Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Diplomatervezés 1
(BMEVIIIM901)

3D felhasználói felület kialakítása Kinect-tel

Témavezető: Dr. Vajda Ferenc

2013. 12. 12.

Tartalomjegyzék

1. Bevezetés

Ergonómusok, kutatók egyre inkább hangoztatják, hogy a hagyományos felhasználói felület eszközkészlete (billentyűzet, egér) nagyon rossz hatékonysággal alkalmasak a feladatok kiszolgálására. Az alternatív felhasználói felületek kutatása a processzorok és párhuzamos műveletek elvégzésére alkalmas eszközök számítási teljesítményének növekedésével egyre fontosabb szerepet kap. Az Irányítástechnika és Informatika Tanszéken működő 3D Érzékelés és Mobilrobotika Kutatócsoport e területet fontos irányvonalnak tartja, amelynek fontos eleme a megfelelő beviteli és kiviteli eszközök elkészítése és integrálása, illetve a megfelelő algoritmusok implementálása. Önálló labor és diplomamunka során a kutatócsoportban kialakítás alatt lévő háromdimenziós munkakörnyezethez olyan beviteli eszközt készítek, amely alkalmas a felhasználó nagyobb mozdulatainak (karmozdulatok, két kezes műveletek) kiértékelésére. A beviteli eszköztől, és a működését segítő szoftvertől fontos elvárás, hogy teljesítse az úgynevezett NUI (Natural User Interface) által támasztott követelményeket.

Önálló labor 1. tárgy során megvizsgáltam a rendelkezésre álló bemeneti eszközöket, majd az eredmények mérlegelése után konzulensemmel közösen a Kinect (for Xbox) mellett döntöttünk. A félév során elkészítettem továbbá néhány példaalkalmazást, valamint megismertem az OpenNI és a NITE programozási környezet használatát Linux alatt.

Önálló labor 2. tárgy során további alkalmazások készítésével próbáltam mélyíteni a tudásomat az eszköz programozását illetően, valamint alkalmazásokat készítettem az eszközben rejlő összes szenzor lehetőségeinek és pontosságának tesztelésére.

Diplomatervezés 1 tárgy során a témavezetőmmel egyeztetve - a kutatócsoportban jelen lévő eszközökkel való minél nagyobb összeegyeztethetőség érdekében - az eddig elkészített alkalmazásaimat portoltam Linuxról Windowsra, valamint OpenNI1.5 rendszerről OpenNI2.0-ra, amely magával vonta az alacsonyszintű driver cseréjét is a Microsoft által készített SDK-ra. Ezen kívül részletesen tanulmányoztam a PCL (Point Cloud Library) függvénykönyvtár használatát, valamint méréseket és becsléseket végeztem egy új alapokon nyugvó jelszókezelő alkalmazás elkészítésére.

2. A hardver környezet ismertetése

2.1. NUI: Natural User Interfaces

A Natural User Interfaces (gyakorta Natural Interfaces néven találkozhatunk vele) egy olyan ember-gép interakciót ír le, amelyben a gép vezérléséhez vagy irányításához a felhasználó külső eszköz segítségét csak közvetetten veszi igénybe, azaz a felhasználó csak a saját adottságaival visz át információt a gép felé. Ezek a saját adottságok legtöbbször mozdulatok, kézjelzések, valamint szóban kiadott utasítások. Ezáltal feleslegessé válnak az olyan jól megszokott beviteli eszközök, mint a billentyűzet és az egér.

Néhány példa a Natural User Interfaces mindennapi használatából:

- Hangfelismerés

- Kézmozdulat felismerés: néhány előre definiált kézmozdulattal (gesztussal) vezérelhetjük a rendszert.
- Mozgás felismerés: a szenzor az egész testet figyeli, a végtagok mozgatásával vezérelhetjük a rendszert. Jelenleg legtöbbször játékok irányítására használják.
- Arcfelismerés: az arcról leolvasott információk bevitele és feldolgozása.

2.2. A Natural User Interfaces-t támogató szenzorok

A négy legelterjedtebb NUI-t támogató szenzor a Microsoft által kiadott (a PrimeSense cég fejlesztésében készült) Kinect, az Asus által forgalmazott „Xtion" és „Xtion PRO", a PrimeSense cég által készítetett Carmine 1.082 és 1.09, valamint az egyre népszerűbbé váló Leap Motion. Ezekben közös, hogy USB interfészen keresztül kommunikálnak a számítógéppel, és főképp programozási, alkalmazásfejlesztési célokkal hozták létre őket. Mindegyikhez szoftveres támogatás is jár a forgalmazótól, amelyben bizonyos alapfunkciókat is implementáltak.

A Leap Motion kivételével mindegyikükben megtalálható valamilyen szintű kéz- vagy testkövetés, ehhez a szenzor előtt tartózkodó személyen több jellegzetes pontot próbálnak azonosítani és követni. Implementálva vannak továbbá magasabb szintű algoritmusok is, ilyenek az arcmimikák felismerése (pl. mosolygás), vagy kézjelek, mint integetés, kattintás, ütés felismerése. Közös tulajdonság továbbá, hogy a mozdulatsorok követése nagy számítási kapacitást igényel, így a programok megfelelő futása érdekében érdemes csökkenteni a mintavételezési frekvenciát.

A Leap Motion kicsit más irányvonalon indult el a NUI-t támogató eszközök között, ugyanis ezt az eszközt leginkább egy kéz követésére készítették (habár korlátokozott körülmények között két kezet is megbízhatóan tud követni), azaz nem alkalmas a kar, a test vagy az arc mozdulatainak követésére. Cserébe viszont jóval nagyon pontosságot nyújt, mint a fent említett három eszköz, és érzékelési távolsága is jóval kisebb (<5 cm) mint a fentieké. Javítottak továbbá a szoftveres támogatáson is, ugyanis a felismerés/követés jóval kisebb számítási kapacitást igényel, mint a másik három megoldásnál.

Ezek az eszközök magas áruk és viszonylagos gyenge támogatottságuk miatt egyelőre nem terjedtek el eléggé PC-s piacon. A Microsoft ezen azzal próbált segíteni, hogy a Windows 8 operációs rendszerébe beépítette a Kinect támogatását, ezáltal a Metro felület vezérelhető ezzel az eszközzel.[1][2]

1. ábra Asus Xtion Pro[3]

2. ábra PrimeSense Carmina 1.09[4]

3. ábra Leap Motion[5]

2.3. A Natural User Interfaces-t nem támogató szenzorok

Megtalálhatóak a piacon hasonló funkcióval rendelkező eszközök, amelyek nem támogatják a NUI előírásait. Ezek közül a legelterjedtebb a Nintendo Wii és a Sony PlayStation-höz készített Move. Ezekről elmondható, hogy leginkább konzolokra készült játékokhoz használják kiegészítő eszközként, és nem terjedt el a PC-hez való felhasználásuk.

