Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Meden

Detekcija dlani in položaja prstov s pomočjo barvno-globinske kamere

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

doc. dr. Iztok Lebar Bajec
MENTOR

Ljubljana, 2013

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: 00501/2013
Datum: 12.04.2013

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

Kandidat: **BLAŽ MEDEN**

Naslov: **DETEKCIJA DLANI IN POLOŽAJA PRSTOV S POMOČJO BARVNO-GLOBINSKE KAMERE**

PALM AND FINGER DETECTION USING A COLOUR-DEPTH CAMERA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V zadnjem obdobju je interakcija z napravami in stroji vse manj »posebna«. Komunikacijo človek-stroj, ki je večinoma potekala preko tipkovnice, počasi nadomeščajo dotiki na dotik občutljivih ekranov. Vedno bolj pogosto pa se srečujemo tudi z napravami, ki odpravljajo še slednje. V teh zadnjih primerih govorimo o brez dotični uporabi. Težava takih sistemov je ravno v odsotnosti dotika; odsotnosti povratne informacije, ki jo je uporabnik imel, ko je še fizično pritiskal tipke tipkovnice ali na dotik občutljivi ekran. Zato je še toliko bolj pomembno kako odzivni so taki sistemi in na kakšen način lahko uporabnik z njimi dejansko komunicira.

V diplomski nalogi preučite princip delovanja brez dotičnih naprav in preglejte načine njihovega upravljanja. Nato z uporabo barvno-globinske kamere in odprtokodnih knjižnic zasnujte aplikacijo, ki bo sposobna zaznavati dlan in položaj prstov uporabnikove roke. Pridobljene izkušnje ustrezno komentirajte.

Mentor:

doc. dr. Iztok Lebar Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

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

— Blaž Meden, Ljubljana, september 2013.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Meden

Detekcija dlani in položaja prstov s pomočjo barvno-globinske kamere

POVZETEK

Naravna interakcija med človekom in računalnikom je v današnjem času zelo popularno raziskovalno področje. Še posebej je veliko truda vloženega v natančno detekcijo in sledenje dlanem ter prstom. Precizni podatki, pridobljeni iz strukture človeške roke v realnem času, bi lahko izboljšali interaktivnost v različnih panogah. Tako bi, na primer, pridobljene podatke uporabili za oddaljeno krmiljenje robotske roke, v pomoč bi lahko bili kirurgom v medicini, z njimi bi izboljšali uporabniške vmesnike interaktivnih aplikacij ali izboljšali igralne in druge sisteme namenjene zabavi. V tej diplomi zato najprej predstavimo stanje in pregled področja detekcije in sledenja rok ter opišemo uporabljene tehnologije. Zatem je predstavljena preprosta implementacija detektorja dlani z uporabo ogrodja OpenFrameworks, knjižnice OpenNI in senzorja Kinect. Končno implementacijo sistema tudi ovrednotimo in prikažemo njeno uporabo v dveh praktičnih aplikacijah.

Kljucne besede: detekcija dlani, sledenje rok, sledenje prstov, prepoznavanje gest, ogrodje OpenFrameworks, OpenNI, NiTE, OpenCV, Kinect

University of Ljubljana
Faculty of Computer and Information Science

Blaž Meden

Palm and finger detection using a color-depth camera

ABSTRACT

Natural user interaction is nowadays a very popular research field. In particular, there is a lot of effort put in accurate tracking of human hands and fingertips. Accurate data taken in real time directly from the human hand structure could improve natural interaction in many fields. For example, we could use such data, to be able to remotely guide robotic hands, use it to help surgeons in the medical field, or make better use of interactive user interfaces and improve home entertainment systems. This diploma thesis starts with a brief overview of techniques, methods and technologies used in hand tracking research field. It continues with a simple implementation of a hand tracker using OpenFrameworks, OpenNI libraries and the Kinect sensor. It concludes by evaluating the implementation through two practical applications.

Key words: palm detection, hand tracking, finger tracking, gesture recognition, OpenFrameworks, OpenNI, NiTE, OpenCV, Kinect

ZAHVALA

Na tem mestu se zahvaljujem mentorju doc. dr. Iztoku Lebarju Bajcu za vse nasvete pri pripravi diplomske naloge. Zahvalil bi se tudi ekipi iz podjetja Uniki d.o.o., saj sem v času izdelave diplomskega dela veliko časa preživel v njihovih prostorih. Za konec naj se zahvalim tudi staršem, za vso podporo, spodbudo in potrpežljivost v celotnem obdobju študija.

— Blaž Meden, Ljubljana, september 2013.

KAZALO

1 Uvod

Tehnologije v računalništvu se v zadnjem desetletju bliskovito razvijajo. Računalniški sistemi so iz leta v leto zmogljivejši, kapacitete pomnilnikov se skokovito povečujejo, komunikacijske povezave lahko v trenutku prenesejo skoraj nepredstavljive količine podatkov. Ob vsem tem hitrem razvoju, pa smo do nedavnega lahko opazili, da so vmesniki za interakcijo z računalniškimi sistemi ostajali v enakih oblikah in funkcionalnostih. Veljalo je, da pri opravilih z računalnikom še vedno največ uporabljamo "tradicionalne" naprave, kot je na primer kombinacija računalniške miške s tipkovnico.

Vendar se tudi na področju upravljanja z računalniškimi sistemi dogajajo izjemne spremembe. Tukaj lahko najprej omenimo naprave z zasloni na dotik, kjer so naravnejši vmesniki prišli najprej do izraza. Pri teh napravah nismo več omejeni na nekaj tipk in gumbov, pač pa nam je na voljo celoten zaslon, ki zaznava dotike, premike prstov ter druge aktivnosti uporabnika. Razvijalcem je tako na voljo bolj dinamičen spekter podatkov, ki jih aplikacijam posreduje uporabnik. S tem so lahko aplikacije enostavnejše ter prijaznejše za uporabo. Lep primer takih sistemov so v zadnjem času zelo popularni pametni mobilni telefoni in tablični računalniki.

Verjetno največji korak naprej v interakciji človeka z računalnikom pa je naredila zabavna industrija. Pojavila se je namreč ideja, da bi kot vmesnik za igranje iger uporabili kar telo in okončine uporabnika. Obstoječim napravam (igralne palice, ploščki) so se tako pridružili sistemi, ki s pomočjo globinskih senzorjev omogočajo detekcijo celotnega telesa. S tem se izboljša igralna izkušnja, saj za izvajanje dejavnosti igralec uporablja celo postavo in se s svojim gibanjem dodatno vključi v dogajanje v virtualnem svetu. Najbolj odmevna in popularna ter tudi cenovno ugodna naprava na tem področju je zagotovo Microsoftova barvno-globinska kamera Kinect. Kinect je bil sprva najprej namenjen kot igralna komponenta, kasneje pa je zaradi svoje uporabnosti zaživel tudi na raziskovalnem in akademskem področju. Podrobnejši opis in delovanje senzorja Kinect ter pripadajočih knjižnic je na voljo v nadaljevanju, tukaj omenimo le, da s pomočjo namenskih algoritmov omogoča zaznavanje globine in rekonstrukcijo prostora ter uporabnikov, ki so prisotni v njegovem vidnem polju.

Obdelava večdimenzionalnih podatkov, najrazličnejše vizualizacije informacij, različne kreativne aplikacije, navigacija po navideznih svetovih ter manipuliranje z navideznimi objekti in navidezna resničnost nasploh so področja, kjer klasične vhodne naprave ne morejo biti optimalno uporabljene. Sem lahko prištejemo tudi nadzor različnih robotskih komponent (denimo robotske roke) na področju medicine ali industrijskih aplikacij. Tako se je pojavila potreba po naravnejšem upravljanju s sistemi, ki ponujajo sodobne medijske vsebine ali krmiljenje specifičnih komponent [39].

Tukaj na primer lahko omenimo tudi nedavno predstavljen senzor LeapMotion[1], ki na svoj način rešuje problem interakcije na področju navidezne resničnosti. Naprava deluje na podoben princip kot Kinect, njene dimenzije pa so, v primerjavi s Kinectom, zelo optimizirane. LeapMotion zaznava globino s pomočjo stereo sistema kamer. Vanj je vgrajena funkcionalnost za prepoznavanje prstov in drugih koničastih objektov, saj je senzor primarno namenjen kot alternativni vmesnik za upravljanje računalniških sistemov brez dotika. Sama detekcija predmetov v videm polju senzorja je zelo precizna. Poleg tega se razvijalcu neposredno ni potrebno ukvarjati z globinsko sliko. Senzor namreč kot izhodne podatke posreduje le ključne značilnosti zajetih objektov v obliki vnaprej definiranih modelov (npr. model dlani, prstov ali drugih koničastih predmetov ter gest). Iz teh modelov je možna pridobitev lokacije, hitrosti, normale ter ostalih geometrijskih podatkov opazovanega objekta [1].

[1] www.leapmotion.com

S prihodom Kinecta in podobnih senzorjev je reševanje naštetih problemov interakcije postalo veliko lažje. Z idejo, da bi računalniški sistem upravljali z rokami (na naravnejši način), se ukvarja tudi diplomska naloga, ki je pred vami. Kot bomo videli v nadaljevanju je namreč zaznavanje drže in položaja rok ter prstov zelo težak problem. Delno ta težavnost izvira iz dejstva, da je človeška roka kinematična struktura segmentov z veliko parametri (kar pomeni veliko dimenzionalnost problema). Hkrati je naravno gibanje rok nelinearno in sorazmerno zelo hitro. Poleg tega sama oblika in topologija dlani ne ponuja veliko kakovostnih značilk, ki bi bile konstantno na voljo med samim sledenjem in detekcijo parametrov. Pri detekciji z globinskimi senzorji pa lahko prihaja tudi do prekrivanja posameznih delov rok (prstov), kar še dodatno oteži pravilno zaznavanje. Velik problem obstoječih metod predstavlja tudi sama preciznost zaznavanja. Nanjo lahko vpliva tako šum v vhodnih podatkih, kot tudi njihova nepravilna interpretacija. Za reševanje nekaterih naštetih problemov je bilo predlaganih več različnih metod, ki jih bomo pregledali in ovrednotili v nadaljevanju. S pregledom metod bomo hkrati predstavili tudi obstoječe rešitve in v zadnjih poglavjih opisali implementacijo preprostega sistema za detekcijo nekaterih značilnosti dlani in položaja prstov.

2 Opis in ozadje problema

V programski opremi, ki jo podjetje Uniki d.o.o. med drugim uporablja tudi za razvoj brezdotičnih interaktivnih aplikacij, se je pokazala potreba po dopolnitvi podsistema, ki skrbi za komunikacijo med človekom in brezdotičnimi vmesniki. Nekaj aplikacij, s katerimi lahko komuniciramo brez dotika, deluje s pomočjo barvno-globinskega senzorja Kinect in vsebuje uporabniški vmesnik, ki je prirejen za uporabo brez dotika (slika 2.1). V trenutni različici pogona, podsistem za komunikacijo z uporabnikom uporablja le informacijo o lokaciji uporabnikove dlani. S stališča uporabniškega vmesnika to pomeni, da mora uporabnik za izbiro gumba roko zadržati nekaj sekund v ustrezni poziciji (nad gumbom). Povsem enak postopek velja za izvajanje drugih aktivnosti uporabniškega vmesnika. Taka interakcija je zelo bazična, lahko postane rutinska ter ima določene slabosti. Zaradi šuma v pridobljeni poziciji dlani iz senzorja lahko na primer prihaja do neželenega premikanja kazalca na zaslonu in s tem je otežena izbira grafičnih elementov v vmesniku.

Sistem bi lahko izboljšali tako, da bi v začetnem koraku iz globinske slike izluščili več informacij o položaju prstov in orientaciji ter položaju dlani uporabnika. S temi podatki,

Slika 2.1 Primer trenutnega vmesnika v brezdotični aplikaciji.

bi lahko v vmesnik dodajali nove aktivnosti, ki bi jih uporabnik nadzoroval z različnimi gestami, položajem svojih prstov ter nagibanjem rok. Pridobljene parametre bi lahko med seboj kombinirali, tako bi na primer lahko realizirali funkcionalnost prijemanja in premikanja predmetov po zaslonu (angl. grabbing and object manipulation). Navpično iztegnjeni palec bi lahko pomenil potrditev izbire, iztegnjenost kazalca pa bi (skupaj s premikanjem roke) na primer lahko bila uporabljena kot podlaga za kazanje ali risanje po površini interaktivnega zaslona. Interakcija z aplikacijo bi bila naravnejša, uporabnik bi imel na voljo več načinov za nadzorovanje aplikacije in izražanje svojih kretenj. V tem raziskovalnem delu se zato ukvarjamo s pridobivanjem značilnosti rok iz globinske slike, da bi se tako izboljšala uporabniška izkušnja v interaktivnih izdelkih podjetja Uniki d.o.o.

3 Pregled področja in obstoječih rešitev

V 70. letih prejšnjega stoletja je Myron Krueger, raziskovalec na univerzi v Conneticutu, ustvaril laboratorij Videoplace. Ideja laboratorija je bila ustvariti okolje, ki bi uporabnike obdajalo z navidezno realnostjo in bi se odzivalo na obnašanje teh ljudi v prostoru. Pri tem okolje ne bi uporabljalo naprav, ki bi jih udeleženci v prostoru morali imeti na sebi (senzorične rokavice ali očala) [2].

Okolje Videoplace je za svoje delovanje uporabljalo projektorje, video kamere ter drugo namensko strojno opremo. Kar je za naše raziskovalno področje zelo pomembno je to, da je interakcija okolja z ljudmi temeljila tudi na obdelavi vizualne informacije, ki je obsegala obrise teles uporabnikov. Uporabniki okolja so tako lahko komunicirali med seboj, čeprav so bili v ločenih prostorih. Premikanje udeležencev je bilo posneto in transformirano v obrise postav, ki so bili kasneje analizirani v okolju umetne realnosti. Uporabniki so odziv na svoje aktivnosti lahko videli na zaslonu, na katerem so bili prikazani analizirani obrisi. Čeprav med uporabniki ni bilo neposredne komunikacije, so lahko preko odziva na zaslonu občutili interakcijo z ostalimi udeleženci ter umetno generiranimi objekti iz navideznega okolja [2]. Na sliki 3.1 lahko vidimo različne načine

Slika 3.1 Načini interakcije z okoljem Videoplace [3].

interakcije z omenjenim okoljem. Opazimo lahko, da analiza obrisa išče predvsem dele obrisa z ostrejšimi krivuljami, ki najbolj izstopajo (torej večinoma konice prstov ali druge koničaste objekte). Na podlagi teh značilnosti se določijo ključne točke, preko katerih se izvaja interakcija (risanje, kazanje, sledenje objekta). Podoben koncept iskanja izstopajočih točk je uporabljen v naši implementaciji, ko na robovih roke iščemo konice iztegnjenih prstov. Danes je interaktivno okolje Videoplace sicer razstavljeno v zgodovinskem muzeju na univerzi v Conneticutu [37].

