnih podatkih [26], potrebuje okoli $90ms$. Na ta čas nimamo nobenega vpliva, tako da je to potrebno upoštevati že pri zasnovi sistema. Možna rešitev je uporaba druge globinske kamere, ki bi omogočala krajši čas zajema in posredovanja.

### 5.4.3 Sončna svetloba

Kamera Kinect, po podatkih s spletne strani skupnosti OpenKinect [34], uporablja za projiciranje IR svetlobe lasersko diodo. Ta oddaja svetlobo z valovno dolžino $830nm$. Na območjih, ki so neposredno obsijana s sončno svetlobo, ima lahko kamera Kinect težave z zaznavo oddaljenosti posameznih točk, saj sončna svetloba vsebuje tudi širok spekter IR svetlobe. Projiciran vzorec, zajet z IR kamero, tako na takih območjih poleg vzorca projiciranega z omenjeno diodo zajame tudi IR spekter sončne svetlobe.

Ker podatke o globini pridobivamo na podlagi slike, zajete z IR kamero, osvetlitev z drugih virov svetlobe (z izjemo IR) ne vpliva na zmožnost in kakovost zajema. Tako lahko sistem deluje v svetlem ali temnem prostoru.

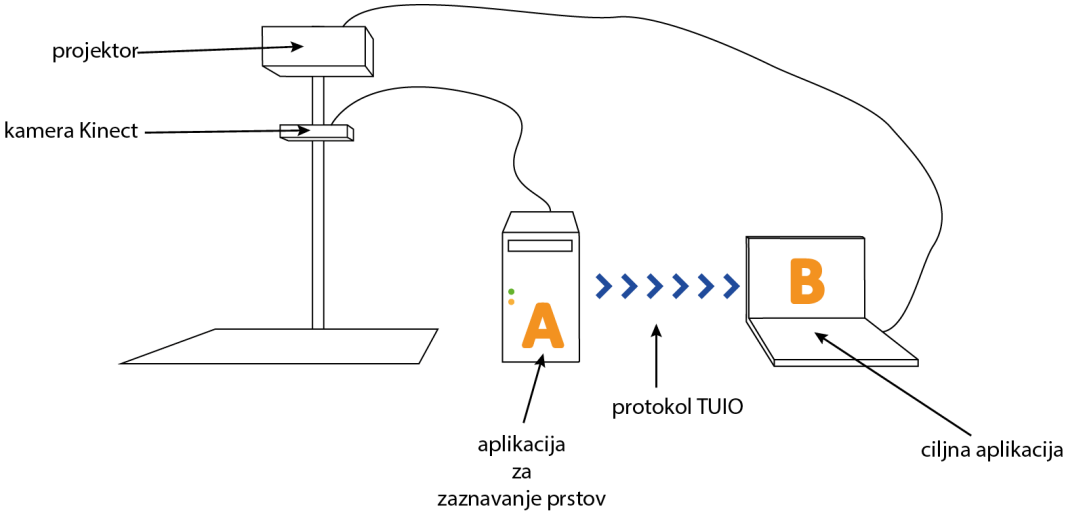
# Poglavje 6

# Primeri uporabe

Z evaluacijo sistema v poglavju 5 smo izmerili natančnost zaznave in odzivnost sistema, medtem ko uporabniški izkušnji nismo namenjali preveč pozornosti. Čeprav rezultati ovrednotenja kažejo, da je sistem zelo natančen (povprečna napaka je $\mathbf{3,1mm}$), to še ni zagotovilo, da bo sistem omogočal uporabniku dobro uporabniško izkušnjo. Morebitne težave bi se lahko pojavile zaradi neuspešnega zaznavanja dotikov površine, napak pri lokaciji zaznave in s tem povezanim nezaželenim delovanjem aplikacij ter z neuspešnim sledenjem prstov skozi čas. To bi onemogočilo zaznavo uporabnikovih kretenj v posameznih aplikacijah.

