Obogatena resničnost za interaktivno analizo omrežij

Matic Potočnik, Marko Tatić, Pavel Stare, Mark Rolih, Oton Žlindra

Povzetek—V okviru predmeta Interaktivnost in Oblikovanje Informacij smo z uporabo senzorja Microsoft Kinect, ter 3D očal Vuzix Wrap 920 AR, izdelali preprost prototip sistema za vizualizacijo grafov v obogateni resničnosti.

Ključne besede—obogatena resničnost, analiza omrežij, naravni uporabniški vmesniki

1 Uvod

OBOGATENA resničnost (angl. *augmented reality*) ter t.i. naravni uporabniški vmesniki (angl. *natural user interface – NUI*) sta v zadnjem času vroči temi v akademskih, umetniških in tudi ljubiteljskih krogih. Gre za tehnologije, ki jih znanstvena fantastika obljublja že desetletja, sedaj pa končno postajajo dostopne vedno širši javnosti. Celo velika podjetja kot sta Google in Microsoft, so vstopila na ta trg in trenutno kaže da se bo razvoj uporabniških vmesnikov premikal v tej smeri.

Še ena tema, ki uživa precej pozornosti pa so velike zbirke podatkov (angl. *big data*), še posebej velika omrežja, kot so npr. družbena omrežja na spletu, gensko-regulatorna omrežja v biologiji, itd. Tu gre za razvoj postopkov za obvladovanje velikih količin podatkov in prikaz teh podatkov v oblikah, ki lastniku podatkov povedo kar največ.

Iz teh razlogov smo se odločili za raziskovalno seminarsko nalogo, ki združuje naštete teme – poskušali smo razviti naravni uporabniški vmesnik za učinkovito analizo velikih omrežij v obogateni resničnosti.

V okviru naloge smo razvili le preprost prototip, smo pa raziskali nekaj rešitev s katerimi bi se lahko bolj približali zadanemu cilju.

10. junij 2012

2 Opis strojne opreme

V tem odseku bomo na kratko predstavili strojno opremo, ki smo jo uporabili pri razvoju.

2.1 Microsoft Kinect

Microsoft Kinect je naprava, ki v sebi združuje več senzorjev:

- štiri mikrofone
- globinsko kamero
- kamero

Slika 1. Senzor Microsoft Kinect

Kinect je prišel na trg novembra 2010, sprva za igralne konzole Xbox 360, kasneje pa tudi za Microsoft Windows. Le nekaj dni po izidu pa so se vmešali tudi ljubitelji in z vzvratnim inženiringom (angl. *reverse engineering*) dosegli, da so lahko napravo prek vodila USB uporabili za svoje namene. Poleg uradnega Microsoftovega paketa za razvoj aplikacij za Kinect, tako obstaja še nekaj drugih ogrodij in knjižnic, npr. Freenect skupine OpenKinect, ter OpenNI/NITE podjetja Primesense, ki je tudi razvilo senzorje v Kinect-u.

2.2 Vuzix Wrap 920AR

Vuzix je eden izmed vodilnih svetovnih proizvajalcev 3D očal in majhnih zaslonov.

Slika 2. 3D očala Vuzix Wrap 920 AR

Njihov produkt Wrap 920 AR je namenjen obogateni resničnosti in kot tak že vsebuje vse potrebno:

- dva majhna zaslona
- kameri na sprednji strani
- senzorje za nagib glave:
  - magnetometere (angl. *magnetometers*)
  - pospeškometere (angl. *accelerometers*)
  - žiroskope (angl. *gyroscopes*)

Slika 3. Senzorji Vuzix Wrap Tracker 6TC

3 PROTOTIP

V tem odseku bomo opisali uporabljene programske rešitve, ter postopek in rezultat razvoja prototipa.

3.1 Obseg prototipa

Po premisleku smo se odločili, da bo prototip omogočal le prikaz manjšega omrežja v prostoru okrog uporabnika, ter premikanje vozlišč omrežja z uporabo klikanja miške, ter zaznavanjem položaja desne roke z uporabo Kinect-a.

Prav tako pa smo se odločili, da bomo namesto razvoja novega sistema, za svoje potrebe raje priredili in uporabili izvorno kodo umetniškega projekta LUMEN, na katerem je delal eden od članov ekipe (Matic Potočnik), skupaj z umetnikom Dominikom Mahničem.

Na začetku načrtovanja smo sicer nameravali razviti popolnoma nov sistem na operacijskem sistemu Windows v programskem jeziku C#, z uporabo ogrodja OpenNI/NITE za delo s Kinect-om, ogrodjem Microsoft XNA za izris grafike, uradnimi Vuzix-ovimi API-ji za delo s senzorji na očalih, ter še z uporabo dodatnih knjižnic za zaznavanje vizualnih markerjev in gest rok, ampak se je kmalu izkazalo, da imamo na voljo premalo časa za razvoj tako obsežnega prototipa.