Kot vidimo je bilo že pred prihodom globinskih kamer področje sledenja uporabnikov in njihovih okončin za uporabo v interakciji med človekom in računalnikom zelo raznoliko. Članek iz leta 1994 [37] poleg okolja Videoplace omenja več metod, ki so bile tedaj uporabljene v raziskavah na tem področju. V njem najdemo različne pristope za določanje parametrov dlani, kot so na primer akustična metoda, metoda z uporabo magnetizma ter metoda, ki uporablja rokavico z množico senzorjev. Omenjeno je tudi optično sledenje, ki deluje na podlagi svetlobnih oznak, ki so fizično pritrjene na roko (množica LED diod ali izvorov infrardeče svetlobe).

Po navedbah omenjenega članka magnetna detekcija položaja dlani in prstov bazira na

generatorju, ki proizvaja magnetno polje. Za izračun pozicije in orientacije posameznih sklepov so uporabljeni senzorji v obliki tuljav, iz katerih se pridobi informacija o jakosti magnetnega polja. Za sledenje celotne strukture roke, so senzorji pritrjeni na posamezne prste ter na hrbtno stran dlani. Sledenje s pomočjo magnetizma se je izkazalo za zelo natančno, vendar ima svoje pomanjkljivosti. Slabost tovrstnega določanja parametrov je ta, da je zelo občutljivo na motnje magnetnih polj, ki lahko prihajajo iz drugih virov v okolju.

Akustično določanje parametrov dlani deluje na podoben princip kot magnetno sledenje, s to razliko, da za detekcijo uporablja zvočno valovanje. Namesto generatorja magnetnega polja je uporabljena naprava, ki producira kratke zvočne signale. Kot senzorji pa so uporabljeni mikrofoni, ki morajo biti v vidni liniji generatorja (angl. line of sight). S pomočjo triangulacije (ob uporabi vsaj treh mikrofonov), se lahko določijo parametri, ki definirajo položaj roke. Slabost je občutljivost na druge zvočne signale in odboje produciranih signalov. Delno se ta občutljivost rešuje z uporabo različnih frekvenc, lahko pa jo odpravljamo tudi s filtriranjem motenj v prejetih signalih iz mikrofonov. Problem predstavlja tudi določanje orientacije, saj jo na ta način ne moremo določiti.

Sledenje s pomočjo senzoričnih rokavic (angl. Data Gloves) se izvaja s pomočjo meritev, ki jih posredujejo senzorji, porazdeljeni v fizični rokavici, ki je pritrjena na roko. Zaradi fizičnega stika je možna meritev upognjenosti prstov ter zapestja. Pridobljene informacije lahko uporabimo za izračun 3D kinematičnega modela. Hkrati lahko uporabimo še senzorja za pospešek in nagib (pospeškomer in giroskop), da določimo globalno orientacijo in pozicijo v prostoru. Prednost pristopa je v tem, da imamo v vsakem trenutku na voljo podatek o stanju vsakega segmenta (popolna rekonstrukcija modela roke v realnem času). Slabost pa je seveda zahtevnost pri namestitvi sistema, njegova dolgotrajna konfiguracija ter okornost pri uporabi in visoka cena strojne opreme (rokavice, senzorji).

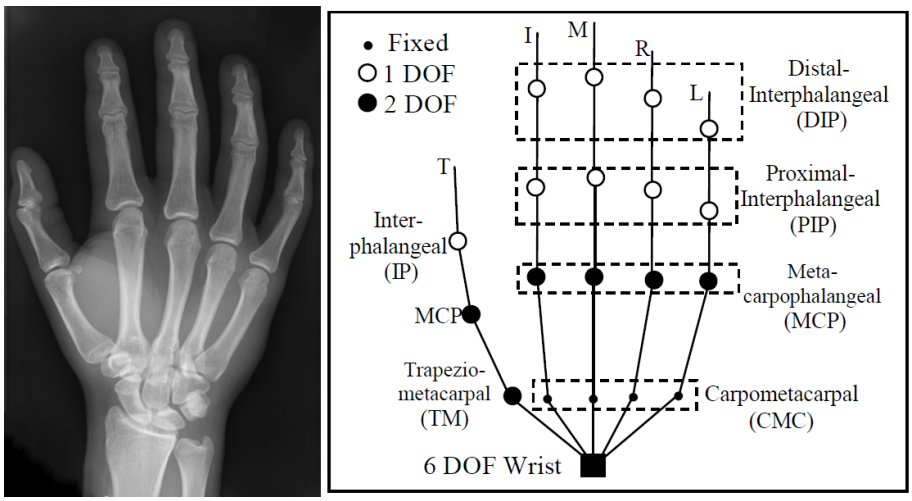
Alternativni načini detekcije bazirajo na vizualnih podatkih (analiza videa) ter v zadnjem času tudi na globinski informaciji (iz globinskih senzorjev). Sledenje na podlagi vizualne informacije in globine postaja za razliko od ostalih metod vse bolj razširjeno in za uporabnika predstavlja naravnejši način upravljanja kot predhodno naštete metode. Detekcija lahko uporablja različne vhodne naprave za sledenje. Uporabljena je lahko npr. samo ena kamera, lahko uporabimo sistem stereo kamer, s prihodom globinskih kamer pa lahko detekcijo kombiniramo tudi z globinskimi senzorji. Na podatkih, ki jih pridobimo iz

Slika 3.2 Primer senzoričnih rokavic [4].

omenjenih naprav se nato izvajajo različni algoritmi računalniškega vida ter druge tehnike segmentacije in tako poskušajo rekonstruirati parametre, katerih preciznost je odvisna od same implementacije sistema. Velika prednost vizualnega zaznavanja je ta, da uporabniku na sebi ni potrebno imeti dodatne opreme, saj se vse informacije pridobijo iz slikovnih (ena kamera) oz. tudi globinskih virov (stereo sistem ali globinska kamera). Slabosti so prisotne tudi pri tem pristopu, kot je bilo že omenjeno lahko pride do prekrivanja delov dlani ali prstov, problem pa predstavlja tudi hitro gibanje, ki onemogoča pravilno klasifikacijo podatkov o dlaneh. Hkrati je težko doseči popolno rekonstrukcijo skeleta v realnem času.

Ne glede na svoje slabosti, globinska in vizualna detekcija predstavlja velik potencial za uporabo v prihodnosti, saj je na voljo kar nekaj cenovno sprejemljive strojne opreme, ki omogoča tovrstno zajemanje podatkov. Tukaj lahko omenimo barvno-globinske kamere (Kinect ter Asus Xtion), stereo sisteme kamer (npr. Bumblebee®2) in TOF (angl. Time-of-Flight) kamere. Vse naštete naprave se uporabljajo tudi na drugih področjih računalniškega vida in še posebej na področju robotike. Zaradi potenciala vizualnih metod bo diplomska naloga in praktična implementacija temeljila na globinskem prepoznavanju gest s pomočjo senzorja Kinect.

Področje vizualnega prepoznavanja lahko v grobem razdelimo na dve večji podskupini pristopov: pristop z uporabo kinematičnega modela (angl. 3D model based) in prepoznavanje značilnosti glede na izgled (angl. appearance based) [36].



Slika 3.3 Rentgenski posnetek roke in njena kinematična struktura [5, 25].

3.1 Uporaba kinematičnega modela

Pristop z uporabo modela poskuša določiti pozicijo in orientacijo dlani ter tudi stanje posameznih prstnih sklepov. S tovrstno popolno rekonstrukcijo je pristop idealen za uporabo v naprednejših interaktivnih aplikacijah ter v navideznih okoljih, ki ponujajo realistično interakcijo. Hkrati so zaradi natančnosti določanja parametrov tovrstne metode zahtevne in v realnem času težko izvedljive. V splošnem pristopi temeljijo na iskanju optimalnih kinematičnih parametrov, ki določajo strukturo 3D modela. Parametri so izbrani na podlagi najboljšega ujemanja 2D projekcije modela in slike segmentirane roke v realnosti [36]. S tem se opravi preslikava parametrov skeleta roke v realnosti v parametre kinematičnega modela. Slika 3.3 prikazuje rentgenski posnetek skeletne strukture roke in kinematični model, ki je bil predlagan v viru [25]. Sam model je lahko sicer bolj ali manj kompleksno definiran in odraža strukturo kosti roke v realnosti. Kot bomo videli pri opisu modela v praktični implementaciji, lahko njegovo kompleksnost določimo s številom prostostnih stopenj v vsakem segmentu.

Kot osnovni elementi za gradnjo modela se lahko (namesto bazične vektorske strukture) uporabijo primitivna geometrijska telesa (cilindri, poloble, elipsoidi) [21]. S tem se poenostavi generiranje projekcije modela, ki je v tem primeru predstavljena v obliki slike robov. Robovi modela se kasneje primerjajo z robovi, najdenimi na realnih slikah. Parametri modela se po navedbah omenjenega vira ocenijo s pomočjo filtriranja

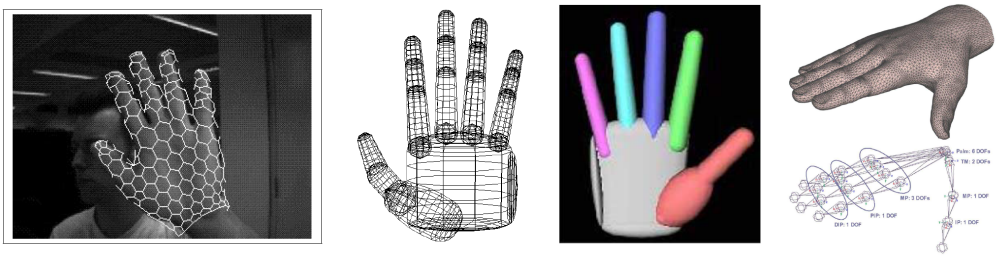
in hierarhične detekcije v obliki odločitvenega drevesa, ki minimizira geometrično napako robov med obema slikama. Odločitveno drevo je sestavljeno na podlagi množice podatkov, pridobljenih iz senzorične rokavice [21].

Predlagani so bili tudi alternativni načini generiranja modela. Tako je v [29] predstavljen model z možnostjo površinskega deformiranja. Površina modela je generirana iz mreže večkotnikov, ki se prilagodi obrisu dlani iz vhodne slike. Optimalno ujemanje modela z vhodno sliko je doseženo na podlagi statistične analize večje množice učnih primerov. Deformacija modela se torej oceni iz predhodno obstoječega znanja v obliki učne množice in informacij pridobljenih iz trenutne slike na vhodu sistema.

3D model uporablja tudi članek [23], kjer so pridobljeni podatki iz detekcije rok uporabljeni pri krmiljenju robotske roke. Posebnost modela je njegova optimizacija prostostnih stopenj s pomočjo predpostavke, da korenski segment prsta vpliva na naslednje segmente. V praksi to pomeni, da če korenski segment rotiramo za določen kot, potem se bo z nekim faktorjem v isto smer rotirala celotna veriga, ki predstavlja prst. Na ta način odpravimo nekaj prostostnih stopenj, ne da bi osiromašili funkcionalnost modela. Podobna predpostavka je bila uvedena v naši implementaciji kinematičnega modela. Podobno kot v viru [21] je tudi tukaj projekcija predstavljena kot slika robov, ki se primerja z sliko dlani iz realega okolja. Primerjanje se izvede z izračunom transformacije razdalje (angl. Distance transform) [6], ki predstavlja učinkovit način primerjanja binarnih slik, kamor spadajo tudi slike robov. Namesto filtriranja rešitev kot delcev je tukaj uporabljeno filtriranje z genetskimi algoritmi, ki do optimalnih rešitev pridejo iterativno z evolucijskimi pristopi.

V to poglavje lahko prištejemo še uporabo modela iz [19]. Model je zelo realistično upodobljen z uporabo poligonov, ki so povezani z matičnim skeletom in se z njim tudi ustrezno deformirajo. Poligoni modela se primerjajo z globinskimi podatki, pridobljenimi iz globinskega senzorja. Uporabljena je napredna metoda inteligentega filtriranja delcev, ki predstavljajo rešitve in stohastično (z vključenim faktorjem naključnosti) organizirano preiskovanje problemskega prostora.

Velja omeniti še izjemno robusten pristop in implementacijo iz [30]. Konkretno je za pridobitev globinskih podatkov uporabljen Kinect. Preko segmentacije kožnih odtenkov in globinske slike se pridobi segmentirana slika dlani uporabnika. Ta slika se primerja z vizualizacijo modela, ki je sestavljen iz geometrijskih primitivov. Prednost pristopa je paralelizacija najbolj računsko zahtevnih izračunov (predvsem primerjalne funkcije) na grafični procesni enoti. Hkrati je preiskovanje prostora realizirano z naprednim sistemom



Slika 3.4 Primeri modelov uporabljenih v virih [19, 21, 23, 29].

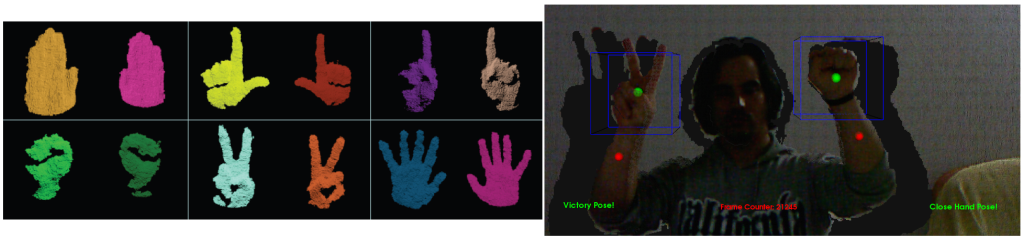
delcev (rešitev oz. hipotez), ki z vsako generacijo konvergirajo bližje optimalni rešitvi.

Objavljenih je sicer še veliko več podobnih pristopov, ki temeljijo na uporabi modelov. V splošnem jim je skupen podoben način delovanja na podlagi primerjanja silhuet ali obrisov ter iskanja optimalnih parametrov vsebujočih prostostnih stopenj. Iz tega sledi velika zahtevnost algoritmov za izvedbo in izvajanja v praktičnih aplikacijah. Velikokrat so za doseganje bolših rezultatov uporabljeni skupaj z ostalimi pristopi na tem področju, o katerih bomo več izvedeli v nadaljevanju.

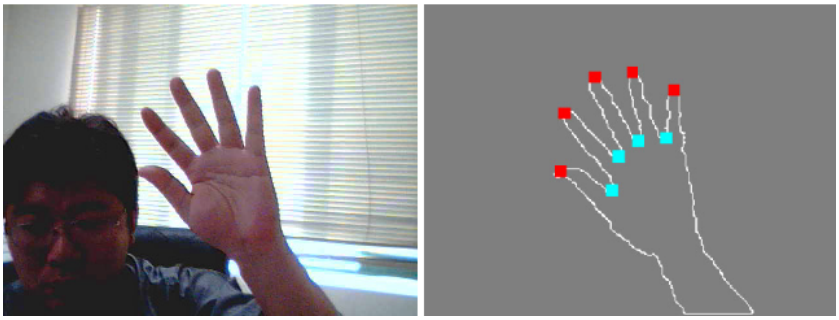
3.2 Prepoznavanje vizualnih značilnosti

Značilnosti so pridobljene neposredno iz informacije, ki jih vsebujejo vhodne slike. Zaradi tega je tovrstna detekcija v splošnem uporabljena predvsem za prepoznavanje omejenega števila gest. Pri tem ne potrebujemo dodatnih modelov, kot v prejšnjem poglavju. Hkrati to pomeni, da tukaj ne gre za iskanje optimalnih parametrov v prostoru razpoložljivih prostostnih stopenj. Informacije se pridobijo na podlagi geometričnih značilnosti in analiziranja robov, včasih pa se te metode kombinirajo tudi z uporabo predhodnega znanja (s strojnim učenjem) ali sistemov delcev in statistične analize. Pridobljeni podatki ne ponujajo popolne rekonstrukcije kot to omogočajo pristopi z uporabo modelov, so pa zaradi tega vizualne metode bistveno manj časovno in računsko zahtevne, ter se v večini primerov lahko izvajajo v realnem času.