Ker ciljnih aplikacij, ki omogočajo primerno izrabo 3D informacij, še ni, smo preizkus uporabniške izkušnje razdelili na dva dela. V prvem smo sistem povezali s prenosnim računalnikom z nameščenim operacijskim sistemom Windows 7. Računalnik smo upravljali preko dotikov na opazovani površini, kar je pravzaprav le za zaznavanje prstov, katerih konice se nahajajo pod določenim pragom nad opazovano površino. Delovanje je tako enako kot ga omogoča Microsoft Surface in odprtokodna rešitev CCV (Community Core Vision) [22]. V drugem delu pa smo preverili zaznavo v 3D prostoru s preprosto aplikacijo, ki smo jo razvili za potrebe preizkusa.

Na sliki 6.1 je prikazana postavitev razvitega sistema med delovanjem. Poleg računalnika (računalnik A) in projektorja, katerih značilnosti so opisane v tabeli 5.1, smo v prvem preizkusu uporabili tudi prenosni računalnik (računalnik B) z operacijskim sistemom Windows 7.



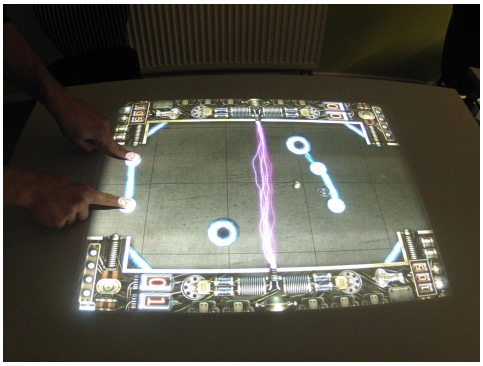

Slika 6.1: Shematski prikaz komponent sistema

## 6.1 Dvo-dimenzionalna zaznava

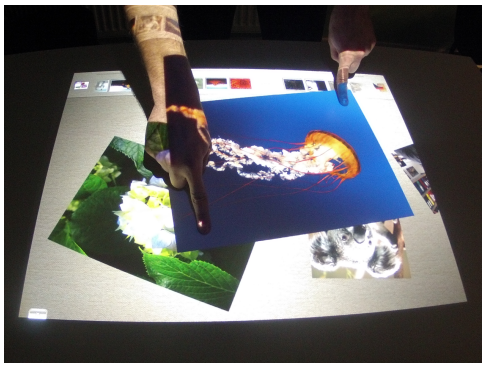
Pred preizkusom smo morali na računalniku B namestiti ustrezen gonilnik (UniSoftHID) in program Multi-touch Vista [30], ki operacijskemu sistemu Windows 7 doda večdotično funkcionalnost. Podatke, prejete preko sporočil TUIO, pretvori v dogodke, na katere se operacijski sistem in posamezne aplikacije odzovejo. Za omenjeno konfiguracijo smo se odločili, ker nam omogoča upravljanje z identičnimi aplikacijami, kot jih drugače upravljamo z miško. To nam je omogočilo verodostojno primerjavo uporabniških izkušenj. Poleg tega upravljanje računalnika v veliko primerih zahteva natančno zaznavo lokacije pritiska, kar bo še dodaten pokazatelj tega, kako natančen je razvit sistem.

Po uspešni inicializaciji in kalibraciji sistema, kjer smo tako kot pri evaluaciji uporabili kalibracijsko mrežo $6 \times 5$ točk, se je odjemalec TUIO na prenosnem računalniku uspešno povezal s strežnikom TUIO v razvitem sistemu. Miško smo brez težav lahko upravljali s prstom. Na samem začetku se je bilo samo potrebno navaditi na dejstvo, da kurzorja miške ni potrebno premikati na želeno mesto, kot je to potrebno z navadno miško, ampak je dovolj dotik na mestu, kjer želimo interakcijo. Uporabniško izkušnjo smo preverili tako z aplikacijami (Microsoft Touch Pack for Windows 7), ki so bile razvite posebej za uporabo na interaktivnih mizah (slike od 6.2 do 6.4), kot z aplikacijami, ki so v osnovi namenjene interakciji z miško (igra Angry Birds (slika 6.5) in program Slikar (slika 6.6)).