A Nintendo cég 2006-ban adta ki a játékok vezeték nélküli irányítását lehetővé tevő Wii Remote nevű eszközét, amely a mozgásérzékelés mellett gesztusokat is képes felismerni. Található benne egy gyorsulásérzékelő, valamint 10 darab LED 2-szer 5-ös elrendezésben, amelyek a központi konzollal kommunikálnak. A két szenzor kombinálásával lehetővé válik a pontos mozgáskövetés.

A Sony 2010 szeptemberében adta ki a Move-ot a PlayStation játékkonzoljához. Ebben is található mozgásérzékelő, valamint az eszköz egyik végében egy RGB LED, amelyek információiból a konzol meg tudja határozni a kézben tartandó eszköz pozícióját és orientációját.

A fenti eszközök nem felelnek meg a NUI előírásainak, ugyanis a jeladó eszközt kézben kell tartani az alkalmazások vezérlése érdekében. Mindenesetre elmondható, hogy mérföldkőnek számítanak a NUI-t támogató szenzorok fejlesztésének világában is.

4. ábra Nintendo Wii Remote[6]

5. ábra Sony Move[7]

2.4. Kinect

2.4.1. Történet

Az eredetileg Project Natal néven futó technológiát 2010 harmadik negyedévében mutatta be a Microsoft (ekkor még csak Xbox-ra). A bemutatott szenzor képes volt felismerni a mozgás-, a hang- és a mélységadatokat. Alig 3 nappal a megjelenés után (ekkor Európában hivatalosan még meg sem jelent) már sikerült feltörni a Kinectet, aminek a felhasználó követésére

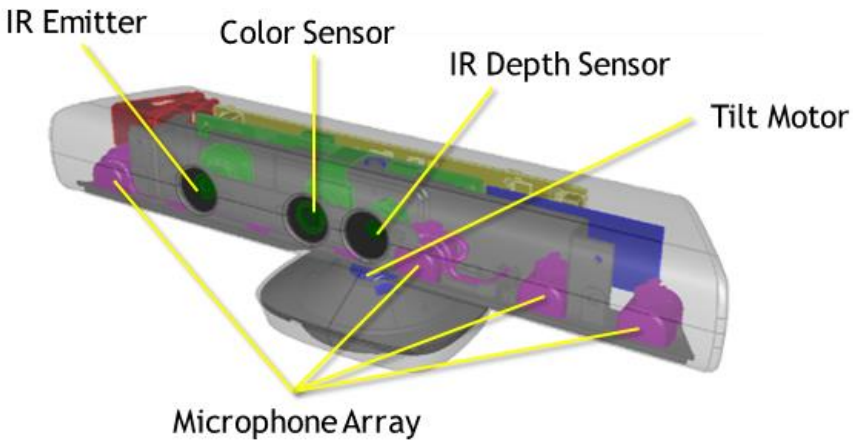
használatos motorját (tilt motor) sikerült PC-ről vezérelni. Ebben ösztönzőleg hathatott egy amerikai hardver cég 1000, majd később 2000 dolláros ajánlata, amit a Kinectet elsőként feltörőnek ajánlott fel. A Microsoft ekkor még ellenezte a Kinect Xbox játékoktól független felhasználását, és jogi lépéseket helyezett kilátásba az eszköz bármely más célra történő felhasználása esetén. Bejelentették továbbá, hogy olyan változtatásokat építenek az eszközbe, amelyek lehetetlenné teszik az Xbox-tól eltérő használatot. Időközben a Kinect bekerült a Guinness világrekordok könyvébe is, ugyanis minden addigi eladási rekordot megdöntött: két hónap alatt nem kevesebb, mint 8 millió darabot adtak el belőle világszerte. Ez naponta több mint 133.000 db-ot jelent. Nem meglepő, hogy a termék terjedésével párhuzamosan elkezdtek elterjedni a PC-s felhasználást elősegítő különböző megoldások, azaz tört SDK-k is. Ezek többé-kevésbé támogatták a Kinect funkcióinak kihasználását, viszont legtöbbjük elég alacsony szintű adatokat továbbított csak a felsőbb szintű alkalmazások felé.

A kezdeti elhatárolódás után (valószínűleg a tört SDK-k terjedését látva) a Microsoft 2011. június 16-án kiadta a Kinect SDK béta verzióját, amivel lehetővé vált az eszköz Windows 7 operációs rendszer alóli programozása Visual Basic, C++ és C# nyelven is. Ezzel a lépéssel a Microsoft feladta addigi nézeteit, hogy kizárólag játékra lehet felhasználni a Kinectet.

2012. január 10-én a CES konferencián (Consumer Electronics Association, az elektronikai termékeket gyártó cégek minden év januárjában megrendezett konferenciája, melyen az év során várható újdonságokat mutatják be) a Microsoft bejelenti a Kinect for Windows elkészültét, és forgalmazásának beindítását 250 dolláros áron. Ez az ár jóval drágább, mint az Xbox-os verzió ára, ugyanis itt a hardver árának egy részét nem tudják finanszírozni a játékok eladásából befolyó bevételből. A hardver megjelenésével párhuzamosan megjelent a hozzá tartozó fejlesztői készlet (SDK) első teljes verziója is. A működéshez egy erősebb 2 magos processzorra volt szükség, továbbá a Visual Studio 2010 valamely verziójára, valamint a .NET Framework 4.0-ra. 2012 májusában jelent meg az SDK 1.5-ös verziója, amely sok újdonságot jelentett az addigi változatokhoz képest. Megjelent a felhasználók arcának követése, valamint bekerült az SDK-ba a mozdulatsorok rögzítését végző, majd később azt lejátszani tudó szoftvermodul is.[8][9][10]

6. ábra Kinect for Xbox[14]

2.4.2. A Kinect hardver felépítése



7. ábra A Kinect felépítése[14]