Preslikava gest pri prepoznavanju značilnosti ni neposredna, kot je to predstavljeno v pristopu z modeli. Tipično se tukaj večkrat uporabljajo statistični klasifikatorji, ki vhodno sliko klasificirajo glede na množico značilnosti s katero je opisan v njej prikazan izgled dlani. Primer pristopa s klasifikatorjem je opisan v [34]. Opisan sistem je zasnovan tako, da prepoznava omejeno število različnih oblik dlani in prstov (slika 3.5). Kot



Slika 3.5 Baza oblik in detekcija dlani iz [34].



Slika 3.6 Detekcija na bazi kožnih odtenkov in analize robov iz [33].

klasifikator je uporabljena metoda podpornih vektorjev (angl. support vector machine - SVM), primerjave med vhodnimi slikami in obstoječimi primeri v bazi oblik pa se izvedejo s pomočjo posebnega deskriptorja. Deskriptor (ali tudi opisnik) je strokovno ime za zgoščeno obliko ključnih informacij, ki so pridobljene iz posamezne slike dlani. Uporaba deskriptorja tako omogoča lažje shranjevanje podatkov ter enostavnejšo primerjavo in klasifikacijo vhodnih primerov.

Za prepoznavanje gest torej načeloma ni potrebno prepoznavanje vseh parametrov dlani. Namesto tega več aplikacij uporablja atribute pridobljene iz preprostejših operacij na slikah, kot je npr. iskanje centroida regije, iskanje orientacije regije dlani preko izračuna višjih momentov, iskanje značilnih ostrih krivulj v robovih (konice prstov, slika 3.6) in podobne lastnosti. Pridobivanje tovrstnih atributov je hitro in učinkovito, ter hkrati do določene mere odporno na šum v vhodnih podatkih [36].

Segmentacije dlani se lahko lotimo z uporabo globinske slike (če imamo na voljo podatke iz globinskega senzorja). V tem primeru se dlani iščejo kot okončine, ki izrazito izstopajo iz postave uporabnika (primer takega iskanja izvaja knjižnica NiTE). Če globinska slika ni na voljo in imamo le slikovno informacijo, lahko dlani segmentiramo glede na

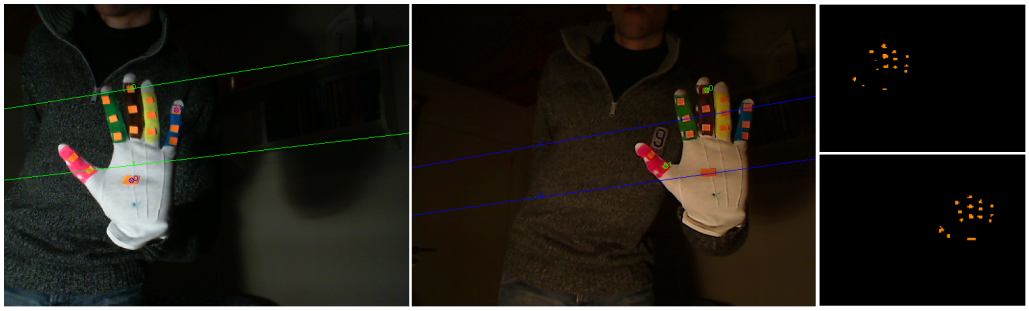
Slika 3.7 Detekcija na bazi kožnih odtenkov in analize robov iz [35].

barvno distribucijo odtenkov kožne barve. Metoda je popularna in jo najdemo v [33] in [35] ter tudi v drugih virih, vendar zahteva dovolj dobre svetlobne pogoje, da zadovoljivo deluje. Hkrati je v navedenih virih uporabljena še analiza pridobljenih robov, s pomočjo katere so bile najdene konice iztegnjenih prstov (sliki 3.6 in 3.7).

Inovativen pristop pridobivanja parametrov dlani je uporaba barvnih rokavic in drugih označb (markerjev). Implementacije so lahko bolj ali manj kompleksne. Tako na primer vir [38] navaja realizacijo prepoznavanja dlani, ki z uporabo barvnih rokavic ponuja zelo natančno rekonstrukcijo parametrov. Ta rekonstrukcija je povsem primerljiva z rekonstrukcijo, ki je dosežena z uporabo prej omenjenih kinematičnih modelov (slika 3.8). Prednost pristopa je v tem, da je barvna shema rokavic vnaprej poznana, kar lahko pomaga odpravljati problem prekrivanja. Hkrati je zaradi tega možna uporaba le ene kamere. Segmentiranje slike se prav tako izvaja glede na poznano barvno shemo uporabljenih rokavic. Določanje poze se izvede s filtriranjem slike, iz katere se tako odstrani morebiten šum. Zatem se slika optimizira s tem, da se skalira v zelo pomanjšano različico. Iz obstoječe baze majhnih sličic se zatem naredi selekcija njenih najbližnjih sosedov. Iz te množice se s primerjavami na nivoju slikovnih elementov izbere sličica z najboljšim ujemanjem, s tem pa se pridobijo tudi parametri oblike dlani ter prstov. Zahtevnost pristopa je predvsem priprava podatkovne baze, saj mora ta vsebovati veliko število sličic za doseganje dobrih rezultatov.

V [26] je predstavljen podoben, vendar nekoliko drugačen pristop, saj je uporabljen sistem stereo kamer. Pred dejanskim sledenjem je tako potrebno opraviti kalibracijo sistema kamer s pomočjo šahovnice. Po tem se izvede triangulacija ter ujemanje slik iz obeh kamer (slika 3.9). Sledi detekcija rok uporabnika preko rokavice, ki je obarvana

Slika 3.8 Pristop z uporabo barvne rokavice ter prikaz uporabe v virtualnem okolju [38].

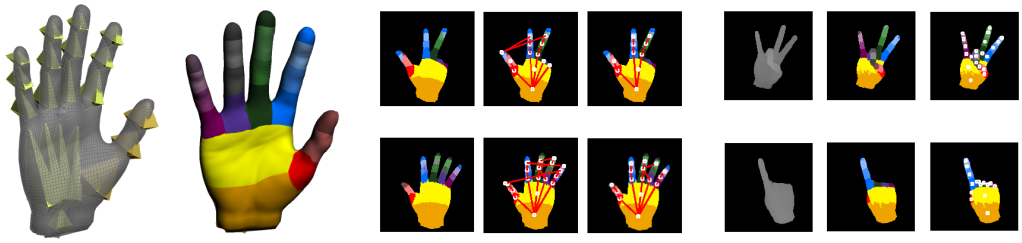


Slika 3.9 Uporaba stereo sistema kamer in barvne rokavice, desno rezultat detekcije (iz [26]).

v središču dlani in v pregibnih delih prstov. Glede na lokacije obarvanih regij se nato izvede rekonstrukcija dlani ter prstov, kot tudi globinska informacija (zaradi sistema stereo kamer). Zaradi zajemanja slik preko video kamer in barvne segmentacije, morajo biti ob uporabi obeh naštetih sistemov svetlobni pogoji seveda primerni.

3.3 Ostale metode, dopolnitve in izboljšave

Podrobnejši pregled in dodatno hierarhično ureditev metod na področju prepoznavanja dlani najdemo v [28]. Metode z uporabo modelov so tako še dodatno razdeljene na metode, ki uporabljajo modele skeleta ter na metode, ki uporabljajo volumetrične modele (s površino in volumnom). Podobno so metode prepoznavanja vizualnih značilnosti razdeljene na področja z uporabo baze vzorcev, na načine iskanja geometričnih in negeometričnih značilnosti ter na pristope, ki za prepoznavanje uporabljajo t.i. lastne vrednosti in lastne vektorje. Povsem ločeno vejo predstavljajo metode, ki za detekcijo potrebujejo predznanje ter tudi dodaten čas za učenje tega predznanja, preden so na voljo za uporabo. Sem spadajo nevronske mreže, skriti modeli Markova (angl. hidden Markov



Slika 3.10 Model iz [32] in njegova labelirana površina, proces iskanja ujemanja ter primera klasifikacije globinske slike dlani s pomočjo metod RDF in SVM.

models - HMM) ter pristopi, ki temeljijo na množici odločitvenih pravil, povezanih z avtomatom končnih stanj.

Pregled in uporaba nevronskih mrež (angl. artificial neural networks - ANN), ki se preko učenja prilagodijo regijam dlani je na voljo v [31]. Nevronske mreže so (podobno kot tudi drugi klasifikatorji) uporabljene predvsem za klasificiranje določenih poz dlani in prstov, zato se največ uporabljajo na področjih prepoznavanja znakovnega jezika. Poleg nevronskih mrež se kot klasifikatorji uporabljajo tudi naključni odločitveni gozdovi (angl. random decision forests - RDF) in že omenjene metode podpornih vektorjev (SVM). Še en primer uporabe klasifikacije s pomočjo RDF in SVM najdemo v [32].

Kot poseben primer lahko omenimo še uporabo pristopa deli in vladaj, skupaj z uporabo 3D modela [40]. Tukaj se iskanje loči na lokalne in globalne parametre. Lokalni parametri predstavljajo stanje prstov, globalni pa orientacijo in lokacijo dlani. Optimalna rešitev se nato sestavi iz obeh optimumov na lokalni in globalni ravni. Konkretno lokalni nivo predstavlja kinematični problem, ki se v tem primeru rešuje s pomočjo inverzne kinematike ob uporabi genetskih algoritmov. Globalni nivo pa tukaj predstavlja problem ocenjevanja poze dlani, ki se rešuje s pomočjo iskanja mediane najmanjših kvadratov v prostoru parametrov dlani.

4 Uporabljene knjižnice in tehnologije

V tem poglavju je najprej opisan Kinect, ki je bil v našem sistemu uporabljen za zajem globinske in vizualne informacije. Hkrati je opisano ogrodje OpenNI in zbirka algoritmov NiTE, ki skupaj nudita dostop do zajetih podatkov iz Kinecta. Nekaj besed je namenjenih tudi knjižnici OpenCV, s pomočjo katere se je izvajala analiza robov na slikah. V zadnjem delu poglavja je predstavljeno še ogrodje OpenFrameworks, v katerem je bil razvit celoten sistem za prepoznavanje položaja dlani in prstov.

4.1 Barvno-globinska kamera Kinect

Kinect v osnovi pri svojem delovanju uporablja strukturiran snop svetlobe in posebne leče za določanje fokusa. Strukturirana svetloba je generirana s pomočjo infrardečega projektorja, ki prostor osvetljuje z vnaprej znanim vzorcem. Ta vzorec je sestavljen iz navidez naključno postavljenih točkastih svetlobnih fragmentov. Kinect za analizo projiciranega vzorca uporablja še infrardečo kamero. Prvotni vzorec iz projektorja se namreč primerja z deformiranim vzorcem, ki ga zajame omenjena kamera. Glede na razlike med zajetim in projiciranim vzorcem se nato s pomočjo triangulacije izračuna

Slika 4.1 Kinect z označenimi komponentami ter primera zajetih slik (iz [27]).

dispariteto, preko katere se določi globinska slika prostora. Stereo analiza s pomočjo disparitete je mogoča zato, ker je razdalja med projektorjem in infrardečo kamero vnaprej definirana.

Kot rečeno je obenem uporabljena še informacija o globini, ki jo s pomočjo posebnih leč senzor pridobi preko fokusiranja. Določanje globine preko fokusa temelji na dejstvu, da so predmeti, ki so od kamere bolj oddaljeni posledično tudi bolj zamegljeni (v primerjavi z bližnjimi predmeti).

Proces določanja teles in identifikacije uporabnikov se na podlagi globinske slike izvede s pomočjo tehnik strojnega učenja. Na podlagi znanja iz obstoječih učnih primerov in množice klasifikatorjev (naključnih odločitvenih dreves) se izvede segmentacija in klasifikacija. Tako se pridobijo ključne točke, ki opisujejo skelet uporabnika.

Avtor tehnologije, s pomočjo katere deluje Kinect je sicer podjetje PrimeSense, ki je hkrati razvilo tudi gonilnike in ostalo pripadajočo programsko opremo (za detekcijo skeleta, pozicije rok ter različnih gest).

4.2 Ogrodje OpenNI

V letu 2010 je bil oblikovan neprofiten konzorcij OpenNI z namenom, da bi se aplikacije in senzorji, ki se uporabljajo na področju naravnih vmesnikov (angl. Natural Interaction) standardizirali ter da bi se ohranila kompatibilnost in interoperabilnost med njimi [7]. Med člani konzorcija velja omeniti PrimeSense, vodilno organizacijo v industriji interaktivnosti in 3D zaznavanja (razvoj senzorja Kinect). Poleg tega konzorcij sestavljajo Willow Garage, katerih specializacija je področje robotike (npr. ROS - robotski operacijski sistem). Pridruženi so med drugimi še neprofitni razvijalci knjižnice za obdelavo

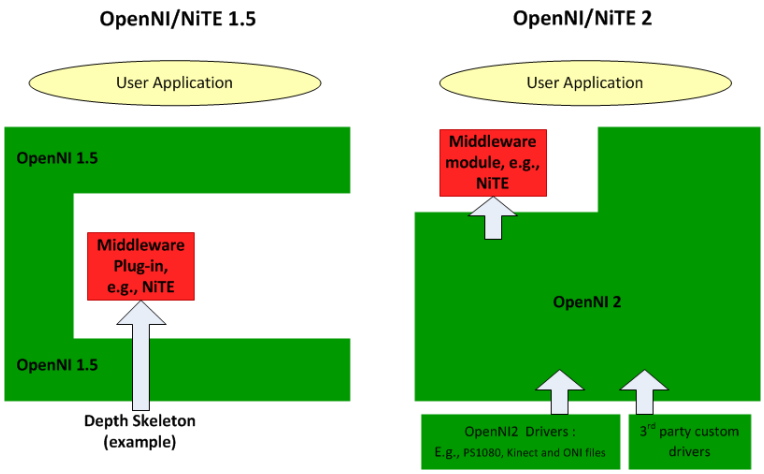
oblakov točk (Point Cloud Library - PCL) Open Perception. Član organizacije je tudi podjetje ASUS, ki razvija strojno opremo za področje 3D zaznavanja in detekcije (predvsem za področje aplikacij ter iger, ki za interakcijo uporabljajo celotno telo uporabnika, npr. globinski senzor Xtion). Poleg naštetih pa ne smemo pozabiti tudi na Side-Kick, vodilno produkcijsko hišo za igre, ki jih lahko uporabnik igra s pomočjo gibanja [8].