Slika 6.2: Microsoft Surface Globe

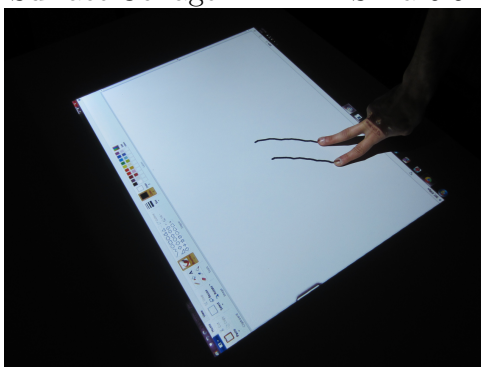


Slika 6.3: Microsoft Rebound



Slika 6.4: Microsoft Surface Collage

Slika 6.5: Angry Birds



Slika 6.6: Slikar

Čeprav so bile določene aplikacije razvite posebej za interaktivne mize, velike razlike v uporabniški izkušnji ni. Edina do neke mere logična razlika, je podpora večdotičnosti, ki je z izjemo Slikarja namizne aplikacije ne podpi-

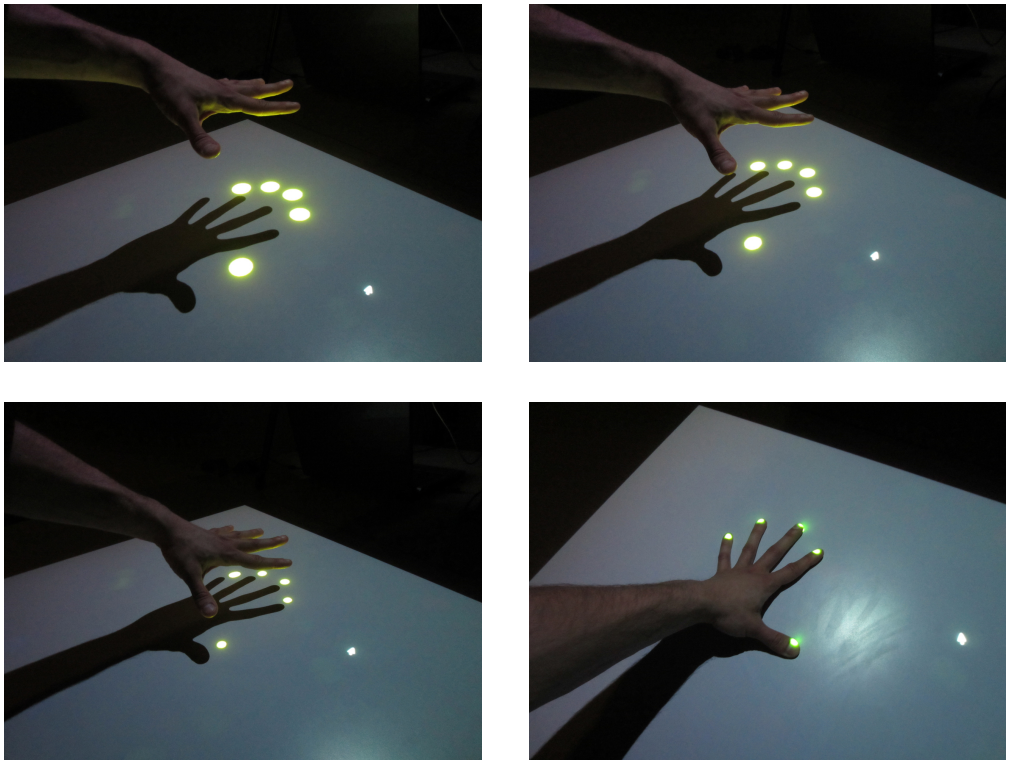
rajo. Glede na rezultate evaluacije sistema bi lahko pričakovali največ težav z odzivnostjo sistema. Majhen zaostanek pride do izraza samo pri aplikacijah, ki potrebujejo resnično takojšen odziv. V našem primeru je bila to igrica Microsoft Rebound (slika 6.3), kjer je manjši zamik povzročil nekaj težav pri igranju. Pri preostali uporabi pa zamik ne pride do izraza in tako ni moteč.

