



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

Gépészmérnöki Kar

Mechatronika, Optika és Gépészeti Informatika Tanszék

Ipari robotkar távoli vezérlése egyedi szenzorok és aktuátorok alkalmazásával

Készítette:

Helli Márton
Mechatronikai mérnök
MSc szakos hallgató

Nyirő Péter
Mechatronikai mérnök
MSc szakos hallgató

Subert Alexandra
Mechatronikai mérnök
MSc szakos hallgató

Konzulens:

Dr. Korondi Péter
tanszékvezető, egyetemi
tanár

Külső konzulens:

Dr. Trygve Thomessen
ipari professzor

**Országos Tudományos Diákköri Konferencia
Budapest, 2015**

Tartalomjegyzék

TARTALOMJEGYZÉK.....	II
ÁBRAJEGYZÉK.....	IV
1 BEVEZETÉS.....	1
2 A MODELL FELÉPÍTÉSE.....	3
2.1 ELMÉLETI MŰKÖDÉS	3
2.2 SZIMULÁCIÓS MODELL	4
2.3 VALÓS MODELL.....	6
2.3.1 Kommunikációk.....	6
2.3.2 Robot program.....	8
2.3.3 Vezérlési program.....	11
3 KINEMATIKAI RENDSZER.....	12
3.1 ROBOT KINEMATIKA	12
3.2 ROBOT INVERZ KINEMATIKA	13
3.3 IMPLEMENTÁLÁS LABVIEW SEGÍTSÉGÉVEL.....	15
3.3.1 Forward kinematics.....	16
3.3.2 Inverse kinematics	17
4 KINECT SZENZOR.....	19
4.1 A SZENZOR MŰKÖDÉSE	19
4.2 MIDDLEWARE	20
4.3 A KINECT SZENZORBÓL SZÁRMAZÓ JELEK FELDOLGOZÁSA	22
4.3.1 Követelmények.....	22
4.3.2 Jelszűrés, simítás	23
4.3.3 Jósági paraméterek.....	26
4.3.4 Orientáció kezelése.....	27
5 A VEZÉRLÉS.....	29
5.1 A VEZÉRLŐ PROGRAM RÉSZEI.....	29
5.2 SKÁLÁZÁS ÉS OFFSET BEÁLLÍTÁS	31
5.3 A VEZÉRLŐ PROGRAM KEZELŐFELÜLETE	32
5.4 VEZÉRLŐ KESZTYŰ.....	33
5.5 A KAR MOZGÁSÁNAK SZIMULÁCIÓJA	34
5.5.1 Pozícióvezérlés	35
5.5.2 Csuklóvezérlés	39
5.6 A FEJ MOZGÁSÁNAK SZIMULÁCIÓJA	40
6 ÉRTÉKELÉS.....	42

7	TOVÁBBI MUNKÁK.....	44
7.1	ERŐSZABÁLYOZÁS, TRAJEKTÓRIA-SIMÍTÁS.....	44
7.2	MANIPULÁCIÓS KÓDRENDSZER TOVÁBBFEJLESZTÉSE	45
7.3	VISSZAJELZÉSI MECHANIZMUSOK KIDOLGOZÁSA	45
7.4	IMPLEMENTÁLÁS KÜLÖNBÖZŐ ROBOTTÍPUSOKRA	45
7.5	KINECT ÉS KÉPFELDOLGOZÁS MAGASABB SZINTŰ FELHASZNÁLÁSA	46
7.6	BEÉPÍTÉS HASZNÁLATBAN LÉVŐ TÁVOLI VEZÉRLÉST ALKALMAZÓ RENDSZERBE.....	46
	IRODALOMJEGYZÉK	47