A Kinect for Xbox a 7. ábrán található modulokból épül fel. Talán a legfontosabb részei az IR Emitter és az IR Depth Sensor. Az IR Emitter infravörös tartományú fényt bocsájt ki, amely visszaverődve a különböző tárgyakról, és az emberi testről az IR Depth Sensor-ban (ez egy monokróm CMOS infravörös szenzor) kerül feldolgozásra. Ebből a szenzorból lehet kinyerni a mélységi adatokat, ami talán a leggyakrabban használt része a Kinectnek. Az IR Depth Sensor a képpontokat távolsági adatokká konvertálja át, így ezek az adatok később mélységi mátrixként (vagy mélységi térképként) is felhasználhatóak. A szenzor megbízható érzékelési tartománya 50 cm-től 3 méterig terjed (a valós, ámde néha bizonytalan érzékelési tartomány 50 cm-től kb. 5 méterig terjed). A szenzor érzékelési pontossága mélységi adatokra (z tengely) 1 cm, míg szélességi és magassági adatokra (x és y tengely) milliméterekben mérhető. Az érzékelési pontosság a távolság változásával nem lineáris arányban változik, erről egy bővebb cikk itt olvasható: http://mathnathan.com/2011/02/depthvsdistance/. Az IR Depth Sensor által továbbított adatfolyam 640x480 pixel felbontású, és 11 bites távolsági adatokat tartalmaz, ezáltal 2048 db különböző távolsági adat ábrázolható. Az adatfolyam maximális frekvenciája 30 Hertz, de ez szoftveresen konfigurálható. Az IR Depth Sensor látószöge horizontálisan (x tengely) 57°, míg vertikálisan (y tengely) 43°. Ez utóbbi a beépített motor segítségével tovább javítható.

Található az eszközben egy szín szenzor (Color Sensor) is, amely egy webkamera lehetőségeit nyújtja a felhasználó számára. Az IR Dept Sensor-hoz hasonlóan ez is VGA-nak megfelelő 640x480 pixel méretű képeket vagy videókat készít 30 Hertzes frissítési frekvencia mellett.

A 4 darab mikrofon (Microphone Array) feladata a hangok rögzítése és továbbítása a konzol vagy a számítógép felé. A mikrofonoktól érkező hangfolyamok egyesével is feldolgozhatóak, ennek segítségével könnyen megállapítható, hogy a hangforrás a szenzorhoz képest milyen irányban található. A mikrofonok 16 KHz-es adatmintákat továbbítanak PCM kódolással (Pulse Code Modulation). A beszédfelismerő algoritmusok implementálása során nagy segítség lehet a hardverbe beépített automatikus zajszűrés (Ambient Noise Suppression).

Belekerült még a hardverbe egy vertikális mozgást lehetővé tévő motor is (Tilt Motor), amely az eszközt a talapzatához viszonyítva ±27°-kal képes mozgatni.

Az IR Depth Sensor és a Color Sensor fent említett 640x480-as felbontása helyett szoftveresen konfigurálható 1280x1024-es felbontás is, viszont itt csak 10 Hertzes frissítési frekvencia érhető el. Ez hasznos lehet, ha nagyobb felbontású képekre vagy mélységi adatokra van szükségünk, viszont ilyen frissítési frekvenciával nem lehet használható mozgóképet előállítani.

Található még az eszközben egy, az ábrán nem látható status LED, amely villogással jelzi, ha az eszköz áramellátás alatt van, és konstans módon világít, ha a szenzor használat alatt is van (ez azt jelenti, hogy a Kinect valamely szenzora éppen adatokat továbbít a konzol vagy a PC felé). További beépített szenzor a gyorsulásmérő is, viszont a Kinectre épülő alkalmazások tekintetében ez elég ritkán használt.

Ahogy a 6. ábrán is látható, a Kinectet nem csak a számítógép vagy az Xbox USB portjához kell csatlakoztatni, de külső áramellátásra is szüksége van. A Kinect maximális fogyasztása 12 watt, és ez jóval több az USB szabványban rögzített 2.5 wattos maximális áramfelvételnél, amely egy USB portra kötött eszköztől elvárható lenne.[11][12]

3. A szoftver környezet ismertetése

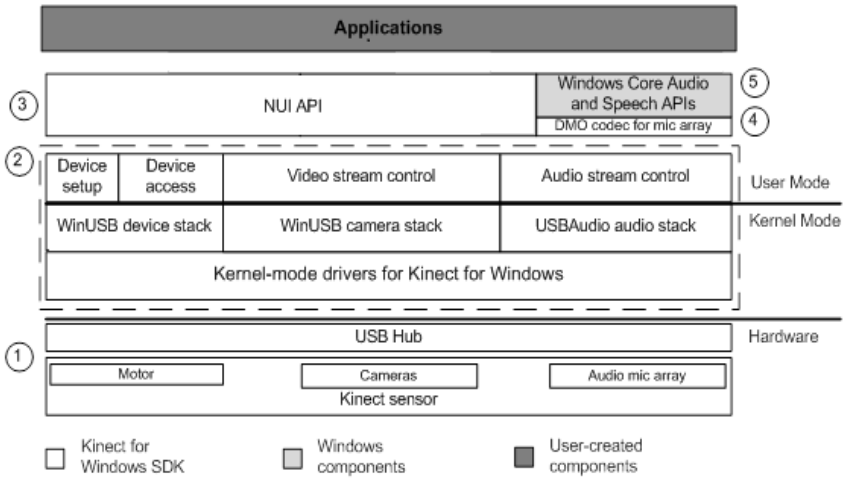
3.1. Áttekintés

A Kinect programozását számos eszköz lehetővé teszi, a legtöbbjük ingyenesen elérhető a fejlesztő számára. Ezek közül a legelterjedtebbek a Microsoft SDK-ja, az OpenNI 1.5 és 2.0 keretrendszerek, az OpenKinect, valamint a CL NUI (CodeLaboratories). Az alábbiakban bemutatom az általam használt fejlesztőeszközöket.

3.2. Microsoft SDK

Először a Microsoft készített PC-hez használható SDK-t béta verzióban a Kinect for Xbox termékhez, majd az első teljes verzió a Kinect for Windows megjelenésével párhuzamosan érkezett. A Microsoft üzletpolitikájából eredően az SDK nem nyílt forráskódú, és csak Windowsra érhető el, de ott legalább ingyenes a felhasználása (nem üzleti célokra). Mivel az SDK-t ugyanúgy a Windows adja ki, mint a Kinectet, ezért mindig ebben jelenik meg először a Kinectre érkező frissítések támogatása. További előnye, hogy rendkívül jól dokumentált, mind a programozói kézikönyv, mind a hozzá tartozó példák részletesen, és jól bemutatják a programozási felületet.