Kot smo že ugotovili z rezultati evaluacije, je sistem natančen, kar se je izkazalo tudi pri upravljanju z računalnikom. Manjša pomanjkljivost je le problem določitve praga dotika. Če je prag postavljen prenizko, sistem določenih dotikov ne prepozna, medtem ko previsok prag povzroči zaznavo dotika, še preden se je uporabnik površine res dotaknil. Med preizkusom smo mejo dotika postavili na $1,5cm$ nad površino, kar je omogočilo dovolj natančno zaznavo in hkrati malo prehitrih zaznav. Občasno pa se nam je vseeno pripetilo, da smo se s prsti premikali preblizu površine, tako da je sistem to zaznal kot dotik. Tako kot v primeru premikanja kurzorja miške je potrebno le nekaj malega časa za prilagoditev na sistem, nato se podobne stvari ne dogajajo več.

## 6.2 Tri-dimenzionalna zaznava

Za preizkus delovanja zaznave v 3D prostoru, smo razvili preprosto aplikacijo, ki prikazuje lokacijo zaznanih uporabnikovih prstov. Na mestu dotika se izriše krog, katerega barva je odvisna od bližine prsta opazovani površini. Če se prst dotika površine, je obarvan zeleno, v nasprotnem primeru pa rumeno. Velikost kroga ponazarja oddaljenost posameznega prsta od površine. Slika 6.7 prikazuje približevanje uporabnikove roke.

Čeprav je bila uporabljena aplikacija izjemno preprosta, smo z njeno uporabo lahko preverili robustnost in natančnost zaznave v 3D prostoru. Sistem se je, tako kot v predhodnih preizkusih, izkazal za zelo zanesljivega, saj nismo imeli nobenih problemov pri zaznavi prstov, prav tako pa so bile zaznave zelo natančne. Sledenje prstov je bilo, tako kot v 2D, povsem brez napak. Preizkus je pokazal, da bi lahko zaznavo kretenj v 3D prostoru uporabili v številnih aplikacijah.



Slika 6.7: Približevanje uporabnikove roke

# Poglavje 7

# Sklepne ugotovitve

V okviru diplomskega dela smo razvili sistem za dodajanje večdotične 3D funkcionalnosti opazovani površini s pomočjo sledenja uporabnikovih prstov. Za zajem podatkov smo uporabili barvno-globinsko kamero Kinect. Razvit sistem smo ocenili na podlagi natančnosti, odzivnega časa in uporabniške izkušnje.

Pri implementaciji sistema smo pozornost namenili tako robustnosti in natančnosti kot optimizaciji količine obdelanih podatkov. V primerjavi s predhodno opisanimi modeli ozadja smo globinski sliki dodali še matematično enačbo ravnine in meje opazovane površine. Z uporabo matematične enačbe smo izboljšali natančnost in robustnost segmentacije ospredja ter natančnost globinske zaznave konic uporabnikovih prstov. Uporaba mej opazovane površine nam omogoča enostavno odpravo nepotrebne obdelave podatkov, ki so izven nam zanimive regije. To je še posebej pomembno, saj smo želeli razviti sistem, ki bi bil sposoben delovati na računalnikih brez podpore GPE. V ta namen smo v več primerih tudi prilagodili posamezne algoritme našim potrebam. Tako smo odstranjevanje ozadja naredili na podlagi 2D globinskih slik ter v 3D pretvorili le elemente, ki smo jih označili kot ospredje. Prav tako smo pri segmentaciji uporabili 8-strane sosede iz prvotne globinske slike, namesto da bi uporabili evklidsko razdaljo kot način določitve sosedov. Za zaznavo konic uporabnikovih rok smo uporabili algoritem *k-curvature*. Ker pri zajemu podatkov s kamero Kinect prihaja do šuma v meritvah, smo obris segmentiranih rok preslikali nazaj v 2D sliko, ki smo jo nato uporabili za analizo. Kot je nam znano, takega postopka segmentacije v 3D in preslikave nazaj v 2D za analizo obrisa ni uporabil še nihče.