OpenNI je tako postal sinonim za najbolj obsežno ogrodje (angl. framework) za razvoj aplikacij, ki uporabljajo barvno globinske senzorje in druge naprave za 3D detekcijo in zaznavanje. Hkrati razvijalcem ponuja podporo za več platform in uporabo v različnih programskih jezikih. Za razvoj ogrodja v prihodnosti in vmesnih knjižnic (angl. middleware libraries) skrbi velika skupnost razvijalcev. Skupnost je razdeljena na raziskovalni del ter del zadolžen za razvoj, obstaja pa tudi del, ki se ukvarja z distribucijo že razvitih knjižnic in aplikacij. Razvijalcem je na voljo odprtokodni razvijalski komplet (SDK), ki predstavlja standardno orodje za razvoj programskih rešitev iz področja računalniškega vida in 3D zaznavanja.

Dostop do podatkov, ki jih posredujejo barvno globinski senzorji je razvijalcu omogočen preko različnih generatorjev. Konkretno v različici OpenNI knjižnice za ogrodje OpenFrameworks (ofxOpenNI) lahko uporabljamo sledeče generatorje:

- ofxDepthGenerator,
- ofxIRGenerator,
- ofxImageGenerator,
- ofxUserGenerator,
- ofxHandGenerator,
- ofxGestureGenerator.

Z generatorji lahko dostopamo do večine podatkov, ki nam jih posreduje senzor. Podatki so lahko globinske točke, postave ljudi pred senzorjem, detektirane točke dlani, globinska slika, slika barvne kamere ter slika infrardeče kamere. Za naš projekt je najbolj pomemben ofxHandGenerator, ki nam s svojo interno logiko NiTE algoritmov posreduje lokacijo rok uporabnika.



Slika 4.2 Arhitekturna razdeljenost knjižnice OpenNI (v uporabljeni različici 1.5 in novejši različici 2.0) [9].

4.3 Zbirka algoritmov NiTE

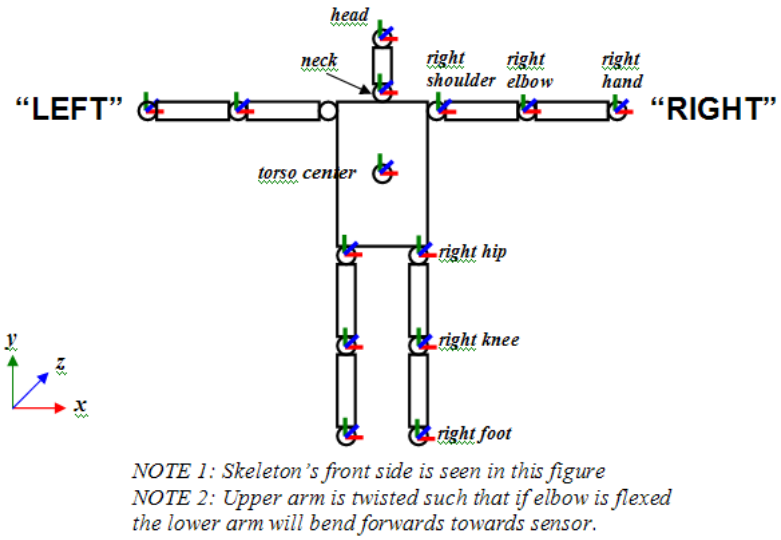
NiTE je skupek orodij in algoritmov, ki omogoča detekcijo uporabnikov na globinski sliki barvno-globinske kamere med njenim delovanjem. NiTE obsega tri skupine algoritmov. Prva skupina določa sledenje in prepoznavanje gest, ki jih uporabnik izvaja pred globinskim senzorjem. Algoritmi v tej skupini predpostavijo, da so roke, ki izvajajo geste v vidnem polju senzorja ter se ne prekrivajo z drugimi objekti ali uporabniki. Pri tem velja, da se nadzor izgubi, če uporabnik zapusti vidno polje globinskega senzorja. Domet delovanja v tem načinu je omejen na področje, ki sega od 1m do 3.5m daleč od senzorja. V tem načinu lahko NiTE razlikuje med nekaj definiranimi gestami, hkrati pa razvijalcu posreduje lokacijo sledene roke na globinski sliki in njeno globino v milimetrih. Pri sledenju uporabnikove roke se lahko zgodi, da točka, ki določa lokacijo dlani, v bližini drugih objektov preide na najbližji objekt. Zaradi tega je priporočeno, da roke ne postavljamo v bližino kateregakoli objekta, sem spada seveda tudi uporabnikova glava, trup in drugi deli telesa. Lahko pa pride tudi do tega, da je lokacija dlani med sledenjem izgubljena. Kljub naštetim izjemam in pomankljivostim sledenje deluje dokaj robustno.

Podprte geste v tem modulu so razdeljene na več podskupin. Ena izmed njih označuje geste, ki jih uporabnik izvede, ko želi prevzeti nadzor nad aplikacijo in hkrati nase fokusirati sledenje. Taki gesti sta mahanje roke (angl. wave) in t.i. klikanje (angl. click). Poleg naštetih obstaja še nekaj gest, ki jih lahko uporabnik aktivira, ko ima nadzor nad

aplikacijo. Sem spadajo geste kot so: premik roke v levo in desno (angl. swipe left, swipe right), dvig roke naslednjega kandidata ter premik roke kandidata. Geste imajo na voljo tudi status, ki pove v katerem stanju se določena gesta nahaja. Ta status ima lahko dve stanji in sicer določa začetek geste (angl. gesture started) ali njen zaključek (angl. gesture completed) [10].

Kot vidimo ima razvijalec iz strani knjižnice NiTE na voljo majhno število gest, poleg tega pa se nekatere med njimi lahko prožijo tudi, če jih uporabnik ni izvedel namerno (angl. false positives). S pomočjo natančnejšega določanja orientacije dlani in parametrov prstov bi lahko omogočili boljšo fleksibilnost nadzora nad aplikacijami in izvajanje večjega števila različnih gest. Uporabniku bi lahko ponudili tudi ustvarjanje novih gest, ki bi bile določene s spreminjanjem parametrov dlani in prstov na nekem časovnem intervalu. Optimalna rešitev bi seveda bila, da bi v vsakem trenutku imeli dostop do vseh prostostnih stopenj kinematičnega modela roke, ki je popolnoma poravnan z roko uporabnika. V tem primeru je iskanje optimalnega ujemanja modela z dlanjo računsko in prostorsko zelo zahteven problem, ki je v realnem času težko rešljiv.

Druga skupina metod v knjižnici NiTE ponuja segmentacijo uporabnikov. To pomeni, da je v sceni lahko navzočih več subjektov, NiTE pa jih poskuša označiti (labelirati njihove regije v globinski sliki) ter jim določiti unikaten identifikator (ID), pod katerim bodo na voljo razvijalcu. Izhodni podatki so v tem primeru labelirana globinska slika, ki vsakemu pikslu določa pripadnost določenemu uporabniku. To globinsko sliko uporablja tudi tretja skupina algoritmov, ki skuša določiti skelet vsakega zaznanega uporabnika. Morebitne napake pri omenjenem labeliranju lahko povzročijo možna odstopanja pri določanju poze in skeleta uporabnikov. Poleg teh odstopanj so znane še nekatere druge omejitve, ki jih je nemogoče odpraviti. Upoštevati moramo, da lahko pride do napak pri segmentaciji, če se osebek premika v bližini sten ali drugih večjih objektov. Poleg tega lahko pride do napačne segmentacije dveh ali večih oseb, če se njihove poze med seboj dotikajo. Nezaželeno je premikanje globinskega senzorja med procesom segmentacije, saj zaradi spremembe vidnega polja prav tako lahko pride do odstopanj. Če uporabnik zapusti vidno polje, se njegov identifikator lahko izgubi ali pa je nepravilno zamenjan z identifikatorjem drugega uporabnika. V kolikor je osebek izven sledilnega polja več kot 10 sekund, potem se njegov identifikator sprosti in sproži se dogodek, ki določa da je bil uporabnik izgubljen. V tem času ima razvijalec možnost posodobiti višje nivoje aplikacije, da se izguba uporabnika ustrezno upošteva [10].



Slika 4.3 Struktura segmentov in sklepov, ki določajo skelet uporabnika [10].

Zadnja skupina algoritmov rešuje problem določanja skeleta posameznih uporabnikov. Pri tem se opira na labelirano globinsko sliko iz prejšnje skupine. Hkrati predpostavlja, da je večina zgornjega dela trupa v vidnem delu senzorja in ni zakrita z drugimi objekti. Optimalna razdalja za delovanje je glede na podane specifikacije 2.5m. Ohlapna oblačila na uporabnikih niso zaželjena, saj otežujejo optimalno ocenjevanje segmentov v skeletu. Določanje skeleta se lahko v trenutni različici algoritmov (1.5) izvaja brez predhodne kalibracije sistema, kar je velika prednost. V predhodnih različicah se je namreč uporabnik moral postaviti v pozo za kalibracijo in počakati, da je sistem zgradil strukturo njegovega skeleta. Sedaj se skelet (po relativno kratkem času) določi samodejno s pomočjo avtomatske kalibracije, po tem, ko uporabnik stopi v vidno polje globinskega senzorja. Kvaliteta zaznanega skeleta se s časom izboljšuje. Čeprav se zdi, da se detekcija opravi skoraj v trenutku, proces kalibracije traja nekaj sekund, ter tako točneje določi segmente skeleta. Končan proces kalibracije odstrani šum in morebitne nepravilnosti v strukturi skeleta. Upoštevati moramo, da samodejna kalibracija deluje le na stoječih osebkih, katerih postava je v večini vidna na globinski sliki in so hkrati od senzorja oddaljeni več kot 1m [10].

Pri detekciji segmentov in sklepov algoritmi NiTE uporabljajo t.i. informacijo o zaupanju, ki določa kvaliteto parametrov posameznih segmentov. Tako npr. zaupanje z

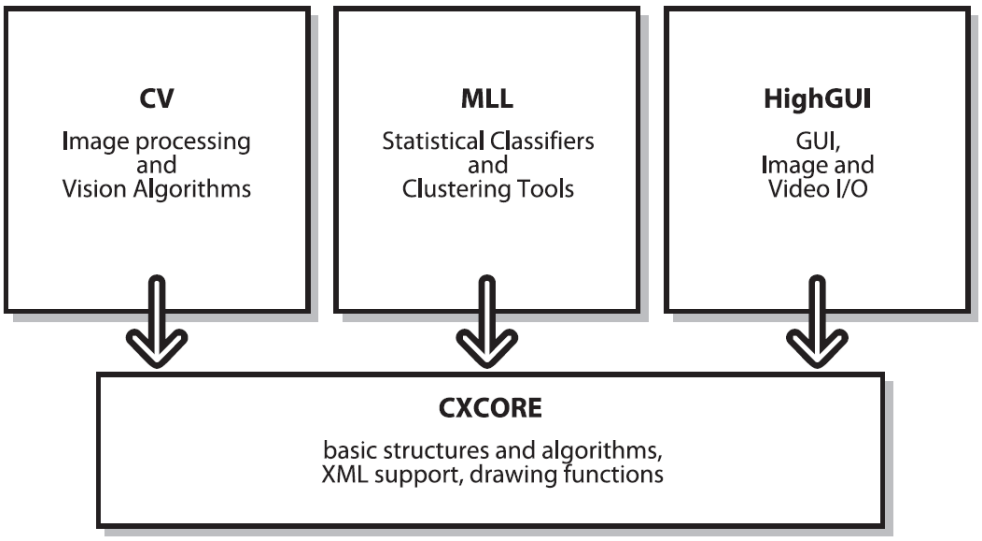
vrednostjo 1 pomeni, da detekcija deluje pravilno. Zaupanje z vrednostjo 0 pa pomeni, da je pri detekciji prišlo do težav in da pridobljeni podatki o sklepih niso več verodostojni. Do tega lahko pride s prekrivanjem telesnih delov ali izgube sledenja zaradi drugih podobnih razlogov. V primerih, da so vhodni podatki uporabnika omejeni ali nepopolni se uporabijo posebne preiskovalne funkcije (hevristike), ki poskušajo oceniti in popraviti nepravilno pozicionirane dele skeleta. Če se zgodi, da pozicijo ali orientacijo segmenta določijo omenjene funkcije, potem se zaupanje primerno nastavi na vrednost 0.5. Tako bo na primer popolnoma zakrita okončina uporabnika (roka ali noga) imela stopnjo zaupanja 0. V popravku bo usmerjena navpično proti tlom (angl. downward pointing), saj je to najbolj optimalna ocena, ki jo lahko podajo omenjene funkcije [10].

4.4 Knjižnica OpenCV

OpenCV je široko uporabljena odprtokodna knjižnica za obdelavo slik in procesiranje podatkov na področju računalniškega vida. Implementirana je v jeziku C++, izrablja zmogljivosti večjedrnih procesorjev in je na voljo na večih platformah (Linux, Windows in Mac OS X) [18].

Arhitekturno je knjižnica razdeljena na 5 ločenih skupin. Slika 4.4 prikazuje 4 skupine, od katerih komponenta CV (angl. Computer Vision) podpira izvajanje osnovnih operacij nad slikami ter naprednejše algoritme iz računalniškega vida (detekcija gibanja in sledenje, prepoznavanje vzorcev in kalibracija). Komponenta ML (angl. Machine Learning), kot že njeno ime pove, vsebuje različne klasifikatorje in druga orodja, ki se uporabljajo na področju strojnega učenja. Skupina HighGUI nudi podporo za nalaganje in shranjevanje slik in videa ter hkrati vsebuje elemente grafičnega vmesnika. Komponenta CXCore povezuje naštete komponente ter vsebuje osnovne podatkovne strukture. Hkrati je zadolžena za upravljanje s pomnilnikom in obravnavo morebitnih napak. Vsebuje tudi funkcije za računanje z matrikami in osnovne operacije risanja na zaslon. Zadnja komponenta, ki je na sliki 4.4 ne vidimo pa se imenuje CvAux in je eksperimentalne narave. Vsebuje opuščene funkcionalnosti ter poskusne implementacije algoritmov v razvoju. V nasprotju z ostalimi komponentami je CvAux slabše dokumentirana in vsebuje module, ki niso neposredno vključeni v jedro knjižnice [18].

V implementaciji sistema je bila knjižnica uporabljena v obliki dodatka za ogrodje OpenFrameworks (ofxOpenCV). Uporabljen je bil le delček velike zbirke funkcij in algo-



Slika 4.4 Arhitekturna razdeljenost knjižnice OpenCV (iz [18]).

ritmov. Konkretno so bile za detekcijo dlani zanimive funkcije za iskanje robov (s pomočjo objekta ofxContourFinder) in metode za izvajanje morfoloških operacij (s pomočjo metod dilate() in erode()) [11].