A 8. ábrán látható, hogy az SDK rétegezett felépítésű, és egy csatolófelületet biztosít a Kinect hardver és magasabb szintű algoritmusokat megvalósító NUI API között. A NUI API tetszőleges API lehet, amely támogatja a Kinectet, a legelterjedtebbeket tartalmazza a 3.1. pont. [13]



8. ábra A Microsoft SDK felépítése[13]

3.3.OpenNI

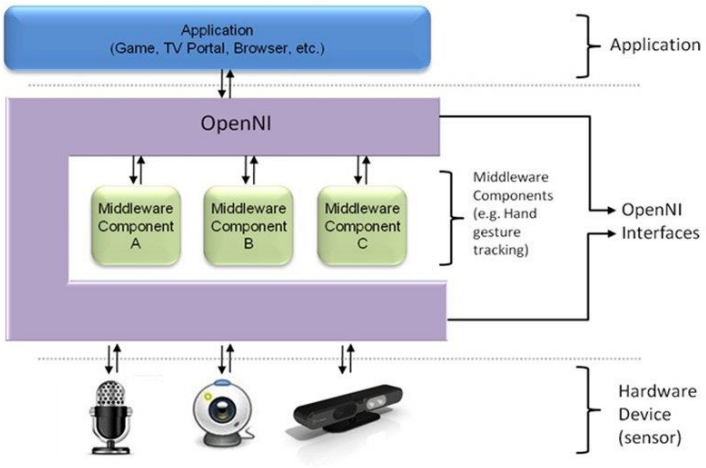
3.3.1. OpenNI 1.5

Az OpenNI nagyon fontos tulajdonsága, hogy nem kifejezetten a Kinecthez készült, hanem támogat minden olyan beviteli hardvert, ami megfelel az OpenNI által támasztott követelményeknek. Ez azt jelenti, hogy az OpenNI magában nem tartalmaz sem drivereket sem a szenzorok által küldött adatokat feldolgozó függvényeket. Ezeket a feladatokat az úgynevezett beépülő modulok végzik. Az OpenNI által ellátott három legfontosabb feladat:

- Biztosítani a beépülő modulok közötti gyors és hatékony kommunikációt.
- Egységes, jól definiált interfészeket nyújtani a beépülő modulok számára.
- Jól definiált és megfelelően dokumentált interfészeket nyújtani az OpenNI-t használó alkalmazások számára.

Nagy előnye az OpenNI-nak, hogy nincs megkötve a beépülő modulok száma, és az általuk támogatott funkcionalitások, így akár ugyanazt a funkcionalitást nyújtó, más gyártótól származó modult is használhatunk az OpenNI vagy az általunk írt programok módosítása nélkül. Ezek a modulok egymásra is épülhetnek, azaz az egyik modul kimenetét használhatja egy másik modul bemenetként.

Jelen pillanatban az OpenNI nem támogatja a Kinectbe épített motor, gyorsulásmérő és a status LED használatát, viszont lehetőséget biztosít az USB interfész programozására, azaz alacsony szintű eszközökkel az OpenNI-on keresztül is elérhetőek ezek a szolgáltatások.

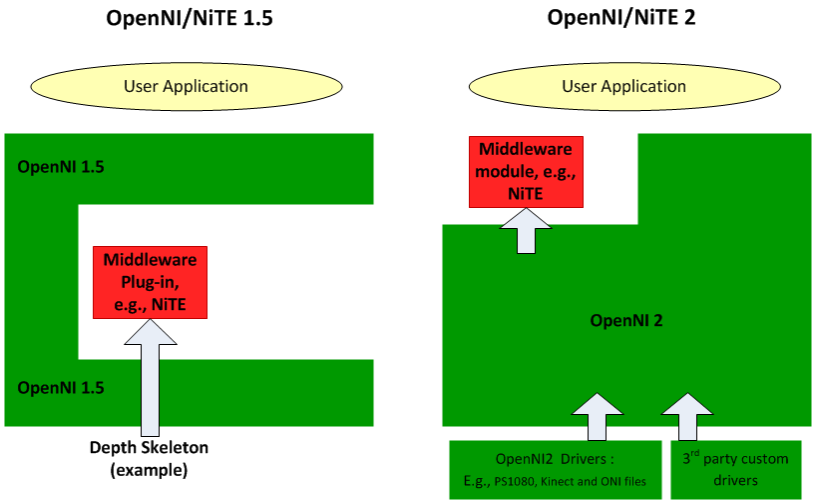


9. ábra Az OpenNI architektúra felépítése[15]

A 9. ábrán látható az OpenNI felépítése. A 3 réteg főbb feladatai:

- Applikáció réteg: Az OpenNI szolgáltatásait használó alkalmazásokat reprezentálja, amelyeknek minden, az OpenNI által támogatott platformon ugyanúgy kell viselkedniük.
- OpenNI Interfaces réteg: Az OpenNI-t megvalósító réteg. Kapcsolatot biztosít az applikációs és a hardver réteg között, valamint ezen rétegen belül futnak a különböző beépülő modulok, amelyek a hardver rétegtől kapott adatokat dolgozzák fel.
- Hardver réteg: A különböző szenzorokat megvalósító hardvereszközök, amelyek mindenfajta feldolgozás nélküli, úgynevezett nyers (raw) adatot továbbítanak az OpenNI felé.[15]

3.3.2. OpenNI 2.0



10. ábra : Az OpenNI 1.5 és a 2.0 közti felépítésbeli különbség[17]

2012 decemberében az OpenNI közösség kiadta az OpenNI SDK 2.0 változatát. Ez nem sok újdonságot tartogatott a felhasználók számára:

- Refaktorálták az API-t, azaz az algoritmusok gyorsabban futnak az 1.x változatokhoz képest.
- A nem közvetlenül a bejövő adatok feldolgozásával foglalkozó osztályok átkerültek a Middleware rétegbe. Ilyenek többek között a magasabb szintű kódok írását elősegítő osztályok (pl. XnList, XnHash, XnBitSet, XnThreadSafeQueue).
- A 2.0-s változatban már közvetlenül is elérhetőek a beépülő Middleware komponensek, nem csak közvetett módon az OpenNI-on keresztül.

Ezeken felül még néhány apróbb módosítás is belekerült a kódba többek között az alkalmazások fordításával kapcsolatban, de ezek nem befolyásolják jelentősen az API használatát.