Sistem se je s povprečno napako v zaznavi **3**, **1mm** izkazal za zelo zanesljivega in natančnega. Odzivni čas je bil v povprečju od **120ms** do **140ms**, kar

pa je, kot smo ugotovili, v veliki večini posledica kamere in ne našega algoritma. Prav tako je bila uporabniška izkušnja celo nad začetnimi pričakovanji, saj je omogočala upravljanje z računalnikom brez težav pri velikosti slike $65cm \times 49cm$ in ločljivosti $1400 \times 1050$. V primerjavi z zelo znano interaktivno mizo Samsung SUR40 (z Microsoft PixelSense), razvit sistem omogoča zaznavo tudi v 3D prostoru, hkrati pa je neprimerno cenejši (1000€ proti 6500€).

Kljub zelo dobrim rezultatom je še veliko prostora za izboljšavo ter nadgradnjo sistema. Največji problem ostaja odzivni čas, ki bi ga lahko zelo skrajšali z uporabo katere izmed drugih globinskih kamer. Pri tem bi bilo potrebno paziti, da bi natančnost sistema ostala primerljiva. Z uporabo več kamer Kinect hkrati bi lahko zelo povečali območje opazovane površine.

Z izdelavo primernih ciljnih aplikacij, ki bi izrabljale 3D informacije, bi lahko sistem uporabili v številne namene. Pri prikazovanju informacij na površini bi lahko uporabnik spreminjal količino prikazanih podrobnosti s približevanjem ali oddaljevanjem prstov od površine. S podobno kretnjo bi lahko povečeval ali pomanjševal zemljevid ali sliko. Informacijo o oddaljenosti bi lahko uporabili tudi pri številnih igricah, kjer bi na primer pomembno vlogo pri igranju igrala tudi hitrost približevanja in oddaljevanja. Prav tako bi lahko v grafičnih aplikacijah oddaljenost vplivala na debelino črte, spreja in ostalih orodij. Sistem bi lahko tudi veliko doprinesel k številnim programom za učenje, kjer bi lahko sledili uporabnikovim prstom, ter zaznali napake, ki jih uporabnik dela. Prav tako bi bil primeren za uporabo v sterilnih pogojih, saj za delovanje ne potrebuje posebnega kontrolerja, kar je v takih primerih vedno izjemno problematično.

Končni izdelek diplomskega dela je delujoč prototip sistema. Z njegovo izdelavo smo dosegli zastavljen cilj in predstavlja dobro osnovo za nadaljne nadgradnje.

# Dodatek A

# Kalmanov filter

Od leta 1960, ko je R. E. Kalman objavil članek [9], v katerem je opisal rekurzivno rešitev linearnega filtriranja časovno diskretnih signalov, je ta postal izjemno priljubljen in uporabljen na področjih letalske, vesoljske in radarske tehnike. Z nadaljnjim razvojem računalniškega vida pa je sedaj velikokrat uporabljen tudi pri vodenju robotov, upravljanju avtomobilov in sledenju ljudi.

Kalmanov filter je množica matematičnih enačb, ki omogočajo računsko efektivno (rekurzivno) optimalno oceno trenutnega stanja linearnega dinamičnega sistema na podlagi meritev, ki so motene z belim Gaussovim šumom, tako da minimizira srednje kvadratno odstopanje napak. Pri oceni upošteva znanje o modelu dinamike sistema in njegovih negotovostih, statističnih podatkih o merskih napakah in morebitne začetne vrednosti opazovanih spremenljivk.