4.5 Ogrodje OpenFrameworks

OpenFrameworks (OF) je odprtokodno programsko ogrodje za hiter razvoj grafičnih in drugih kreativnih aplikacij v programskem jeziku C++. Ogrodje je oblikovano kot združena celota različnih knjižnic. Razvijalec se knjižnic na najnižjem nivoju skoraj ne zaveda, saj so učinkovito ovite v abstrakten sistem razredov in metod, ki delujejo kot lepilo in povezujejo posamezne komponente v enotno ogrodje. Najpomembnejše knjižnice [22], ki jih vsebuje jedro ogrodja so:

- OpenGL, GLEW, GLUT, libtess2 in cairo (grafična podpora),
- rtAudio, PortAudio ali FMOD in Kiss FFT (analiza zvoka, rokovanje z audio vhodom in izhodom),
- FreeType (podpora za uporabo pisav),
- FreeImage (odpiranje in shranjevanje slik),

- Quicktime, GStreamer in videoInput (predvajanje in zajemanje videa),
- Poco (spekter pomožnih metod, proženje dogodkov),
- OpenCV (uporaba metod iz računalniškega vida),
- Assimp (nalaganje 3D modelov).

Koda ogrodja je napisana z namenom podpore na različnih platformah (poudarek na prenosljivosti). Trenutno je podprtih pet operacijskih sistemov (Windows, Mac OS X, Linux, iOS in Android), prav tako so podprta štiri integrirana razvojna okolja (XCode, Code::Blocks, Visual Studio ter Eclipse). Programski vmesnik za razvoj (API) je preprost in razumljiv za uporabo. Velika prednost ogrodja je natančno definirana dokumentacija, razširjenost uporabe in velika skupnost razvijalcev ter aktivna podpora uporabnikom. Zaradi modularnosti in različnosti vključenih komponent, lahko tako z enim programskim vmesnikom uporabljamo metode iz kompleksnih knjižnic (npr. OpenCV), obenem lahko učinkovito izrabimo grafično strojno opremo našega sistema in med seboj povezujemo tudi druge periferne naprave, kot so npr. kamere in različni senzorji. Hkrati je okolje zelo razširljivo, saj omogoča dodajanje in ustvarjanje dodatnih komponent [12].

Kot rečeno lahko v ogrodje po želji vključimo dodatne knjižnice v obliki dodatkov (angl. addons). V ta namen obstaja organizirana spletna zbirka[1], kjer so objavljeni vsi aktualni dodatki, ki jih lahko vključimo v naših aplikacijah. Poimenovanje dodatkov je določeno z dogovorom, po katerem se ime dodatka in pripadajoči razredi s programsko kodo začnejo s predpono ofx, kar pomeni OpenFrameworks eXtension. Tovrstno poimenovanje je uporabno pri ločevanju programske kode dodatkov od drugih komponent ogrodja. Običajno dodatki vsebujejo tudi primer projekta, ki prikazuje uporabo dodatnih komponent. Na ta način se razvijalec hitro seznani z delovanjem in organizacijo izbranega dodatka. Za enostavno vključevanje dodatkov, je struktura direktorijev ogrodja vnaprej določena. Tako je za dodatne komponente rezerviran korenski direktorij addons/. Direktorij libs/ pa npr. vsebuje omenjene knjižnice, ki so v jedru ogrodja. Najdemo tudi direktorij examples/, ki je namenjen projektom, ki prikazujejo, kako se določene komponente uporabljajo. Direktoriji, v katerih se razvijajo aplikacije se nahajajo v korenskem direktoriju z imenom apps/. V novejših različicah (0071+) najdemo tudi direktorij projectGenerator/, ki vsebuje generator projektov za vsako platformo, kar še dodatno

[1] www.ofxaddons.com

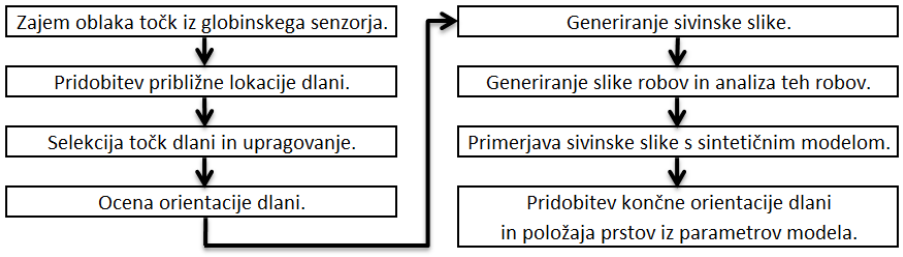
Slika 4.5 Hierarhična urejenost direktorijev (levo) in osnovna arhitektura aplikacije v ogrodju OpenFrameworks (desno) [22].

olajša razvoj in zmanjša čas, ki je potreben za začetno konfiguracijo projekta in razvojnega okolja. Omenjeni generator projektov pri ustvarjanju novega projekta upošteva strukturo direktorijev in tako namesto razvijalca poskrbi za vključitev vseh potrebnih datotek ter dodatkov in njihovih relativnih poti, ki morajo biti vključene v projekt, da se ta lahko prevede [12]. Organiziranost direktorijev je prikazana na sliki 4.5.

Arhitekturno urejenost najdemo tudi v vsaki aplikaciji, ki jo razvijemo z OpenFrameworks ogrodjem (slika 4.5 desno). Aplikacija je v osnovi definirana z razredom, ki ima že določene osnovne metode, ki se ustrezno kličejo. Za inicializacijo objektov tako obstaja metoda `setup()`, ki se pokliče enkrat, ob samem zagonu aplikacije. Zatem se izmenično izvajata metodi `update()` in `draw()`. Metoda `update()` je namenjena posodabljanju logike naše aplikacije, namen metode `draw()` pa je predvsem izris slikovnih vsebin in osveževanje grafičnega okna. Poleg tega je v aplikacijo vgrajena podpora za branje znakov iz vhodnih naprav, kot je denimo tipkovnica in miška. V ta namen imamo več metod (npr. `keyPressed`, `keyReleased`, `mousePressed`, `mouseMoved` itn.), ki se kličejo glede na dogodke (angl. events). Dogodek se proži, ko npr. pritisnemo na tipko ali premaknemo kazalec na nove koordinate na zaslonu. Ob zaustavitvi aplikacije nam je na voljo še metoda `exit()`, v kateri lahko zapremo odprte podatkovne tokove in sprostimo morebiten alociran pomnilnik [20].

5 Rešitev problema in implementacija

Rešitev problema bi lahko povzeli s shemo, prikazano na sliki 5.1. Iz zajetega oblaka globinskih točk se na podlagi približne lokacije dlani izbere množico ključnih točk. Iz njih se oceni orientacija dlani. Zatem se generira sivinska slika dlani ter slika robov. Z analizo robov se pridobijo lokacije prstnih konic ter središče dlani. Sivinska slika se nato primerja s projekcijo sintetičnega modela. Na podlagi analize robov in izvedenih primerjav z modelom se določi končna orientacija dlani ter položaj prstov. V nadaljevanju poglavja sledi podrobnejši opis vseh faz procesa detekcije.



Slika 5.1 Proces prepoznavanja značilnosti dlani v implementiranem sistemu.

Slika 5.2 Primer celotnega oblaka globinskih točk, ki nam ga posreduje Kinect.

5.1 Pridobivanje oblaka točk

Selekcijo točk iz celotnega oblaka (slika 5.2), ki ga generira Kinect lahko pridobimo iz objekta ofxUserGenerator [13], ki ima v ta namen implementirano metodo getWorldCoordinateAt. Tej metodi podamo x in y koordinato točke, kjer želimo pridobiti globino. Prav tako lahko razlikujemo med uporabniki, ki so trenutno v sceni, zato imamo možnost, da metodi kot parameter podamo še ID uporabnika. Za upoštevanje vseh točk (ne glede na labeliranje uporabnikov) lahko kot ID podamo 0.

5.2 Določanje pozicije dlani

Za pridobitev točk v bližini uporabnikove roke potrebujemo najprej njeno lokacijo. Če je sledenje rok omogočeno in če je roka uporabnika zaznana, nam njeno približno lokacijo posreduje objekt ofxHandGenerator [13]. Pozicijo roke iz generatorja lahko pridobimo kot normalizirano vrednost na intervalu med 0.0 in 1.0, lahko jo pridobimo v merah projekcije (kar pomeni glede na dimenzije zaslona), možna pa je tudi pridobitev globine v milimetrih. S pridobljeno lokacijo roke, ki nam jo posreduje vtičnik NiTE preko ofxHandGeneratorja lahko filtriramo oblak točk in izločimo le tiste, ki so v njeni neposredni bližini (primera detektiranih točk dlani sta na sliki 5.3).

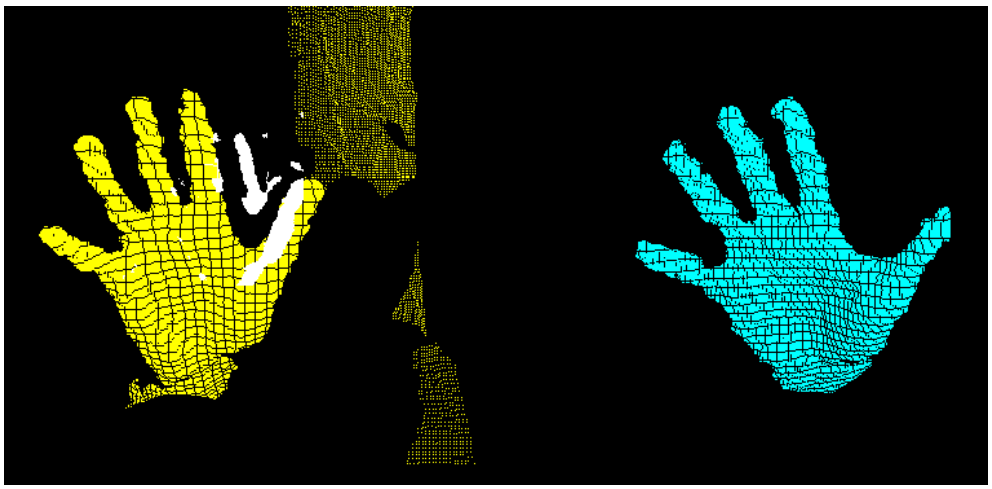
Slika 5.3 Primera globinskih slik s pravilno (in zgoraj tudi napačno) detekcijo dlani ter pripadajočih IR slik iz infrardeče kamere.

5.3 Selekcija točk v okolici dlani

Za izbiro bližnjih točk definiramo pravokotno območje zanimanja (angl. Region of Interest - ROI) s središčem v točki, ki nam jo poda ofxHandGenerator. Dolžino stranic tega območja določimo s testiranjem. Pri določanju vrednosti dolžin stranic (in s tem mej upoštevanih točk oblaka) je pomembno predvsem to, da s spreminjanjem dolžin stranic in s tem površine pravokotnika v obsegajočo regijo (ROI) vključimo vse globinske točke, ki pripadajo roki uporabnika. Hkrati moramo upoštevati, da se ločljivost oblaka točk, ki določa uporabnikovo roko, zmanjšuje glede na naraščanje razdalje od globinskega senzorja. To pomeni, da bo na večji razdalji roke od senzorja velikost oblaka manjša, temu pa je potrebno dinamično prilagajati tudi velikost regije. S tem zmanjšamo možnost, da bi poleg oblaka točk roke upoštevali točke morebitnih drugih objektov v bližini, hkrati pa zmanjšamo tudi število točk, ki bodo v nadaljevanju uporabljene za procesiranje.

5.4 Filtriranje točk po globini

V tem trenutku imamo iz senzorja pridobljen oblak točk v bližini roke, ki je omejen z definirano regijo zanimanja (slika 5.4 levo). Naslednji korak je izločitev točk, ki glede

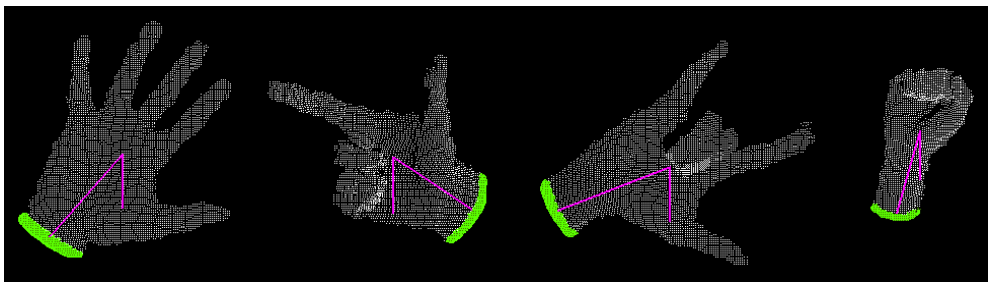


Slika 5.4 Selekcija točk v ROI območju najdene pozicije dlani (levo) in izločitev točk, ki izrazito izstopajo, npr. točke ozadja (desno).

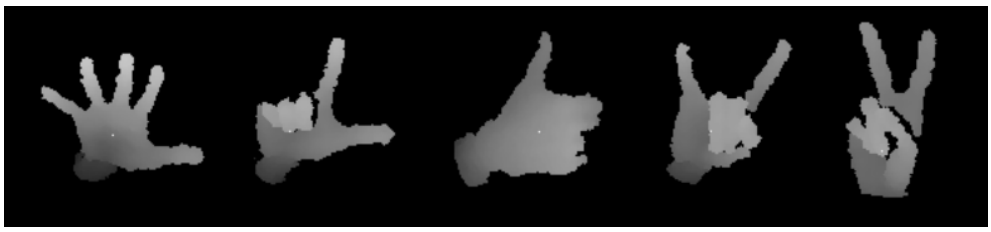
na globino ne pripadajo roki, temveč pripadajo ozadju oz. drugim oddaljenim objektom. Izločitev lahko izvedemo s preprostim upragovanjem globine, kjer izločimo točke brez globine in tiste točke, ki glede na podan mejni parameter globine izrazito izstopajo (slika 5.4 desno). Parameter za upragovanje globine prav tako pridobimo s testiranjem. Pri upragovanju predpostavljamo, da uporabnik roke drži relativno daleč od telesa oz. drugih objektov. Če je roka nekemu objektu po globini bližje, kot je določen parameter za upragovanje, potem globinske točke tega objekta ne bodo izločene v celoti. Posledično oblak točk v regiji ne bo določal pravilne oblike roke. Težava z določanjem oblaka točk dlani v neposredni bližini drugih objektov izvira že iz generatorja (ofxHandGenerator), ki nam posreduje lokacijo dlani. V primeru premajhnih razlik v globini, algoritem za sledenje namreč predpostavi, da je bila roka izgubljena in sledenje ni več na voljo. Selekcijo globinskih točk bi v takih primerih lahko izboljšali z naprednejšimi metodami izbiranja točk, kot je na primer segmentacija.

5.5 Ocenjevanje orientacije oblaka točk

Orientacija oblaka točk se okvirno oceni iz vektorja med dvema ključnima točkama. Prva točka je centroid vseh točk v oblaku. Centroid je definiran kot aritmetična sredina koordinat vseh točk v smereh x, y in z in predstavlja težišče oblaka točk (regije). Da dobimo drugo točko vektorja pa izračunamo centroid točk, ki ležijo v bližini zapestja.



Slika 5.5 Primeri ocenjene orientacije oblaka točk z dlanjo v različnih pozah.