3.2 Razvoj prototipa

3.2.1 LUMEN

Aplikacija LUMEN, ki smo se jo namenili prirediti, je bila napisana v jeziku C++ za operacijski sistem GNU/Linux, uporabljene pa so bile naslednje knjižnice in ogrodja:

- OpenNI/NITE za delo s Kinect-om
- OpenGL za izris grafike
- OpenCV za zajem slike iz kamer
- lastna rešitev za branje podatkov z očal
- DCM algoritem[1] za interpretacijo podatkov z očal

Ta aplikacija je bila sicer namenjena risanju v obogateni resničnosti, a nismo imeli večjih težav s tem, da smo iz nje izluščili tiste dele, ki so nam prišli prav pri razvoju našega prototipa.

3.2.2 Generiranje omrežja

Omrežje, ki smo ga uporabili v prototipu se generira povsem naključno in se umesti v prostor, ko Kinect zazna uporabnika sistema. Za umestitev omrežja v prostor smo uporabili preprost algoritem, ki pa za večja omrežja ne bi bil primeren, saj pri umeščanju sploh ne upošteva povezav v omrežju in ker je vsaj v osnovni implementaciji precej časovno potraten, prav tako pa zahteva vnaprej znano velikost prostora, ki mora vsaj približno ustrezati številu vozlišč. Algoritem deluje tako, da umešča vozlišča v prostor enega po enega in sicer tako, da 100-krat izbere naključne koordinate v prostoru, nato pa uporabi tisto, ki je kar najbolj stran od najbližjega vozlišča.

3.2.3 Izris omrežja

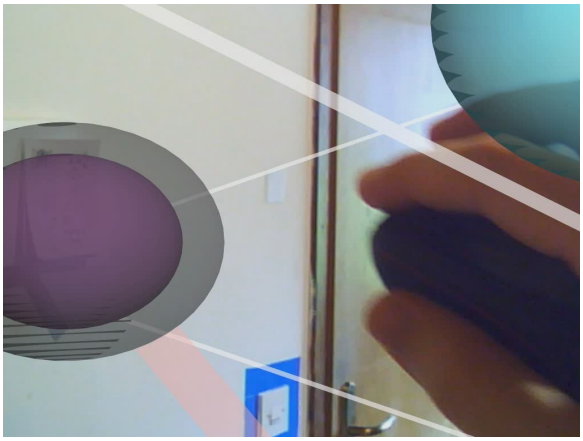
Tu ni bilo posebnosti. Omenili bi morda le, da smo naknadno vsakemu vozlišču določili še unikatno barvo, saj smo ugotovili, da se je tako veliko lažje orientirati v prostoru. Poleg tega smo vozlišča naredili rahlo prosojna, saj je to izboljšalo efekt obogatene resničnosti.

Slika 4. Zaslonska slika prototipa sistema

3.2.4 Interakcija

Kot že omenjeno smo interakcijo v prototipu omejili le na premikanje vozlišč v omrežju.

Uporabniški vmesnik deluje tako, da premikamo roko po prostoru in ko smo dovolj blizu vozlišču, se okrog njega pojavi dodatna prosojna krogla, ki pove da to vozlišče lahko izberemo. S klikom miške, lahko vozlišče primemo in ga premaknemo (angl. *drag and drop*) na druge koordinate v prostoru.



Slika 5. Zaslonska slika prototipa sistema

Ker je včasih prišlo do odstopanj med tem, kar se je uporabnik videl skozi očala in tem, kar je zaznaval Kinect, smo se odločili dodati še prosojno črto, ki ponazarja uporabnikovo roko, kot je razvidno na sliki 5.

4 NADALJNJE DELO

V tem odseku bomo opisali nekaj rešitev, s katerimi bi lahko naš sistem nadgradili in izboljšali.

4.1 Obogatena resničnost in interakcija

Za samo manipulacijo oziroma interakcijo uporabnika z sistemom se trenutno uporablja miška, kar ni v skladu z odločitvijo o razvoju naravnega uporabniškega vmesnika. Z algoritmi računalniškega vida (angl. *computer vision – CV*), bi lahko prepoznavali geste rok, ali pa sledili določenim točkam ali predmetom v prostoru.

4.1.1 Prepoznavanje gest

Tu velja omeniti zelo pogosto uporabljeno knjižnico OpenCV (v prototipu jo že uporabljamo, a zaeenkrat le za zajemanje slike iz kamer). Podporo za prepoznavanje gest ima tudi knjižnica Primesense NITE, ki prepoznava enostavne geste in pri tem uporablja Kinectovo globinsko kamero.

4.1.2 Vizualni markerji

Z uporabo markerjev, bi lahko npr. izboljšali natančnost zaznavanja. Tu bi omenili ARToolKit, ki je za nekomercialno uporabo odprtokoden in zmore z uporabo le ene kamere umestiti predmete v sliko na položaj vizualnega markerja.