Linearno dinamični sistem opišemo z enačbo za prehajanje stanj, ki opisuje soodvisnost med dvema časovno zaporednima stanjema

$$x_{k+1} = A_k x_k + w_k \tag{A.1}$$

in izhodno enačbo, ki povezuje stanja ter meritve:

$$z_k = H_k x_k + v_k, \tag{A.2}$$

kjer je k časovni korak, x vektor stanj, A matrika prehajanja stanj, z vektor meritev, H izhodna matrika, w šum procesa in v merilni šum.

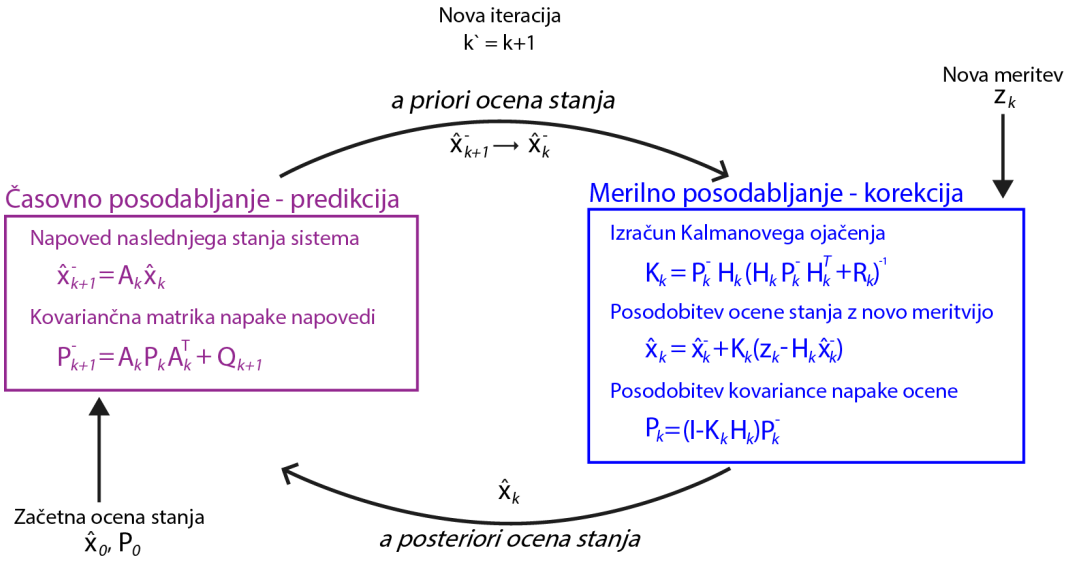
Da Kalmanov filter zagotavlja optimalno rešitev ocene stanja sistema, morajo biti izpolnjeni določeni pogoji. Poznati moramo matriko prehajanja stanj, izhodno matriko ter n-dimenzionalni vektor šuma procesa $w_k$ in m-dimenzionalni vektor šuma meritve $v_k$. Vektor šuma procesa in meritve morata biti medsebojno neodvisna, normalno porazdeljena ter imeti srednjo vrednost enako nič.

Med delovanjem Kalmanovega filtra uporabljamo dve oceni vektorja stanj sistema.

- **A posteriori ocena** ($\hat{\mathbf{x}}_\mathbf{k}$) predstavlja optimalno oceno vektorja stanja ob času $t_k$. Dobimo jo na podlagi predhodne *a priori ocene* stanja ($\hat{x}_k^-$) in meritve $z$ ob času $t_k$;
- **A priori ocena** ($\hat{\mathbf{x}}_{\mathbf{k+1}}^-$) predstavlja projekcijo *a posteriori ocene* ($\hat{x}_k$) v naslednji časovni korak $t_{k+1}$ s pomočjo matrike prehoda stanj.

$$\hat{x}_{k+1}^- = A_k \hat{x}_k, \tag{A.3}$$

Kalmanov filter deluje na principu prediktorsko-korekcijskega cikla (slika A.1) z uporabo povratne informacije (meritve). Filter v prediktorskem koraku napove prihodnji vektor stanj, ki ga nato ob prejetju povratne informacije (meritve) v korekcijskem koraku popravi.