A Kinect felhasználók számára viszont jelentős változás történt. A fórumok tanúsága alapján igazából emiatt volt szükség az új változat kiadására, és nem a fenti néhány módosítás miatt. A Microsoft rosszallóan tekintett az OpenNI közösségre, ugyanis hivatalosan terjesztette a Kinect olyan driverjét, amihez a Microsoft nem járult hozzá. Jelenleg a Microsoft a Kinect for Windowst támogatja hivatalosan, a Kinect for Xbox eszköz használatát pedig „eltűri” PC-s környezetben, viszont mindkettőt csak a saját SDK-jával és Windows operációs rendszer felett. Mivel az OpenNI 1.x saját drivert használt a Kinect minden szenzorához, így nem teljesítette a fenti feltételt. Az OpenNI 2.0-ban kidobták ezt a drivert, és a Microsoft hivatalos SDK-ja szükséges a Kinect használatához OpenNI felett. Ez azt jelenti, hogy OpenNI 2.0-val megszűnt a Kinect nem Windows platformon történő támogatása.

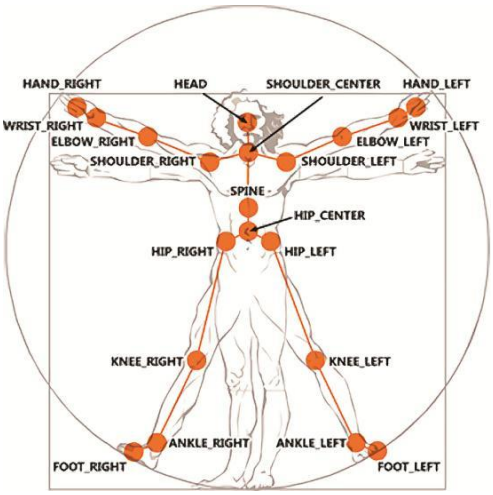
Mégsem teljesen reménytelen a helyzet azok számára, akik nem Windows felett szeretnék használni a Kinectet OpenNI 2.0-val. Elkészült ugyanis egy OpenNI2-FreenectDriver nevezetű szoftver az OpenKinect közösség jóvoltából, amely lehetővé teszi a Kinect használatát Linux platformon OpenNI2-vel. Ez hivatalosan nem az OpenNI 2.0 része, és hivatalosan nem is támogatott (inkább amolyan hobbiprojektként fut), viszont teljesen jól használható OpenNI 2.0-val Linux platform felett.[16][17]

3.3.3. NITE

A 9. és 10. ábrákon látható OpenNI Middleware rétegbe különböző modulokat (middleware components) építhetünk be. Ezek nagyrészt hardver specifikus modulok, amelyek elrejtik az OpenNI elől a hardver egyes tulajdonságait. Léteznek nem hardver specifikus modulok is, ezek tipikusan egy vagy több hardver specifikus modul kimenő adatait dolgozzák fel.

A NITE a PrimeSense cég által fejlesztett middleware komponensek halmaza az OpenNI-hoz. Ugyanez a cég fejlesztette ki a Microsoft számára a Kinect hardvert, így a megfelelő tudás birtokában ők készítették el a legjobban használható middleware komponenseket az OpenNI-hoz. A NITE az alábbi szolgáltatásokat nyújtja az alkalmazásfejlesztők számára:

- Teljes test analízis (full body analysis middleware): a teljes testet analizálja, és előállít néhány, az applikációs rétegben felhasználandó adatot. Ilyenek lehetnek pl. a testen azonosított pontok, vagy az ezekből a pontokból összeállított csontváz.
- Kéz analízis (hand point analysis middleware): A kéz pontjait analizálja, és előállít, majd követ egy pontot a tenyér közepén. Felfelé már csak ennek a pontnak az adatait propagálja.
- Gesztus analízis (gesture detection middleware): Végtag mozdulatokkal különböző, előre megadott mintákat lehet leírni (pl. integetés, kéz elmozdítása a testtől egy bizonyos szögben), amelyeket a NITE azonosítani tud. Felfelé már csak ezeknek a gesztusoknak a bekövetkeztét továbbítja.
- Kép analízis (scene analyzer middleware): Analizálja, és elkülöníti az előtérben található személyeket a háttérben található tárgyaktól, valamint azonosítja az előtérben található személyek egyes testrészeit, és különböző színekkel jelöli az egyes embereket. Ajánlott homogén hátteret használni ezen szolgáltatáshoz, ugyanis a követendő személlyel azonos mélységben található tárgyak (pl. szék vagy asztal) megzavarhatják a komponens működését. Előfordulhat, hogy egy szék vagy asztallábat sikerül a felhasználó lábának azonosítani. Ezt viszonylag gyorsan korrigálja a NITE, viszont a valós idejű használatban gondot okozhat.[18]



11. ábra A NITE teljes test analízise által előállított pontok, és azok nevei[18]

3.3.4. Példaalkalmazások

Önálló laboratórium 1. és 2. tárgyak keretében négy nagyobb példaalkalmazást készítettem, amelyek bemutatják a Kinect különböző szenzorainak felhasználását OpenNI 1.5 és NITE segítségével Linux platform alatt. Ezen alkalmazások a következőek:

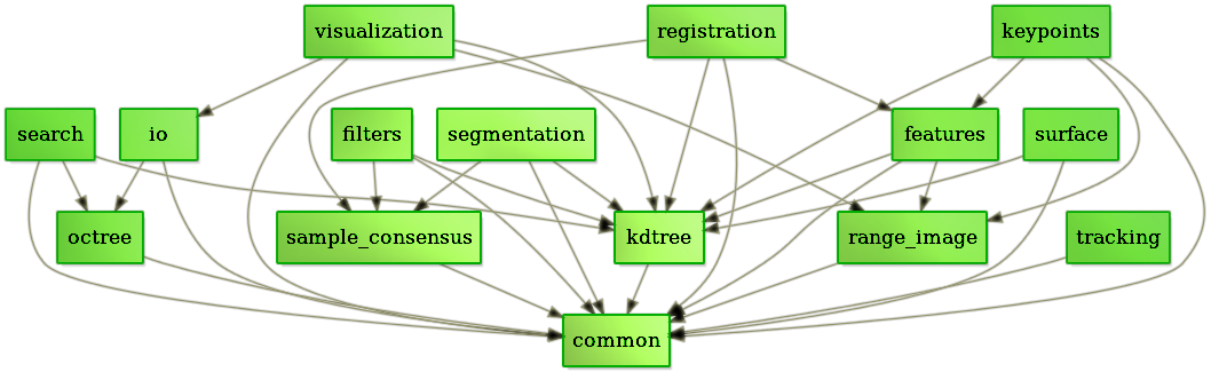
- MiddlePoint
- HandSlider
- ReactionGame