Ábrajegyzék

1. ábra: A szoftvercsomag teszteléséhez használt robot	2
2. ábra: Nachi MC70 robot mechanikai felépítése	4
3. ábra: A szimulációs modell blokk-diagramja	6
4. ábra: Egyszerű TCP/IP kapcsolat LabVIEW alapon	7
5. ábra: Robot program működése közben	9
6. ábra: A robot program 3D-s megjelenítése	10
7. ábra: A vezérlési program felhasználói felülete	11
8. ábra: A kinematikai feladat	12
9. ábra: Az inverz kinematikai feladat	14
10. ábra: Direkt kinematika - front panel	16
11. ábra: Direkt kinematika - blokk diagram	16
12. ábra: Inverz kinematika - front panel	17
13. ábra: Inverz kinematika - blokk diagram	17
14. ábra: Jacobi mátrix számítása	18
15. ábra Link transzformáció	18
16. ábra: Microsoft Kinect szenzor	19
17. ábra: OpenNI platform	21
18. ábra: A Kinesthesia motion tracking rendszere	21
19. ábra: A Kinesthesia feldolgozatlan jelei	23
20. ábra: Mozgó átlagolás	24
21. ábra: Jelsimítás	25
22. ábra: A jelszűrés programrészlete	26
23. ábra: Jelkövetés	26
24. ábra: Szabálytalan üzemállapot	27
25. ábra: A vezérlő program kezelőfelülete	33
26. ábra: A rendelkezésünkre bocsájtott vezérlő kesztyű	34
27. ábra: A kar mozgásterjedelme	36
28. ábra: Cirkumdukciós kúp	36
29. ábra: A pozícióvezérlés paraméterei	37
30. ábra: A pozícióvezérlést megadó programrészlet	39
31. ábra: A szögvezérlés paraméterei	40

32. ábra: A kamera helyzetének megadása nyomógombokkal	41
33. ábra: Vezérlő kesztyű használat közben	43
34. Ábra: Robot mozgásának tesztje.....	43

1 Bevezetés

A gyártóipari cégek jelentős mennyiségben alkalmaznak ipari robotokat a legkülönbözőbb gyártási folyamatokhoz. Széles körű alkalmazásuk miatt ezeknek a gépeknek az installációs költségei egy robotfejlesztő cég életében gyakran rendkívül fontos kérdésként merülnek fel. Egyik legnagyobb probléma az operátor járulékos költségei.

Egy gyártó cég ritkán foglalkoztat robotok telepítésében jártas alkalmazottat, aki képes lenne a telepítési feladatot megoldani, ezért egy új robot vásárlásakor (esetleg egy meglévő javításakor vagy újrakonfigurálásakor) a robotfejlesztő cégtől rendel egy operátort, aki feltelepíti és beüzemeli a robotot a megfelelő kéréseknek és körülményeknek megfelelően. Az operátor utazásának, ellátásának költségeit és az egyéb, a személyes jelenlét miatt felmerülő kiadásokat a robotfejlesztő cégnek kell állnia. Ez a költség a telepítés helyétől és körülményeitől függően lehet igen számottevő nagyságú is. Az operátor személyes jelenléte a gyártóipari cégnél nem csak a robotfejlesztő cég költségeit növeli meg, de az installálás idejét is meghosszabbítja az utazási idő és az út megszervezése miatt.

Az itt bemutatott projekt egy alternatív lehetőséget vázol fel az ipari robotok telepítésére, melynek célja a robotfejlesztő cégek telepítési költségeinek csökkentése. Egy telemanipulációt megvalósító szoftvercsomag és néhány berendezés segítségével megoldhatóvá válik, hogy az operátor személyes jelenléte nélkül is be lehessen üzemeltetni az ipari robotot. A dokumentum bemutatja ennek a módszernek az elméleti összefoglalását, valamint leírja a szoftvercsomag megírásához szükséges technikai háttérrel fejezetekre osztva. A dolgozat írásával párhuzamosan különböző tesztek készültek el, ezeknek rövid bemutatására is sor kerül. Végül a dolgozat a modell elemeinek felépítése alapján levonható tanulságokat és a továbbfejlesztési lehetőségeket részletezi.



1. ábra: A szoftvercsomag teszteléséhez használt robot

Forrás: Saját fénykép

A dolgozat elkészítéséhez nagymértékben hozzájárult Prof. Ph. D. Trygve Thomessen (PPM AS cégvezető), valamint Dr. Korondi Péter (Budapesti Műszaki és Gazdaságtudományi Egyetem). Köszönetet mondunk a tesztek elvégzésében és a programfejlesztésben Suzuki Keita (Chuo University) segítségével. Az itt felsorolt személyek támogatása nélkül ez a projekt nem jöhetett volna létre.

2 A modell felépítése

2.1 Elméleti működés

A telemanipulációs berendezés két szoftverből, egy robotból, egy irányító kesztyűből, egy Kinect kamerarendszerből és egy virtuális 3D-s szemüvegből áll. Az egyik szoftver az operátor számítógépén, a másik a gyártó cég egyik számítógépén található. Mivel a továbbiakban többször is szó lesz a két oldalról, az operátor szoftverére vezérlési programként, a gyártó oldali szoftverre robot programként hivatkozik a dokumentum.