Slika A.1: Kalmanov cikel

V prediktorskem koraku s pomočjo enačbe (A.3) in matrike prehajanja stanj ($A_k$) projicira a posteriori oceno vektorja stanj $\hat{x}_k$ v naslednji časovni korak $t_{k+1}$. S tem dobimo novo a priori oceno stanja $\hat{x}_{k+1}^-$ ter posodobljeno kovariančno matriko napake napovedi $P_{k+1}^-$. Ob prejetju meritve $z_k$ oceno vektorja stanj izboljšamo s pomočjo Kalmanove ojačevalne matrike $K_k$. Kalmanova

ojačevalna matrika $K_k$ je neodvisna od meritve $z_k$ in je odvisna le od kovariančnih matrik šuma procesa $Q_k$ in meritve $R_k$ ter kovariančne matrike napak ocene $P_k$. Vedno je izbrana tako, da je varianca vektorja napake ocene minimalna. Po izboljšavi dobimo optimalno a posteriori oceno vektorja stanj $\hat{x}_k$ ter posodobljeno kovariančno matriko napake ocene $P_k$.

Ob inicializaciji filtra je potrebno podati začetno stanje $\hat{x}_0$ ter začetno kovariančno matriko te ocene $P_0$. V primeru nepoznavanja začetnega stanja lahko podamo kovariančno matriko z dovolj velikimi vrednostimi in bo tako podano začetno stanje povsem nepomembno. Stanje bo v vsakem primeru zelo hitro konvergiralo k pravi vrednosti.

# Literatura


[1] H. Benko, R. Jota, A. Wilson, "MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop", v zborniku *SIGCHI Conference on Human Factors in Computing Systems*, str. 1999-208, 2012.

[2] D. Fiedler, H. Müller, "Impact of Thermal and Environmental Conditions on the Kinect Sensor", v zborniku *International Workshop on Depth Image Analysis*, 2012.

[3] C. Harrison, H. Benko, A. Wilson, "OmniTouch: wearable multitouch interaction everywhere", v zborniku *UIST'2011*, str. 441-450, 2011.

[4] D. Herrera C., J. Kannala, J. Heikkilä, "Joint depth and color camera calibration with distortion correction", v *IEEE Transactions on Pattern Analysis and Machine Intelligence*, št. 34 zv. 10, str. 2058-2064, 2012.

[5] O. Hilliges, S. Izadi, A. Wilson, S. Hodges, A. Garcia-Mendoza, A. Butz, "Interactions in the Air: Adding Further Depth to Interactive Tabletops", v zborniku *UIST'09*, str. 139-148, 2009.

[6] L. Hyung-Min, "Kinect Multitouch Table:A Platform for Creative App Development", 2012.

[7] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Bidermann, E. Ben-Joseph, "Bringing clay and sand into digital design - continuous tangible user interfaces", *BT Technology*, št. 22, zv. 4, str. 287-299, 2004.

[8] B. R. Jones, R. Sodhi, R. H. Campbell, G. Garnett, B. P. Bailey, "Build Your World and Play In It: Interacting with Surface Particles on Complex Objects", v zborniku *ISMAR 2010*, str. 165-174, 2010.

[9] R. E. Kalman, "A new approach to linear filtering and prediction problems", *Transactions of the ASME, Journal of Basic Engineering*, št. 82, str. 35-45, 1960.

[10] M. Kaltenbrunner, T. Bovermann, R. Bencina, E. Costanza, "TUIO - A Protocol for Table Based Tangible User Interfaces", v zborniku *Proceedings of the 6th international workshop on gesture in human-computer interaction and simulation (GW 2005)*, Vannes, Francija, 2005.

[11] K. Khoshelham, "Accuracy analysis of kinect depth data", v zborniku *ISPRS workshop laser scanning*, Calgary, Kanada, 2011.