- DrawGame

Az alkalmazások forráskódjai nyilvánosan elérhetőek a projekt git tárolójában az alábbi címen: https://github.com/palgabor/Onlab--Kinect-?source=cc, valamint a hozzájuk tartozó dokumentáció megtalálható ugyanezen tárolóban található önálló labor beszámolókban.[1][2]

4. Point Cloud Library (PCL)

A PCL egy BSD licence alatt kiadott C++ alapú függvénykönyvtár, amely 2D és 3D pontok és pontfelhők feldolgozását teszi lehetővé. A függvénykönyvtár használata kereskedelmi, oktatási és kutatási célokra is ingyenes. Elérhető Windows, Linux, MacOS, Android és iOS operációs rendszerekre. A PCL a Boost-hoz hasonlóan a modularitás és a rétegezett architektúrák támogatásának érdekében modulokra osztott, amelyek külön-külön is használhatóak.



12. ábra A PCL moduljai[19]

A PCL alap adattípusa a PointCloud névre hallgató C++ template osztály, amely az alábbi függvényekkel rendelkezik:

- width(): A PointCloud-ban található rendezetlen pontok száma, vagy mátrixba rendezett pontok esetén a mátrix szélessége (oszlopok száma).
- height(): Mátrixba rendezett pontok esetén a mátrix sorainak száma (rendezetlen ponthalmaznál 1-gyel tér vissza).
- points(): A PointCloud által tartalmazott pontokat adja vissza (PointT típusokat tartalmazó vektor).
- is_dense(): A PointCloud-ban található összes pontra megvizsgálja, hogy azok koordinátái léteznek-e és végesek-e. Ha az előbbi két feltétel a ponthalmaz összes pontjára fennáll, akkor a függvény igaz értékkel tér vissza, minden egyéb esetben hamissal.

A PointCloud template osztályban használt főbb pont típusok:

- PointXY: pont X, Y koordinátákkal.
- PointXYZ: pont X, Y, Z koordinátákkal.
- PointXYZI: pont X, Y, Z koordinátákkal és intenzitás értékkel

- PointXYZRGB: pont X, Y, Z koordinátákkal és R, G, B színértékekkel
- PointXYZRGBA: pont X, Y, Z koordinátákkal és uint32_t-ben megadott színértékkel

A fentieken kívül a felhasználó definiálhat saját pont típusokat is, amelyben akár egyéni módon is tárolhat színinformációkat (pl. CYMK). A PointCloud és a különböző Point osztályok a PCL Common moduljában találhatóak.

A PCL használatához nincs minden pillanatban szükségünk egy valós idejű ponthalmazt szolgáltató szenzorra, a PCD szolgáltatás használatával el tudunk tárolni fájlban is pontokat, amin a PCL segítségével műveleteket is tudunk végezni. Ezeket a pontokat megadhatjuk ASCII formátumban, illetve a PCL által előzőleg a memóriában eltárolt Point vektorok bináris memóriaképének formátumában. ASCII formátum használata esetén a fájl elején néhány vezérlőinformációt is meg kell adnunk, például a használt PCD verziót, a pontok dimenzióinak számát, az ábrázolás pontosságát, a pontok darabszámát, valamint hogy ASCII vagy bináris formátumot használunk-e. Utóbbira látható egy jó példa a [20]-as hivatkozás 12. oldalán.[19][20]

4.1.A PCL Registration modulja

A félév során a PCL Registration moduljával ismerkedtem meg mélyrehatóbban. A Registration modul PointCloud előállítására képes két különböző nézőpontból készült kép összefűzésével, vagy ugyanabból a nézőpontból, de különböző időpillanatokban készült képek összefűzésével. A modulban elérhetőek különböző algoritmusok, amelyek szignifikáns pontok megjelölésével meghatározzák a két kép közötti elmozdulást, legtöbbször négyzetes hiba formájában. Természetesen ez az ismert algoritmusokhoz hasonlóan viszonylag kicsi elmozdulásra működnek helyesen, ugyanis nagy elmozdulás esetén jelentkezhet az apertura probléma, amelyet külön szükséges kezelni.[22]

Az modulból az alábbi szolgáltatásokat használtam fel a féléves munkám során:

- IterativeClosestPoint (ICP): Páronként összehasonlítja a PointCloud pontjait a legközelebbi pontokkal, és két-két pont között négyzetes hibát mér (ez a hibamérés történhet színre, intenzitásra, és sok egyéb dologra is). Ha a mért hiba egy megadott kritérium alá csökken, akkor feltételezi, hogy a régi képen lévő pont az új képen az éppen mért pontnak felel meg. A kritériumot (threshold) nem érdemes túl nagyra állítani, ugyanis ebben az esetben sok fals pontot is megtalál az algoritmus, viszont túl kicsire se érdemes állítani, ugyanis két mérés között változhatnak egyes pontok intenzitásértékei, vagy akár a háttér megvilágítása is. Nem lehet általánosságban elfogadható ajánlást mondani a kritérium értékére, ez leginkább az alkalmazástól, az alkalmazás használatának körülményeitől, a kívánt pontosságtól és a rendelkezésre álló feldolgozási kapacitástól (avagy a valós idejűség elvártságától) függ.[20][21][23]
- Sample Consensus Initial Alignment (SAC-IA): Előfordulhat, hogy a túl nagy elmozdulás miatt nem célszerű az ICP-t használni a ponthalmazra, ebben az esetben célszerű először lefuttatni a SAC-IA algoritmust a ponthalmazra, majd ezután az ICP-t. A SAC-IA nevezetes pontokat (legtöbbször sarokpontokat) próbál keresni a

ponthalmazban, és ezen pontok között az elmozdulás függvényében egy transzformációt határoz meg. Fontos tulajdonsága, hogy a futási idő gyorsítása érdekében nem minden pontra hajtja végre a transzformációt, csupán az előbb említett sarokpontok között keresi az elmozdulás nagyságát. A SAC-IA algoritmus kimenete átadható az ICP bemenetének.[21][24]

- Túl nagy pontfelhő, túl nagy elmozdulás, vagy a valós idejűség követelménye miatt előfordulhat, hogy a fenti két algoritmus futási ideje túl nagy lesz. Ez esetben célszerű először mintavételezni a pontfelhőt a VoxelGrid algoritmussal. A félév során végzett munkám során jellemzően nem volt rá szükség, de elképzelhető, hogy más környezetben történő felhasználás esetén szükséges lesz a használata.[25]