Slika 5.6 Primeri slik dlani, pridobljenih preko preslikave oblaka točk.

Množico točk zapestja dobimo tako, da izberemo vse točke oblaka, ki so od središča regije zanimanja oddaljene z dovolj veliko razdaljo, da spadajo pod obrobne točke regije (zeleno obarvane regije primerov na sliki 5.5). Razlika med obema točkama predstavlja vektor orientacije (prav tako prikazan na sliki 5.5). Metoda v večini primerov deluje zadovoljivo, problem ponovno nastane, če upragovanje v prejšnji točki ne uspe izločiti vseh točk ozadja. V takih primerih lahko pride do večjih napak pri določanju orientacije.

5.6 Generiranje sivinske slike točk

Sivinsko sliko oblaka točk zgeneriramo zato, da lahko na njej izvajamo operacije, ki jih omogoča knjižnica OpenCV. Preslikava oblaka točk v sivinsko sliko poteka tako, da preslikamo x in y koordinate vsake globinske točke v pripadajoč slikovni element na sivinski sliki enake širine in višine, kot so dimenzije oblaka točk dlani. Vrednost slikovnega elementa (sivinski nivo) pa se določi s pomočjo globinske informacije (koordinate z) vsake točke, katero še prej normaliziramo in omejimo na vrednosti med 0 in 255. Omejitev globine izvedemo zato, da jo lahko v slikovni element zapišemo v obliki barve. Na ta način globinsko sliko predstavimo z 255 različnimi sivinskimi nivoji (slika 5.6).

5.7 Analiza robov sivinske slike

Iz sivinske slike pridobimo sliko robov s pomočjo knjižnice OpenCV v obliki dodatka za ogrodje OpenFrameworks (ofxOpenCV). Na voljo nam je objekt ofxCvContourFinder [11], ki ima implementirano metodo findContours. Metoda kot parametre sprejme sivinsko sliko, na kateri bomo iskali robove, določimo lahko velikost najmanjše in največje regije za iskanje robov, poleg tega pa lahko vključimo ali izključimo iskanje lukenj ter izvajanje aproksimacije nad najdenimi robovi [11]. Z omenjeno metodo nad sliko opravimo iskanje robov in pridobimo največjo regijo, ki predstavlja oris uporabnikove roke. Metoda robove poišče na podlagi lokalnih sprememb v sliki. V splošnem lokalne spremembe v intenziteti določimo tako, da sliko odvajamo. Odvod slike dobimo tako, da odštevamo sosednje piksle v obeh dimenzijah slike in tako pridobimo vrednosti razlik sosednjih pikslov. Izrazitost teh razlik nam pove, s kako magnitudo se slika spreminja. Robove tako zaznamo na mestih, kjer so razlike sosednjih pikslov zelo velike. Za iskanje robov se uporabljajo različni operatorji, ki nastopajo v obliki matrik.

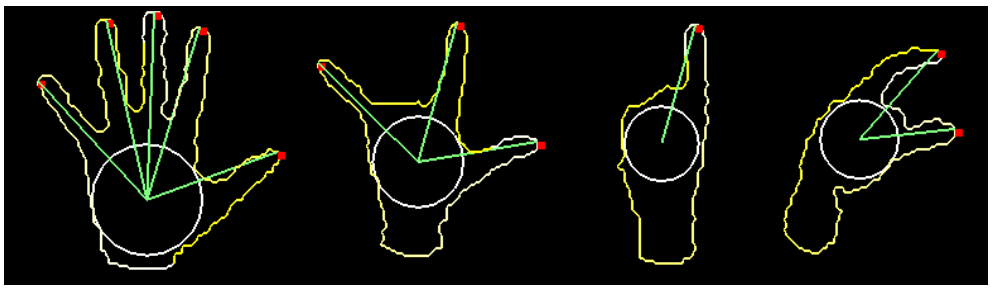
5.7.1 Iskanje centra dlani

Izračun težišča dlani (centroid vseh točk), ki ga pridobimo z izračunom aritmetične sredine koordinat oblaka točk v smereh x, y in z ne predstavlja natančnega središča dlani, saj se spreminja skupaj s silhueto dlani in prstov. Boljši pristop je iterativno iskanje središča preko analiziranja robne krivulje. Ker vemo, da je dlan vedno kompaktne oblike (ne glede na to ali so prsti iztegnjeni ali ne), jo lahko v grobem opišemo s krožnico. Pri tem iščemo krožnico z največjim radijem, ki jo še lahko vsebuje regija omejena z robno krivuljo (iskanje največje možne včrtane krožnice). Središče najdene krožnice pa tako določa tudi sam center dlani.

Za iskanje potrebujemo urejen seznam točk na robu ter seznam kandidatov za središče dlani. Urejen seznam robnih točk nam posreduje knjižnica OpenCV preko že omenjenega objekta ofxCvContourFinder. Pridobimo lahko točke poljubne regije, ki je bila najdena na vhodni sliki. Ker je na naši vhodni sliki dlani le ena regija (uporabnikova dlan) lahko robne točke regije dobimo brez dodatnega preverjanja. Množico kandidatov izmed katerih eden predstavlja središče dlani pa moramo pridobiti ročno iz okolice težišča dlani. Pri tem izberemo empirično določen radij, ki je v implementaciji določen s 60% razdalje od težišča do najbolj oddaljene točke v oblaku točk. Glede na ta radij iz celotnega oblaka

Algoritem 1 Iskanje središča dlani

```
Require: array [contour points], array [palm center candidates]
  min_distance ← MAX
  max_min_distance ← 0
  point min_candidate
  point max_min_candidate
  for each palm_point in [palm center candidates] do
    min_distance ← MAX
    for each contour_point in [contour points] do
      current_distance ← distance(contour_point, palm_point)
      if current_distance < min_distance then
        min_distance ← current_distance
        min_candidate ← palm_point
      end if
    end for
    if min_distance > max_min_distance then
      max_min_distance ← min_distance
      max_min_candidate ← min_candidate
    end if
  end for
  return max_min_candidate, max_min_distance
```



Slika 5.7 Primeri najdenih dlani (določenih s krožnicami) na nekaterih slikah robov.

točk zatem izberemo podmnožico elementov (v okolici težišča), ki predstavljajo kandidate za določitev središča dlani. Kandidate zaradi zahteve po izvajanju v realnem času še dodatno izločimo in za manjšo kompleksnost izvajanja uporabimo npr. le vsako drugo točko v tej podmnožici. Sledi linearna transformacija izbranih kandidatov v koordinatni sistem slike z robovi. S tem se opravi poravnava s koordinatnim sistemom v katerem so robne točke regije (ki so pridobljene na podlagi robov na sliki).

Prikazan algoritem 1, povzet po [14] prikazuje logiko iskanja središča dlani. Konstanta MAX, ki je uporabljena v algoritmu je definirana kot velika začetna vrednost (ugotovljena s testiranjem), ki zagotovi konvergenco pri iskanju minimalne razdalje. Točneje bi jo lahko definirali kot razdaljo med najbolj oddaljenima točkama in bi lahko bila določena kot dolžina diagonale slike na kateri se izvaja iskanje središča dlani. Podatkovna struktura *point* predstavlja točko v dvodimenzionalnem prostoru in je določena s koordinatama x in y. Ključna beseda *array* v algoritmu pa določa seznam elementov (v tem primeru omenjenih točk). Uporabljena funkcija $distance(P1, P2)$ vrne razdaljo med dvema točkama v dvodimenzionalnem prostoru slike in je za točki

$$P1 = (x1, y1), P2 = (x2, y2)$$

definirana kot

$$distance(P1, P2) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}.$$

Algoritem 1 kot rešitev vrne točko središča dlani v koordinatnem sistemu slike in podatek o radiju največje krožnice, ki jo lahko iz dobljenega središča očrtamo orisu uporabnikove dlani. Grafično je rešitev prikazana na slikah 5.7 in 5.8 v obliki bele krožnice.

5.7.2 Določanje konic prstov

Drugi del analize robov se ukvarja z določanjem točk, ki predstavljajo konice iztegnjenih prstov. Pri tem se ponovno (kot pri določanju središča dlani) uporabi urejen seznam N robnih točk. Za vsako $i$to robno točko se izračuna kot Θ, ki ga objemata dva vektorja [33, 35]. Prvi vektor je definiran kot razlika med točkama i in $(i-k)$. Drugi vektor je podobno kot prvi definiran kot razlika med točkama i in $(i+k)$. Vrednost k je empirično določena vrednost, ki torej pomeni razliko med indeksi sosednjih točk. Manjši kot je k, večja bo gostota točk, na katerih se bo izračunal kot, ki ga objemata definirana vektorja za vsako točko. Podobno bo z večjimi vrednostmi k, razmik med točkami večji in tako bo tudi gostota teh točk manjša. Optimalna vrednost k se določi s pomočjo testiranja (v našem primeru je bila uporabljena vrednost 12).

Kot med vektorjema, Θ, lahko dobimo z izračunom skalarnega produkta vektorjev [24], ki je za poljubna vektorja $\vec{a}$ in $\vec{b}$ določen z izrazom

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos(\Theta) \Rightarrow \cos(\Theta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}.$$

Za določitev konic iztegnjenih prstov ni dovolj, da poznamo le kot Θ, saj lahko na ta način pridobimo tudi točke, ki predstavljajo vrzeli med prsti (zaradi prstom podobne koničaste oblike). Tako potrebujemo dodaten kriterij za ločevanje - pomagamo si lahko z vektorskim produktom [15]. S pomočjo vektorskega produkta lahko izračunamo normalo na ravnino, ki jo določata vektorja in glede na usmerjenost te normale ločimo med točkami, ki predstavljajo vrzeli med prsti in točkami, ki predstavljajo konice iztegnjenih prstov. Vektorski produkt vektorjev $\vec{a}$ in $\vec{b}$ je določen z izrazom

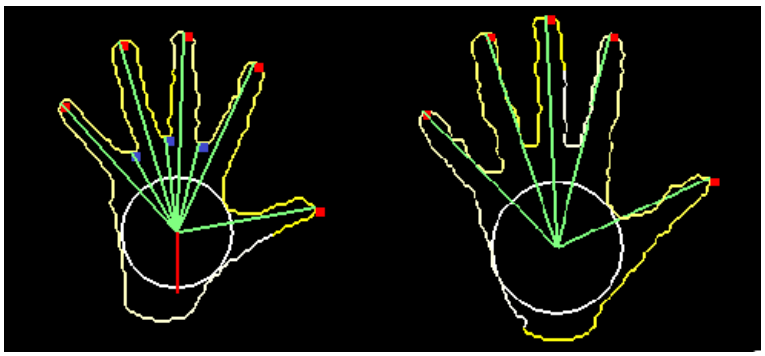
$$\vec{a} \times \vec{b} = |\vec{a}||\vec{b}| \sin(\Theta)\vec{n},$$

kjer je $\vec{n}$ enotski vektor normale, ki je pravokoten na ravnino, določeno z vektorjema $\vec{a}$ in $\vec{b}$. Vrzeli med prsti tako lahko filtriramo z uporabo pogoja $normal.z \geq 0$, kot je prikazano v algoritmu 2.

Algoritem 2 Iskanje konic iztegnjenih prstov

Require: *array* [contour points]
  NUM_POINTS ← *size*([contour points])
  STEP ← 12
  THRESH_ANGLE ← $\pi/3$
  min_angle ← 2π
  min_index ← -1
  array [finger tips] ← [$\emptyset$]
  for i=STEP to (NUM_POINTS + STEP) **do**
    point prev_point ← [contour points][(i-STEP)]
    point curr_point ← [contour points][(i)%NUM_POINTS]
    point next_point ← [contour points][(i+STEP)%NUM_POINTS]
    vector prev_vector ← curr_point - prev_point
    vector next_vector ← curr_point - next_point
    cosine ← (prev_vector · next_vector)/(|prev_vector| |next_vector|)
    angle ← arccos(cosine)
    vector normal ← prev_vector × next_vector
    if angle < THRESH_ANGLE **and** normal.z ≥ 0 **then**
      if min_angle ≥ angle **then**
        min_index ← i
        min_angle ← angle
      end if
    else if min_index ≠ -1 **then**
      insert([finger tips], [contour points][(min_index)%SIZE])
      min_angle ← 2π
      min_index ← -1
    end if
  end for
  return *array* [finger tips]

Algoritem 2 na podlagi mejne vrednosti THRESH_ANGLE poišče konice prstov, ki ustrezajo danemu pogoju. Kot rezultat algoritem vrne točke detektiranih prstnih konic v obliki seznama. Operacija % v algoritmu predstavlja deljenje po modulu (vrača ostanek



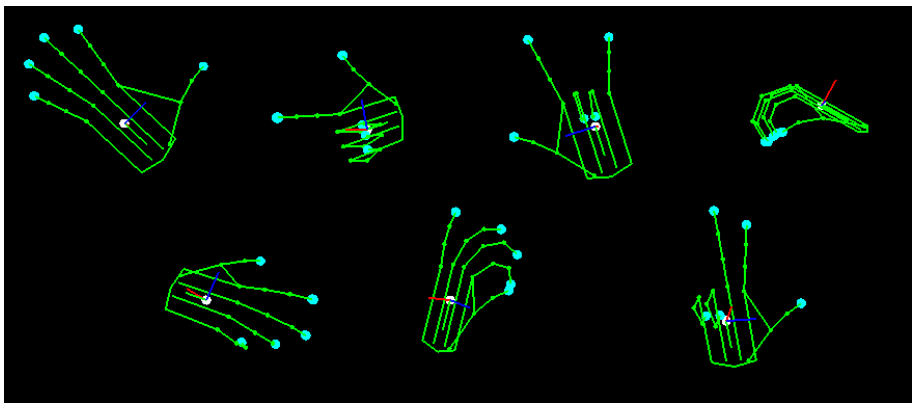
Slika 5.8 Detekcija vrzeli in prstov, ter filtriranje vrzeli s pomočjo normale.

pri deljenju) in zagotavlja, da indeksiranje seznama ne preseže njegove velikosti. Zapis $[array][i]$ predstavlja dostopanje do elementa v seznamu $array$ z indeksom i. Uporabljena operacija $size(array)$ vrne velikost seznama elementov. Funkcija $insert(array, value)$ pa predstavlja operacijo vstavljanja elementa v seznam. Podatkovna struktura *vector* podobno kot točka (*point*) predstavlja strukturo s koordinatami, vendar za razliko od točke, v trodimenzionalnem prostoru. Čeprav sliko predstavlja dvodimenzionalni prostor, je trodimenzionalnost vektorja pomembna pri izračunu normale, saj njena z koordinata predstavlja usmerjenost in s tem prej omenjen pogoj za ločevanje vrzeli od prstnih konic. Vrednost STEP predstavlja prej omenjen korak, ki določa gostoto točk, uporabljenih pri procesu iskanja.

V implementaciji je bila kot točka uporabljena podatkovna struktura ofPoint, ki je del ogrodja OpenFrameworks. Podobno je vektor predstavljen s strukturo ofVec3f, ki izvira iz istega ogrodja [11]. Seznam (*array*) je bil realiziran s standardnim gradnikom v jeziku C++ (objektom vector<>).