[12] F. Klompmaker, K. Nebe, A. Fast, "dSensingNI: a framework for advanced tangible interaction using a depth camera", v zborniku *TEI'12*, str. 217-224, 2012.

[13] D. Koller,J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russell, "Towards Robust Automatic Traffic Scene Analysis in Real Time", v zborniku *IEEE International Conference on Pattern Recognition (ICPR)*, št. 1, str. 126-131, 1994.

[14] J. Letessier, F. Bérard, "Visual tracking of bare fingers for interactive surfaces", v zgorniku *UIST'04*, str. 119-122, 2004.

[15] S. Murugappan, et al., "Extended multitouch: recovering touch posture and differentiating users using a depth camera", v zborniku *UIST'12*, str. 487-496, 2012.

[16] M. Piccardi, "Background subtraction techniques: a review",v zborniku *IEEE International Conference on Systems, Man and Cybernetics*, št. 4, 2004.

[17] Z. Ren, J. Yuan, Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera", v zgorniku *19th ACM international conference on Multimedia*, str. 1093-1096, 2011.

[18] C. Schwarz, N. Lobo, "Segment-Based Hand Pose Estimation", v zborniku *2nd Canadian conference on Computer and Robot Vision*, str. 42-49, 2005.

[19] C.H. Teh, R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, št. 11, str. 859-872, 1989.

[20] A. Wilson, "Using a Depth Camera as a Touch Sensor", v zborniku *ITS'2010*, str. 69-72, 2010.

[21] A. Wilson, H. Benko, "Combing multiple depth cameras and projectors for interactions, on, above and between surfaces", v zborniku *UIST'10*, 2010.

[22] (2013) Community Core Vision. Dostopno na: http://ccv.nuigroup.com/

[23] (2013) Community Earth. Dostopno na: http://nuicode.com/projects/earth

[24] (2013) Dell Studio 17 Touch adds multitouch to monster notebook. Dostopno na: http://www.slashgear.com/dell-studio-17-touch-adds-multitouch-to-monster-notebook-1963918/

[25] (2013) Dell 3400MP DLP. Dostopno na: http://www.projectorcentral.com/Dell-3400MP.htm

[26] (2013) Durango Next-Generation Kinect Sensor. Dostopno na: http://www.vgleaks.com/durango-next-generation-kinect-sensor/

[27] (2013) Glassomium. Dostopno na: http://www.glassomium.org/

[28] (2013) I Have Seen The Future, And The Future Is Xbox Kinect. Dostopno na: http://blogs.forrester.com/james_mcquivey/10-06-14-i_have_seen_future_and_future_xbox_kinect

[29] (2013) Microsoft PixelSense. Dostopno na: www.microsoft.com/en-us/pixelsense/

[30] (2013) Multi-touch Vista. Dostopno na: http://multitouchvista.codeplex.com/

[31] (2013) NASA World Wind. Dostopno na: http://worldwind.arc.nasa.gov/java/

[32] (2013) Nintendo Wii Controller. Dostopno na: http://list.qoo10.sg/item/NINTENDO-WII-CONTROLLER/407342260

[33] (2012) NUI Group - Natural User Interface Group. Dostopno na: www.nuigroup.com

[34] (2013) OpenKinect - Hardware info. Dostopno na: http://openkinect.org/wiki/Hardware_info

[35] (2012) ROS - Technical description of Kinect calibration. Dostopno na: http://www.ros.org/wiki/kinect_calibration/technical/

[36] (2013) The Microsoft Accelerator for Kinect. Dostopno na: http://www.microsoft.com/bizspark/kinectaccelerator/

[37] (2013) The PEREGRINE Wearable Interface - Medium Glove. Dostopno na: http://www.amazon.com/The-PEREGRINE-Wearable-Interface-Medium/dp/B0035HABMM

[38] (2012) TUIO. Dostopno na: http://www.tuio.org/

[39] (2013) Ubi Interactive. Dostopno na: http://www.ubi-interactive.com/

[40] (2013) Windows Touch. Dostopno na: http://windows.microsoft.com/en-us/windows7/products/features/touch