5. Új alapokon nyugvó jelszókezelő alkalmazás tervezése

5.1. Azonosítási lehetőségek napjainkban

A számítástechnika világában gyakran előfordul, hogy bizonyos erőforrásokhoz vagy információkhoz való hozzáférést korlátozni szeretnénk egy felhasználóra vagy felhasználók egy csoportjára. Ennek érdekében a felhasználónak azonosítania kell magát a szolgáltatás használata előtt. Az ilyen típusú korlátozásokat három nagy csoportra oszthatjuk:

- Tudás alapú korlátozás: a felhasználó ismer egy jelszót vagy egy PIN-kódot, amellyel azonosítja magát a használni kívánt rendszerben.
- Birtok alapú korlátozás: a felhasználó rendelkezik egy tárggyal, amely segítségével azonosítani tudja magát a használni kívánt rendszerben. Ilyen tárgy lehet például egy kulcs, token, vonalkód, mágneskártya, chipkárta vagy valamilyen RFID alapú megoldás.
- Biometria alapú korlátozás: a felhasználó bizonyos biológiai tulajdonságai segítségével azonosítja magát a használni kívánt rendszerben. Ilyen tulajdonságok lehetnek például hang, ujjlenyomat, arcfelismerés, írisz scannelés vagy retina scannelés.

Napjainkban a számítástechnikában messze a legelterjedtebb az első módszer. Kétség kívül implementálni és használni is ezt a legkönnyebb, de számtalan hátránnyal is rendelkezik. A felhasználók általában egy könnyen megjegyezhető jelszót vagy PIN-kódot választanak maguknak, amelyet több rendszerben is használnak, és jellemzően nagyon ritkán, vagy sosem módosítják azt. Ez az adat sokszor akár social engineering használatával is megszerezhető a felhasználótól, vagy egyéb, megtévesztő weboldalak használatával is ellopható.

5.2.Lehetőségek a jelszó alapú azonosítás fejlesztésében

Az előző pontban említett hátránya a jelszó alapú azonosítási módoknak, hogy a felhasználók gyakran könnyen megjegyezhető jelszavakat választanak. Nem elhanyagolható szempont, hogy a könnyen megjegyezhető jelszavak gyakran könnyen meg is szerezhetőek a

felhasználótól. A jelszólopások jelentős része manapság nem tradicionális informatikai eszközökkel történik (feltörés, visszafejtés, stb), hanem egyéb, emberi tulajdonságok kihasználásával, mint social engineering, megtévesztő weblapok készítése, stb. A jelszó alapú azonosítás biztonságának növelése érdekében ezért célszerű lenne csökkenteni az ilyen módon történő jelszólopásokat. Ennek egyik módszere lehet, hogy elérjük azt, hogy a felhasználó ugyan tudja a jelszavát, de nem tudja azt elmondani senkinek, ugyanis a jelenleg elterjedt adatbeviteli eszközökkel nem lehet azt digitalizálni, továbbá szavak helyett használhatunk alakzatokat is.

5.3. Jelszókezelés NUI segítségével

Joggal merülhet fel a felhasználókban az az igény, hogy az alkalmazások olyan azonosítási lehetőségekkel rendelkezzenek, hogy a felhasználóknak ne kelljen félniük attól, hogy illetéktelenek is hozzáférnek az ő nevükben az adott szolgáltatáshoz. Jó megoldás lehet erre, ha az azonosítandó és az azonosító fél is rendelkezik megfelelő hardveres és szoftveres környezettel, akkor NUI technológiák segítségével is azonosíthatja magát a felhasználó. Természetesen itt a jelszó alapú azonosítás filozófiájától gyökereiben kell eltérnünk, ugyanis egy jelszó „elmutogatásának” sok értelme nem lenne, viszont egy előre meghatározott alakzattal könnyen azonosíthatja magát a felhasználó. Ezzel a módszerrel a fent említett jelszókezelési problémák nagy részét megoldjuk, ugyanis egy alakzatot sokkal nehezebb (vagy akár lehetetlen is) akarva vagy akaratlanul elmondani egy harmadik félnek, ugyanis akár egy egyszerű alakzatot is rengeteg módon lehet bevinni egy NUI eszköz segítségével. A pontos alakzaton kívül számít a felhasználó intuíciója, dinamikája és mozgási tulajdonságai. Ezáltal elérjük azt, hogy a felhasználó a bevitt információ egy részének tudatában van (a rajzolt alakzat), míg másik részének nincs tudatában, viszont tudta nélkül is átviszi ezeket az információkat a szenzor felé. Az azonosító alkalmazás természetesen ezen információkat is felhasználja a felhasználó azonosítására, így egy egyszerű alakzat ismerete nem elég a sikeres azonosításhoz.

Természetesen ennek a módszernek is megvannak a saját hátrányai. Kétszer egymás után pontosan ugyanazt az alakzatot lehetetlen bevinni, ezért nem tudjuk egyértelműen egymáshoz rendelni az eltárolt és a bevitt alakzat pontjait, mindig lesz valamekkora hiba a megadásban. Ebből kifolyólag 100%-os pontosság soha nem érhető el az azonosításban, azaz mindig lesz valamekkora bizonytalanság abban, hogy a bevitt alakzat valóban megegyezik-e tárolttal. Itt kísérletek sorozatával egy kritérium érték meghatározására van szükség, amelyben megadhatjuk, hogy mekkora abszolút vagy relatív pontosság esetén fogadható el egy alakzat.

További hátrány, hogy a szenzor előtti mozgások könnyen láthatóak egy másik személy által is, sőt, még akarva sem tudjuk elrejteni azt egy érdeklődő elől. Mindenképp szükség van ezért arra, hogy az azonosító alkalmazás ne csupán a bevitt alakzatot hasonlítsa össze a tárolttal, hanem egyéb információkat is gyűjtsön a felhasználóról, lehetőleg olyanokat, amelyekkel a felhasználó tudata alatt rendelkezik és egy illetéktelen felhasználó nem, vagy csak nehezen tudja azokat meghamisítani.