5.8 Uporaba sintetičnega modela roke

Sintetični model roke lahko definiramo kot povezano strukturo segmentov. Za vsak segment izberemo prostostne stopnje (angl. degrees of freedom - DOF), ki določajo osi, po katerih bo mogoča njegova rotacija. Ena prostostna stopnja tako pomeni, da se bo segment lahko rotiral le okrog ene osi (ali se premikal le vzdolž ene osi, če gre za translacijo). Primer ene prostostne stopnje je v primeru modela roke sklep, ki je najbližje konici poljubnega prsta. Dve prostostni stopnji pa pomenita, da ima segment omogočeno



Slika 5.9 Implementacija modela, prikazanega v različnih pozah.

rotacijo okrog dveh osi. Primer rotacije okrog dveh osi predstavlja korenski sklep vsakega prsta. Prostostnih stopenj je lahko tudi več. Na splošno velja, da lahko lego in orientacijo poljubnega telesa v tridimenzionalnem prostoru opišemo s 6 DOF. To pomeni, da 3 prostostne stopnje določajo translacijo v treh smereh (XYZ) ciljnega koordinatnega sistema. Preostale 3 prostostne stopnje pa določajo rotacijo okoli treh osi v omenjenem koordinatnem sistemu.

5.8.1 Definicija kinematičnega modela

Kinematični model človeške roke tako v naši implementaciji (slika 5.9) parametriziramo s 26 prostostnimi stopnjami. Razdelimo jih na 6 globalnih (ki določajo globalno lego in orientacijo) ter 20 lokalnih stopenj (ki določajo rotacije členkov za vse obstoječe prste, translacij lokalno namreč ni, saj v realnosti prstov na roki ne moremo poljubno raztegovati). Teh 20 stopenj lahko dodatno razdelimo po posameznih prstih. Pri tem ima korenski segment vsakega prsta 2 prostostni stopnji (veljavna rotacija okrog dveh osi), preostala dva segmenta pa 1 prostostno stopnjo (omogočena rotacija okrog ene osi). Možna je tudi definicija modela s 27 DOF, pri tem se upošteva, da ima palec dva segmenta s po 2 DOF (kot prikazuje tudi slika 3.3).

Če se vrnemo na problem določanja parametrov rok iz globinske slike, s katerim se prvotno ukvarjamo, lahko vidmo, da ima primerjalni model z 26 DOF zelo velik preiskovalni prostor za iskanje optimalne rešitve (podobnosti med parametri roke v realnosti in parametri kinematičnega modela). Vsaka prostostna stopnja ima lahko več vrednosti in predstavlja svojo dimenzijo v prostoru preiskovanja. Tega prostora v realnosti ni mogoče

preiskati v celoti, zato pri implementaciji zmanjšamo njegove dimenzije.

To lahko storimo na več načinov. Eden izmed njih je, da poenostavimo rotacije prstov in predpostavimo, da rotacija korenskega segmenta vpliva na vse naslednje segmente (podobno kot v [23]). Gibanje modela roke bo še vedno zadovoljivo, hkrati pa smo iz parametrov vsakega prsta odstranili 2 DOF, kar skupno pomeni $5 \cdot 2 = 10$ DOF. Poleg tega nam podatek o legi dlani posreduje že sam ofxHandGenerator, kar pomeni zmanjšanje preiskovalnega prostora za 3 dodatne prostostne stopnje (ki določajo lego dlani). Kakorkoli, preiskovalni prostor bo še vedno preveč obsežen, zato bi ga bilo potrebno še bolj zmanjšati.

Pri trenutni implementaciji se je pojavil tudi problem kompleksnosti primerjav, ki se izvedejo pri ocenjevanju podobnosti modela in parametrov v realnosti. Običajno se pri metodah ujemanja, ki temeljijo na modelu, ta podobnost računa na podlagi projekcije modela v obliki slike in globinske slike roke v realnosti ali tudi glede na najdene robove v obeh slikah. To pomeni, da se izvede primerjava po pikslih med obema slikama, kar pa predstavlja poleg obsežnosti preiskovanja v prostoru parametrov modela še dodatno računsko zahtevnost v smislu primerjav silhuet (ali robov) med modelom in roko uporabnika. S temi omejitvami je implementacija sistema, ki bi deloval z zadovoljivo odzivnostjo v realnem času, zelo otežena. Kot bomo videli v nadaljevanju ta problem v realni implementaciji rešujemo tako, da uvedemo predpostavko, da bo naš sistem prepoznaval omejeno število različnih poz, ki jih uporabnik lahko oblikuje s svojimi prsti.

5.8.2 Generiranje projekcije modela

Globinska slika se generira s pomočjo matrične transformacije (rotiranja) lokalnih koordinat ključnih točk v modelu. Tako pridobimo globalne koordinate dlani ter vseh segmentov prstov. Naslednji korak je risanje linij med temi točkami. Tem linijam se hkrati doda še informacija o globini, ki je predstavljena s sivinskimi nivoji. Globinsko informacijo pridobimo preko z koordinat dobljenih točk (glede na središče dlani). Tako bodo točke, ki so od središča dlani bližje glede na tretjo koordinato obarvane s svetlejšim sivinskim odtenkom. Podobno bodo točke, ki so oddaljene dlje imele temnejši sivinski odtenek. S tem simuliramo podobne sivinske odtenke, kot jih generira Kinect pri detekciji globine v prostoru. Linije modela se rišejo na sliko s pomočjo B-line algoritma [16] (implementacija povzeta iz [17]), ki opravi rasterizacijo točk linij v slikovne elemente na sivinski sliki. Po fazi risanja se s pomočjo knjižnice OpenCV na obstoječi sliki opravijo

Slika 5.10 Projekciji modela, pred operacijo širjenja (levo) in po njej (desno).

še morfološke operacije (širitev oz. dilatacija), ki narisanim linijam poudarijo debelino (slika 5.10).

Generiranje projekcije modela na ta način ni optimalno izvedeno, vendar bi zaradi enostavnosti modela težko izbrali boljšo metodo za določitev njegove sivinske slike. Model trenutno namreč vsebuje le osnovne točke, ki opisujejo kinematično strukturo, vendar je brez volumna, ki bi določal bolje definirano obliko dlani. Model bi bilo zato potrebno dopolniti s podrobnejšim 3D modelom, ki bi ga lahko deformirali na podlagi trenutne skeletne strukture.

V kolikor bi bil model realiziran na ta način, uporaba B-line algoritma in morfoloških operacij ne bi bila več potrebna. Projekcijo bi enostavno pridobili z risanjem modela v začasni slikovni pomnilnik (angl. frame buffered object) s pomočjo objekta ofFbo, katerega implementacijo ponuja ogrodje OpenFrameworks [11]. Omenjeni objekt ponuja neposreden dostop do slikovnih elementov, iz katerih lahko pridobimo izris modela v sliko projekcije. Pri tem pristopu je oteženo le pridobivanje informacije o globini, zato bi morali za to poskrbeti v fazi samega risanja modela.

5.8.3 Primerjanje projekcije s sliko oblaka točk

V trenutni različici sistema je primerjalni model uporabljen na zelo bazičnem nivoju. V nasprotju z omenjenimi metodami in pristopi določanja parametrov dlani z modeliranjem, tukaj ni realiziranega preiskovanja večdimenzionalnega problemskega prostora. Primerjave se preprosto izvedejo glede na število prstov, ki jih zaznamo v fazi analize robov. Tako lahko v vsakem koraku primerjamo le kombinacije, ki vsebujejo prepoznano število prstov in tako zelo zmanjšamo število primerjav. Same primerjave so izredno časovno zahtevne, saj gre pri njihovi trenutni implementaciji za primerjanje dveh sivinskih slik. To pomeni, da moramo v vsakem slikovnem elementu izračunati absolutno razliko med

soležnima elementoma na obeh slikah in dobljen rezultat akumulirati v skupen seštevek, ki nam predstavlja podobnost med generirano sliko modela in sliko roke uporabnika v realnosti. Prav zaradi tega je število primerjav omejeno, če želimo prepoznavanje v aplikaciji izvajati v realnem času. Za večje število razpoložljivih primerjav, bi bilo potrebno narediti kompromis med zahtevnostjo primerjave, ki bi podala še zadovoljivo podobnost (npr. ločljivost primerjanih slik) in številom zaželenih primerjav, ki jih lahko opravimo v enem koraku določanja parametrov dlani. Alternativna možnost omenjenemu pristopu je paralelizacija izvajanja primerjav. Pri tem bi potrebovali zmogljiv grafični procesor, kar pa ne predstavlja glavnega problema. Večji problem predstavlja prenosljivost aplikacije, saj paralelizacije na vseh računalniških sistemih ne moremo enolično in zagotovo realizirati. Sem spada tudi dodatna kompleksnost razvoja programske logike za izvajanje paralelizacije in logike za dodeljevanje opravil večim procesorjem na grafičnem čipu in zlivanje dobljenih rezultatov po končanem vzporednem računanju primerjanja. Obstaja še možnost primerjanja značilk in korespondenčnih točk na obeh slikah. Pri tem pristopu bi morali implementirati pridobivanje pomembnih (informativnih) točk na obeh slikah, ki bi imele dovolj veliko korelacijo, da bi z njo lahko zadovoljivo določili ujemanje in s tem podobnost obeh slik. Pri tem pristopu bi si lahko pomagali z obstoječimi metodami iz področja računalniškega vida in obdelave slik (npr. iskanje kotov na obeh slikah na podlagi spreminjanja gradientov po obeh dimenzijah slik). Velja pa opozoriti, da je tudi pri tem načinu potreben vsaj en prehod čez obe sliki, kar posledično pomeni, da časovna kompleksnost primerjave ni manjša kot pri trenutni implementaciji primerjanja. Povsem analitično lahko časovno kompleksnost določimo, če upoštevamo število slikovnih elementov, ki so uporabljeni pri primerjanju. Iz implementacije lahko privzamemo, da je resolucija primerjanih slik enaka v obeh dimenzijah (d). Če upoštevamo, da primerjavo izvajamo nad dvema slikama (faktor 2), lahko časovno zahtevnost zapišemo kot

$$O(2 \cdot d \cdot d) = 2 \cdot O(d^2) = O(d^2),$$

kar pomeni, da gre za zahtevnost, ki narašča s kvadratom resolucije primerjanih slik (kvadratna časovna zahtevnost).

Če se vrnemo na določanje položaja prstov in vzamemo za primer, da v fazi določanja ekstremov na robovih ugotovimo, da ima uporabnik izravnan en prst, vsi ostali pa so sklenjeni, potem sedaj vemo, da se rešitev problema skriva le v 5 permutacijah razporejenosti prstnih konic. Če je iztegnjen en prst to pomeni, da se lahko nahaja na 5

različnih lokacijah. Na ta način lahko zavržemo ostalih $2^5 - 5$ $(32 - 5 = 27)$ permutacij, kar je velika prednost. Slaba stran metode je seveda ta, da zveznosti in s tem vmesnih stanj med iztegnjenimi in sklenjenimi prsti ni. Na ta način lahko določimo le kateri prsti so v danem trenutku izravnani ali ne, poleg tega pa lahko zaradi šuma v podatkih iz globinskega senzorja pride tudi do napačnih rešitev, ki zmanjšajo robustnost prepoznavanja. Velik vpliv na pravilno določitev prstov ima tudi poravnanost slike v realnosti s projekcijo sintetičnega modela. Tako v določenih primerih dodatno pridejo do izraza nepravilno prepoznane rešitve, predvsem zato ker orientacije dlani ne moremo točno določiti in jo lahko le ocenimo iz oblaka točk. Hkrati zaradi dveh prostostnih stopenj korenskih segmentov v vsakem prstu lahko pride tudi do rotacije v smereh sosednjih prstov, kar lahko povzroči, da aplikacija prav tako napačno napove kombinacijo z napako enega prsta. Napačne napovedi takega tipa bi lahko reševali z zaklepanjem stanj. To pomeni, da bi onemogočili prehajanje med zelo podobnimi stanji, pri katerih lahko prihaja do napak.

Prednosti tovrstnega določanja kombinacij prstov je več. Kot prvo velja omeniti njeno enostavnost. Permutacija in s tem položaj petih prstov se zaradi preproste zasnove lahko določi le s 5 biti. To pomeni, da v praksi lahko vsako kombinacijo predstavimo s pozitivnim celim številom med 0 in 31. Tako na primer število 0 predstavlja zaprto pest, saj so vsi biti s katerimi je število predstavljeno enaki 0. Če kot primer vzamemo število 17 (katerega binarna vrednost je enaka zapisu 10001) ta predstavlja iztegnjenost mezinca in palca ter sklenjenost ostalih prstov. Taka definicija stanj skupaj sestavlja 32 različnih kombinacij postavitve prstov. Iz tega prostora lahko dodatno izločimo še nekaj kombinacij, saj se izkaže, da je določene poze težje oblikovati. Te določene kombinacije izločimo povsem brez zadržkov, saj od uporabnika aplikacije ne pričakujemo, da bo svoje geste izvajal v oblikah, ki so zanj težko izvedljive. S tem prostor kombinacij z opravljenim testiranjem zmanjšamo na okvirnih 20 kombinacij. Kombinacije lahko kot razvijalci poljubno prilagajamo glede na potrebe aplikacije. Izločene kombinacije še vedno lahko vključimo nazaj ali pa prostor še dodatno zmanjšamo in tako ohranimo ključen nabor kombinacij in obenem izboljšamo odzivnost aplikacije.