Slika 6. Vizualni marker

Z uporabo markerjev pa bi lahko tudi avtomatizirali kalibracijo sistema, ki trenutno ni avtomatska – vedno ko sistem prestavimo, je

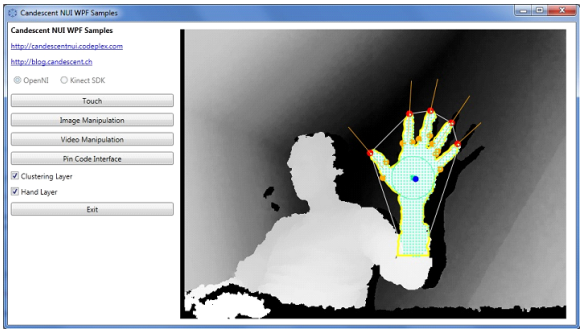
potrebno ročno uskladiti prostor Kinect-a (ki kaže ven iz Kinecta) in 3D očal (ki zaradi magnetometrov kaže proti magnetnemu severu).

Če bi namestili marker na Kinect, bi lahko uporabnik opravil kalibracijo preprosto z enim pogledom proti Kinect-u. Markerje pa bi lahko uporabili tudi za boljše sledenje rokam, ali celo kot del uporabniškega vmesnika.

Markerji so dobro podprti v večini programov za delo z obogateno resničnostjo in enostavni za uporabo, vendar pa morda niso primerni za npr. sledenje prstom, saj bi moral imeti vsak prst marker, kar pomeni dodatno pripravo uporabnika in odmik od naravnih vmesnikov, prav tako pa obstaja problem z zakrivanjem markerjev.

4.1.3 Zaznavanje prstov

Alternativa markerjem so algoritmi za zaznavanje prstov, posredno tudi zaznavanje gest. To lahko dosežemo z uporabo Kinect-a in dodatne programske opreme. Eden izmed programov ki to omogočajo, je Candescent NUI. S tem načinom interakcije bi se najbolj približali naravni komunikaciji uporabnika s sistemom. Težavo predstavlja jezik v katerem je aplikacija spisana, to je C#, vendar pa je možno z uporabo C++ME neposredno komunicirati z aplikacijami napisanimi v C++, verjetno pa bi šlo tudi na GNU/Linux-u s pomočjo ogrodja mono.



Slika 7. Program Candescent NUI

4.2 Omrežja

Tudi za delo z omrežji obstajajo namenske knjižnice. Ogledali smo si Gephi in Tulip, ki oba podpirata veliko različnih načinov vizualizacije grafov in uspešno delujeta tudi z zelo velikimi omrežji.

Knjižnici imata v splošnem podobni funkcionalnosti in podpirata razmeroma raznoliko paleto formatov zapisa grafov. Poleg samostojnih uporabniških vmesnikov imata tudi orodjarni, ki bi ju lahko uporabili za naš namen.

Na ta način bi uporabniku omogočili uvoz že obstoječih grafov in poenostavili tako izris, kot tudi manipulacijo grafov v naši aplikaciji.

4.2.1 Gephi

Je bil naš prvi kandidat za avtomatizacijo izrisovanja in prilagajanja grafov v realnem času. Napisan je v Javi, njegov grafični vmesnik pa nam omogoča precej raznovrstno prilagajanje vizualiziranja samega grafa. Z njim lahko poženemo algoritme, ki vozlišča grafa samodejno razporedijo po prostoru, jim prilagajajo velikost ali barvo izrisa glede na njihovo vrednost, pa tudi analizirajo omrežje in odkrijejo območja, ki so med sabo močneje povezana ter jih barvno ločijo od ostalih.

Vsebuje tudi orodjarno, ki je razmeroma dobro dokumentirana ter široko uporabna, zato bi jo po vsej verjetnosti v nadaljnji izvedbi projekta tudi izbrali, če ne bi bila napisana v Javi in posledično težko združljiva z našo obstoječo osnovo v jeziku C++.

4.2.2 Tulip

Je za razliko od Gephi-ja razvit v jeziku C++, sicer pa prav tako podpira široko paleto funkcij za manipulacijo grafov. Enako kot Gephi vključuje poleg grafičnega vmesnika tudi orodjarno, ki pa je v tem primeru bolj neposredno dostopna, saj je sestavljena iz .dll knjižnic, ki bi jih lahko v našem C++ programu uporabili brez posebnih prilagoditev in težav z združljivostjo. Vsebuje tudi funkcije, ki z uporabo OpenGL grafičnega pogona graf izrišejo na zaslon.

Grafe lahko naložimo iz datoteke (formatov podpira sicer nekoliko manj, obstajajo pa dodatki, ki to pomanjkljivost odpravijo), lahko pa jih tudi gradimo ali spreminjamo popolnoma programsko. Predvsem zaradi enostavnejše uporabe in boljše združljivosti, se nam zdi za nadaljnje delo Tulip boljša rešitev.

LITERATURA

[1] W. Premerlani and P. Bizard, *Direction Cosine Matrix IMU: Theory*, Draft. 2009.