5.4. A jelszókezelésben használható paraméterek

A Kinect szenzor és szoftveres környezetének ismeretében az alábbi paraméterek használhatóak a felhasználó azonosítása céljából:

- Bevitt alakzat felismerése
- Az alakzat bevitelének sebessége
- Különböző referenciapontok közti sebesség
- A hajlítások szögének számítása
- Az alakzat bevitelének dinamikája
- Jellemző gesztusok felismerése
- Arcfelismerés

5.5. Jelszókezelő alkalmazás megvalósítása Kinecttel és PCL-lel

A félév során megvizsgáltam a lehetőségét a fenti pontokban ismertetett alkalmazás elkészítésének. Terveim szerint egy spam szűrőhöz hasonló tanuló rendszert szeretnék készíteni, amely az 5.4. pontban említett tulajdonságokat vizsgálja a felhasználó azonosítása közben. Az alkalmazás felhasználná a spam szűrőkben alkalmazott pontozási rendszert, azaz az 5.4. pontban említett tulajdonságokat súlyokkal látná el, és ha a bevitt tulajdonság megegyezik annak tárolt értékével, akkor egy megfelelő pontszámot adna az eredményre. Az alkalmazásban definiálva lenne egy kritérium érték, és ha a bevitt alakzat tulajdonságai teljesítik ezt a kritérium értéket, akkor az azonosítást sikeresnek tekinthetjük.

Az alkalmazás tanuló rendszer is lenne, ugyanis minden sikeres azonosításról információkat tárolna el, és akár futás közben is módosítaná a különböző tulajdonságokhoz rendelt súlyokat. Elképzelhető, hogy egy felhasználó mindig ugyanazzal a dinamikával adja meg a jelszavát, de jellemzően pontatlanabb alakzatot „rajzol”. Ebben az esetben természetesen célszerű csökkenteni az alakzat pontosságára adott pontok súlyát, és növelni a dinamikára adott pontok súlyát. Ezzel elérhető lenne, hogy a kezdeti bizonytalanságok után az alkalmazás kényelmesen használható legyen, és automatikusan személyre szabja magát a felhasználó szokásaihoz alkalmazkodva.

Az alkalmazásnak természetesen meg kell küzdenie azzal a hátránnyal, hogy az azonosítás pontossága soha nem lesz 100%-os (hasonlóan a spam szűrőkhöz), így az alkalmazás elkészítéséhez definiálni kell egy megfelelő kritérium értéket, amelyet nagy számú mérésnek kell megelőznie.

Az alkalmazás a PCL függvénykönyvtár segítségével fogja összehasonlítani a bevitt és a tárolt pontfelhőket, továbbá az 5.4. pontban említett egyéb tulajdonságok vizsgálatához is fel lehet használni. Az alkalmazás a biometriai tulajdonságok felismeréséhez az OpenNI és a NITE magasabb szintű szolgáltatásait fogja felhasználni. Ezen biometriai tulajdonságok nagyon kis súllyal fognak szerepelni a végső alkalmazásban, nagyrészt inkább tesztelési célokra kerülnek felhasználásra. Ha a későbbi mérések biztosítják a biometriai azonosítás pontosságát, akkor akár nagyobb súllyal is szerepet kaphatnak az azonosítás menetében.

6. További lehetőségek a diplomamunka során

A diplomamunkám folytatásában elsősorban az 5.5. pontban részletezett alkalmazást szeretném elkészíteni, valamint végrehajtani azt a nagy számú mérést, amely szükségeltetik az alkalmazás elkészítéséhez. Ehhez részletesen át kell tekintenem a szakterület irodalmát olyan szempontból, hogy az ember természetes mozgása során milyen ergonómiai tulajdonságokkal rendelkezik, valamint ezen tulajdonságok közül melyek egyeznek meg két, közel azonos mozgássorozat végrehajtása során. Ezen ergonómiai jellemzők egyezőségéből következtetéseket kell levonnom arra, hogy az alkalmazásban melyik tulajdonság milyen súllyal szerepeljen az azonosítás folyamatában.

A bevitt alakzat feldolgozásában ki kell választani a megfelelő algoritmusokat, hogy a feldolgozás minél pontosabb és gyorsabb legyen, így lerövidítve az azonosításhoz szükséges időt.

A tárolt adatokat megfelelő titkosítással kell ellátni, hogy abból adott esetben ne lehessen visszafejteni a felhasználó egyes ergonómiai tulajdonságait.

Szükséges továbbá egy jól használható felhasználói felület elkészítése is. Ennek a legnagyobb jelentősége az alkalmazás tesztelésében van, ugyanis fontos lesz a tesztelés során a különböző kritérium értékek és tulajdonságok súlyozásának változtatása az alkalmazás futása közben.

7. Irodalomjegyzék

[1] https://github.com/palgabor/Onlab--Kinect-/blob/master/docs/onlab1/beszamolo.pdf

[2] https://github.com/palgabor/Onlab--Kinect-/blob/master/docs/onlab2/beszamolo.pdf

[3] http://www.primesense.com/casestudies/xtion/

[4] https://www.faceshift.com/news/carmine-1-09/

[5] http://lifelabs.ucp.org/tiny-device-huge-potential-how-leap-motion-will-change-computing/

[6] http://metroid.wikia.com/wiki/Wii_Remote

[7] http://news.softpedia.com/newsImage/Sony-Plans-For-Move-to-Come-to-the-PC-2.jpg/

[8] http://pcforum.hu/hirek/12381/Feltortek+a+Microsoft+Kinectet.html

[9] http://www.hwsw.hu/hirek/46889/microsoft-kinect-sdk-fejlesztes.html

[10] http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx

[11] http://msdn.microsoft.com/en-us/library/jj131033.aspx

[12] Andrew Davison – Kinect Open Sourse Programming Secrets: 2. oldal

[13] http://msdn.microsoft.com/en-us/library/hh855347.aspx

[14] http://blogs.msdn.com/b/kinectforwindows/

[15] http://openni.org/Documentation/ProgrammerGuide.html

[16] http://www.openni.org/about/

[17] http://www.openni.org/reference-guide/?t=index.html

[18] http://www.primesense.com/nite

[19] http://pointclouds.org/about/

[20] http://www.cse.buffalo.edu/~jryde/cse673/files/pcl_tutorial.pdf

[21] http://www.pointclouds.org/assets/icra2012/registration.pdf

[22] http://docs.pointclouds.org/trunk/group__registration.html

[23] http://pointclouds.org/documentation/tutorials/iterative_closest_point.php

[24] http://docs.pointclouds.org/trunk/classpcl_1_1_sample_consensus_initial_alignment.html

[25] http://pointclouds.org/documentation/tutorials/voxel_grid.php

8. Ábrajegyzék