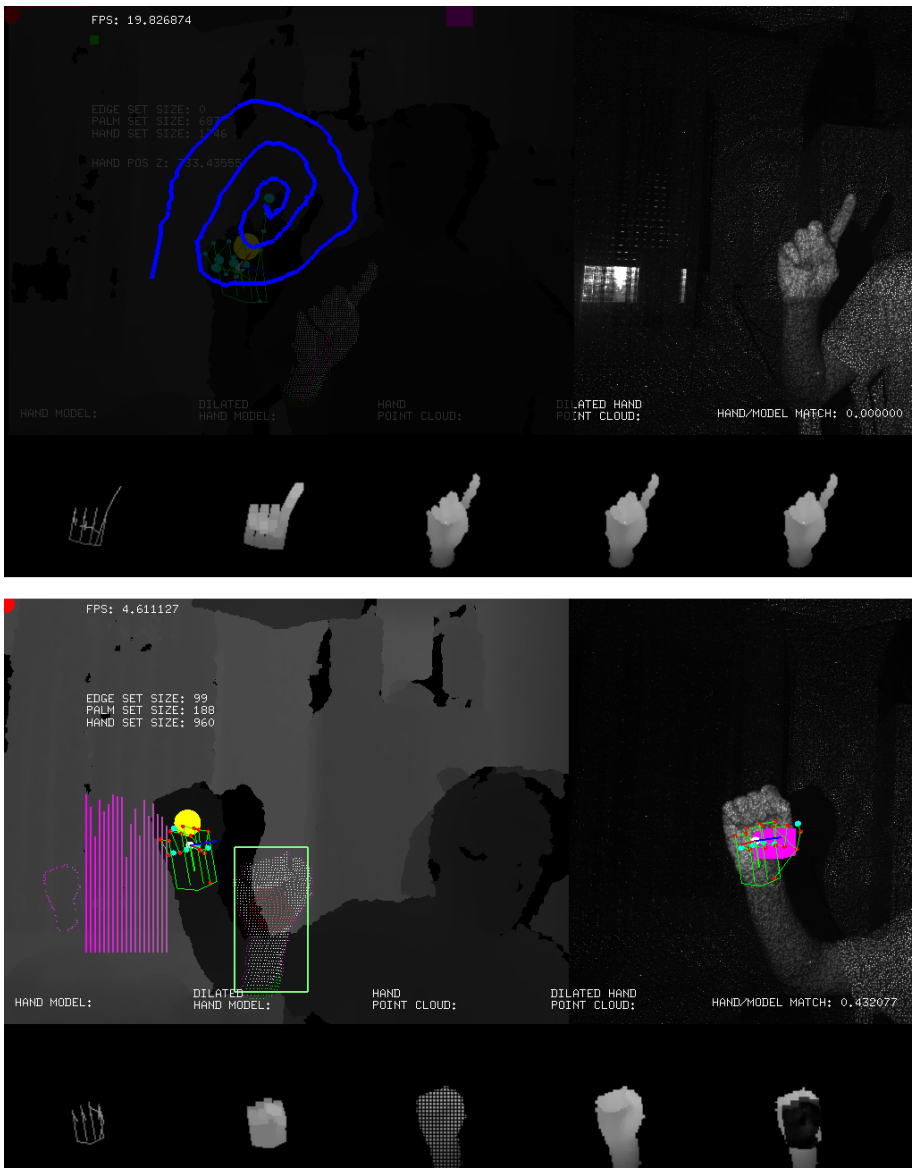
Posebnost takega načina prepoznavanja je tudi v tem, da pri tem ni uporabljenih tehnik strojnega učenja. Zaradi tega se kažejo tako slabosti kot tudi prednosti. Pod prednosti lahko uvrstimo to, da za delovanje ne potrebujemo niti dodatnega časa niti (velike) množice primerov za učenje algoritma. Slaba stran je seveda ta, da se algoritem

med svojim delovanjem ne razvija in tako ne nadgrajuje oz. izboljšuje svojega odločanja glede na odločitve iz preteklosti.

Logika določanja kombinacij je, kot rečeno v nadaljevanju določena s številom prstov, ki jih dobimo pri analizi robov. Če je ugotovljeno število konic med 1 in 4, potem se za dobljeno število opravi 5 izbranih primerjav, ki določijo končno kombinacijo. Ta kombinacija se izbere glede na največje ujemanje med vsemi izvedenimi primerjavami in se v končnem koraku aplicira tudi na model, ki nam klasifikacijo vizualizira. V primeru, da analiza robov vrne 5 točk (kar pomeni, da je bilo zaznanih vseh 5 prstov uporabnika) se primerjave ne izvedejo, saj v prostoru kombinacij le ena ustreza pogoju povsem odprte dlani. V primeru, da analiza robov pride do zaključka, da ne obstaja nobene točke, ki bi predstavljala konico prsta, to lahko pomeni, da gre za stisnjeno pest uporabnika, obstaja pa tudi možnost, da je uporabnik poravnal dlan tako, da so vrzeli med prsti izginile. Prav zaradi tega je potrebno v tem primeru izvesti dodatni dve primerjavi, ki odločita ali gre za gesto, ki predstavlja pest ali pa mogoče uporabnik želi ponazoriti dlan brez vrzeli. Dobra plat tega robnega primera je ta, da so razlike v omenjenih pozah zelo velike, posledično pa izvedeni primerjavi vedno pravilno napovesta lego dlani in iztegnjenost ali sklenjenost prstov.

5.9 Ovrednotenje in uporaba rezultatov

Kot primer praktične uporabe sta bili razviti dve dodatni demonstracijski komponenti, ki prikazujeta interakcijo s pomočjo detekcije dlani, ki jo izvaja implementiran sistem. Na podlagi pozicije dlani, števila iztegnjenih prstov in drugih pridobljenih podatkov je bilo omogočeno risanje po zaslonu ter premikanje in manipulacija z objekti (slika 5.11). Praktični komponenti sta sicer eksperimentalne narave in imata še veliko pomankljivosti pri prepoznavanju ter interpretaciji podatkov pridobljenih iz globinskega senzorja. Za dejansko uporabo detekcije dlani in prstov v realnih aplikacijah bo tako potrebna dodatna optimizacija na področju izvajanja sistema in njegove celotne strukture.



Slika 5.11 Interaktivni komponenti, ki prikazujeta uporabo sistema za detekcijo dlani in položaja prstov (zgoraj risanje po površini, spodaj manipulacija z objektom v obliki kvadrata).

6 Možne izboljšave in nadalnje delo

Zastavljena implementacija ponuja veliko možnosti za izboljšave. Zaželjena je posodobitev modela, ki je bil uporabljen za primerjanje dlani in vizualizacijo. Skupaj z modelom lahko optimiziramo tudi metode za generiranje njegove projekcije.

Hkrati je veliko možnih izboljšav na področju logike odločanja in ocenjevanja parametrov. Tukaj lahko omenimo optimizacijo in zmanjšanje zahtevnosti izvajanja samih primerjav med modelom in dlanjo uporabnika. K boljši uporabi aplikacije bi pripomogla tudi izboljšava ocenjevanja orientacije dlani. Poleg tega v določenih segmentih kode primanjkuje strukturiranosti in arhitekturno dobrih rešitev. S tem se odpre možnost izboljšav v smislu dodajanja modularnosti komponentam, ki so nastale v sklopu raziskovalnega razvoja.

Določen problem predstavlja tudi šum v samih vhodnih podatkih ter relativno nizka resolucija globinskega senzorja. Zaradi tega je pravilna napoved parametrov dlani na večjih razdaljah od senzorja zelo otežena. Problem lahko predstavlja tudi hitro gibanje rok, zaradi česar je pridobljena globinska slika nenatančna (predvsem na področjih prstov).

Sistem trenutno prepoznava omejen nabor različno oblikovanih dlani in prstov. V načrtu za prihodne izboljšave je tudi posodobitev tega nabora in vključitev zveznejših prehodov med stanji segmentov v kinematičnem modelu. Namesto binarnega sistema za določanje položaja prstov, bi lahko na primer uporabili trojiški sistem. Trojiški zapis v permutacijah bi omogočil prepoznavanje tudi polovično odprtih prstov, saj bi z njim (namesto dveh obstoječih stanj) lahko definirali tri možna stanja, v katerih se prsti lahko nahajajo. Hkrati bo seveda potrebna izboljšava prepoznavanja nekaterih obstoječih oblik dlani, saj v določenih primerih sistem teh ne prepozna pravilno.

Prav tako bo v sistemu potrebno izboljšati podporo za zajem dlani različnih velikosti. Trenutno je namreč potrebna ponovna kalibracija sistema za vsakega novega uporabnika. Za dinamično kalibracijo bi lahko uporabili znana razmerja človeškega telesa. Lahko bi na primer upoštevali korelacijo med velikostjo dlani in površino obraza uporabnika. Poleg tega bi lahko upoštevali še druga razmerja (npr. višina glave predstavlja približno 1/7 višine celotnega telesa). S pomočjo znanih razmerij in nekaj dodatne logike bi lahko omogočili dinamično kalibracijo, s tem pa bi se dodatno izboljšala uporabniška izkušnja. Hkrati bi na podoben način lahko določili tudi dolžine prstov. S pomočjo teh podatkov bi prilagodili uporabljen sintetični model. Tako bi dosegli še boljše rezultate pri iskanju optimalnega ujemanja.

Implementacija sistema bo, kot rečeno, uporabljena v programski opremi podjetja Uniki d.o.o. V ta namen bo potrebna izvedba dodatne integracije razvitih komponent v omenjen pogon. Omogočiti bo potrebno ustrezno komunikacijo z obstoječimi komponentami sistema in zagotoviti združljivost z obstoječimi izdelki ter aplikacijami v razvoju. Poskrbeti po potrebno tudi za ustrezno dokumentacijo vključenega podsistema, v kolikor bo ta zaradi modularnih izboljšav in dopolnitev postal bolj razdrobljen in kompleksen. S pomočjo dokumentacije bo razvoj aplikacij z detekcijo dlani olajšan tudi bodočim razvijalcem.

7 Zaključek

Predstavljeno področje prepoznavanja rok uporablja in povzema metode iz strojnega vida (obdelava barvnih in globinskih slik, njihova segmentacija in iskanje vizualnih značilnosti), računalniške grafike (modeliranje pomožnih struktur) ter umetne inteligence (klasifikacija primerov in strojno učenje). Trenutno na področju še ni pristopa, ki bi le na podlagi vizualnih informacij ter brez uporabe drage opreme v realnem času ponujal popolno rekonstrukcijo dlani in prstov. Specifični sistemi se sicer temu zelo približajo in dobro delujejo v nadzorovanih razmerah, ki so omejene z množico predpostavk. V prihodnosti zato lahko pričakujemo še večji porast raziskav na tem področju, kot tudi prihod zmogljivejših senzorjev in naprav, ki bodo omogočale podrobnejše in natančnejše zaznavanje.

Literatura

[1] Leap Motion Overview, (5. 9. 2013), https://developer.leapmotion.com/documentation/Languages/C++/Guides/Leap_Overview.html#hand-model.

[2] Videoplace - Wikipedia, (12. 8. 2013), http://en.wikipedia.org/wiki/Videoplace.

[3] Youtube - Videoplace (1985) [VintageCG, 2009], (12. 8. 2013), http://www.youtube.com/watch?v=d4DUIeXSEpk.

[4] Data Gloves, (17. 8. 2013), http://www.5dt.com/?page_id=34.

[5] Hand - Wikipedia, (17. 8. 2013), http://en.wikipedia.org/wiki/Hand.

[6] Morphology - Distance Transform, (4. 9. 2013), http://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm.

[7] About OpenNI, (22. 7. 2013), http://www.openni.org/about/.

[8] Organization (The people behind OpenNI), (22. 7. 2013), http://www.openni.org/organization/.

[9] OpenNI Migration Guide, (22. 7. 2013), http://www.openni.org/openni-migration-guide/.

[10] PrimeSense^{TM} NITE Algorithms 1.5, (23. 7. 2013), http://www.openni.org/wp-content/uploads/2013/02/NITE-Algorithms.pdf.

[11] OpenFrameworks documentation reference, (10. 8. 2013), http://www.openframeworks.cc/documentation/.

[12] About OpenFrameworks, (8. 8. 2013), http://www.openframeworks.cc/about/.

[13] ofxOpenNI - Wrapper for OpenNI, NITE and SensorKinect, (23. 7. 2013), https://github.com/gameoverhack/ofxOpenNI/.

[14] Center of the Palm (Hand Tracking), (12. 5. 2013), http://blog.candescent.ch/2011/04/center-of-palm-hand-tracking.html.

[15] Cross product - Wikipedia, (16. 6. 2013), http://en.wikipedia.org/wiki/Cross_product.

[16] Bresenham's line algorithm, (2. 5. 2013), http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm.

[17] Bitmap/Bresenham's line algorithm, (2. 5. 2013), http://rosettacode.org/wiki/Bitmap/Bresenham's_line_algorithm.

[18] G. Bradski, A. Kaehler, Learning OpenCV, O'Reilly Media, 2008.

[19] M. Bray, E. Koller-Meier, L. Van Gool, Smart particle filtering for 3D hand tracking, in: Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings., 2004, pp. 675–680.

[20] A. Castro, How OpenFrameworks works, (9. 8. 2013), http://www.openframeworks.cc/tutorials/developers/001_how_openFrameworks_works.html.

[21] R. Cippola, B. Stegner, A. Thayananthan, P. H. S. Torr, Hand tracking using a quadric surface model, in: Proc. 10th IMA Conference on The Mathematics of Surfaces, Leeds, UK, 2003, pp. 129–141.

[22] J. Crouse, ofTutorials - Introduction, (8. 8. 2013), http://www.openframeworks.cc/tutorials/introduction/000_introduction.html.

[23] J. Cui, S. Zengqi, Model-based visual hand posture tracking for guiding a dexterous robotic hand, Optics Communications 235 (4-6) (2004) 311.

[24] R. Dunlop, Understanding the Dot Product, (16. 6. 2013), http://www.mvps.org/directx/articles/math/dot/.

[25] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, X. Twombly, A review on vision-based full DOF hand motion estimation, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops (2005) 75.

[26] G. Geebelen, C. Tom, S. Maeseny, P. Bekaerty, Real-time hand tracking with a colored glove, in: Conf. Proc. 3D Stereo Media.

[27] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with Microsoft Kinect sensor: A review, IEEE Transactions on Cybernetics PP (99) (2013) 1.

[28] M. M. Hasan, P. K. Mishra, A review on vision-based full DOF hand motion estimation, Canadian Journal on Image Processing and Computer Vision, (2012) 3 (1).

[29] T. Heap, D. Hogg, Towards 3D hand tracking using a deformable model, in: Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, Killington, VT, 1996, pp. 140–145.

[30] N. K. Iason Oikonomidis, A. Argyros, Efficient model-based 3D tracking of hand articulations using Kinect, in: Proc. BMVC, 2011, pp. 101.1–101.11.

[31] N. A. Ibraheem, R. Z. Khan, Vision based gesture recognition using neural networks approaches: A review, International Journal of human Computer Interaction (IJHCI) 3 (1) (2012) 1–14.

[32] C. Keskin, F. Kırac, Y. E. Kara, L. Akarun, Real time hand pose estimation using depth sensors, in: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 2011, pp. 1228–1234.

[33] D. Lee, S. Lee, Vision-based finger action recognition by angle detection and contour analysis, ETRI Journal 33 (3) (2011) 415–422.

[34] O. Lopes, M. Pousa, M. Reyes, S. Escalera, J. Gonzalez, Multi hand pose recognition system using Kinect depth sensor, in: Kinect Demonstrator Challenge, International Conference in Pattern Recognition, Tsukuba, Japan, 2012.

[35] S. Malik, Real-time hand tracking and finger tracking for interaction. csc2503f project report., Tech. rep., University of Toronto, Department of Computer Science (December 2003).

[36] G. Simion, G. Vasile, M. Ote'teanu, A brief review of vision based hand gesture recognition, in: Proceedings of the CSECS '11, MECHANICS '11 and ICAT '11, Montreux, Switzerland, 2011, p. 181.

[37] D. J. Sturman, D. Zeltzer, A survey of glove-based input, Computer Graphics and Applications, IEEE 14 (1) (1994) 30–39.

[38] R. Y. Wang, J. Popović, Real-time hand-tracking with a color glove, ACM Transactions on Graphics 28 (3) (2009) 1.

[39] T. Weise, Hand tracking for virtual object manipulation, MSc thesis, IMPERIAL COLLEGE LONDON, Department of Computing (June 2005).

[40] Y. Wu, T. S. Huang, Capturing articulated human hand motion: A divide-and-conquer approach, in: Proc. 7th Int. Conf. on Computer Vision, Vol. 1, Corfu, Greece, 1999, pp. 606–611.