A vezérlési program és a robot program is a robotfejlesztő cég által kiadott szoftver, amelyek közül a robot programot a robottal együtt vásárolja meg a gyártó cég. A robot installálása során szükséges egy felügyelő személy, aki a robot oldalán ellenőrzi, hogy minden rendben megy-e és aki internet segítségével folyamatosan kapcsolatban marad az operátorral. A felügyelő feladata, hogy telepítse a robot programot. Ezután a robot program automatikusan felismeri a helyi hálózatra kötött robotot és innentől kezdve folyamatosan kapcsolatban marad vele.

Az operátor oldalán a vezérlési program, az irányító kesztyű, a kinect kamera és a virtuális szemüveg mind az operátor számítógépére van telepítve. A vezérlési program képes kommunikálni ezekkel a perifériákkal, továbbá képes interneten keresztül kapcsolatot létesíteni a robot programmal.

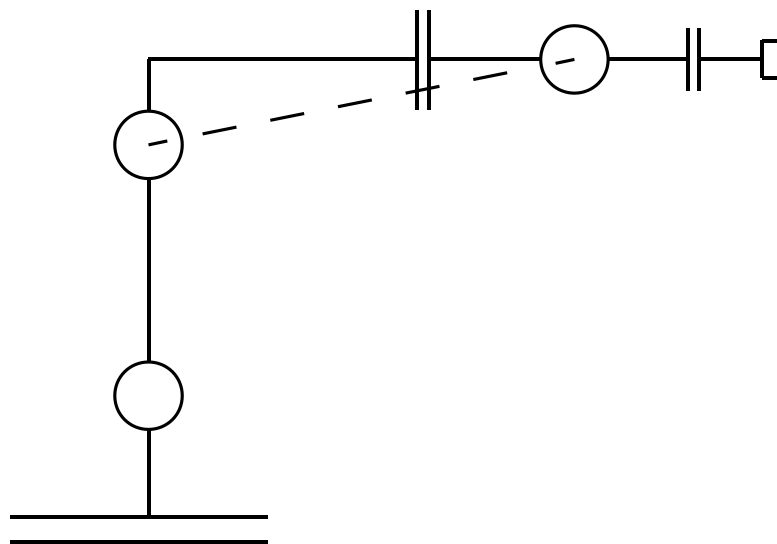
A robot telepítését az operátor fogja tehát végezni az eszközök segítségével. A kesztyű és a kamera az operátor kezének és ujjainak megfelelő mozgásával jeleket ad a vezérlési programnak, amely továbbítja ezeket a robot programnak. A robot program a robot konfigurációjától függően értelmezi, és parancsok formájában átadja a robot számítógépének, amely ad valamilyen visszajelzést. Ezután a robot program válaszol a vezérlési programnak, amely megjeleníti az eredményt az operátor virtuális szemüvegében. Az egész folyamatot a felügyelő ellenőrzi, aki egy deadman switch kapcsoló segítségével képes azonnal leállítani a robotot balesetveszély esetén. A vezérlési program segítségével az operátor is teljes körű információt kap arról, hogy a

vezérlés alapján hogyan kell működnie a robotnak, illetve a visszajelzések alapján hogyan változott valójában a robot állapota. Az operátor feladatát továbbá segíti a két szoftver között megvalósított dinamikus kommunikáció és az automatikus funkciók, valamint a robot munkaterében elhelyezett IP kamerák, amelyek valós képet adnak a robotról.

2.2 Szimulációs modell

A projekt jelenlegi verziója két részletben készült el. Először még nem valós roboton lett letesztelve, hanem kizárólag szoftver szintjén lettek kipróbálva a számolások. Ez a verzió a szoftver valós környezetét szimulálta, ezért a szimulációs modell nevet kapta.

Már a szimulációs modell is konkrét eszközök kiválasztásával valósult meg. A telepítendő robot egy Nachi gyártmányú, FD11 controller-re kapcsolt MC70 típusú ipari robot. Ez egy 6 szabadságfokkal rendelkező, AC Servo motorokkal meghajtott, közepesen nehéz kategóriába sorolt pakoló robot.



2. ábra: Nachi MC70 robot mechanikai felépítése

Forrás: Saját készítés

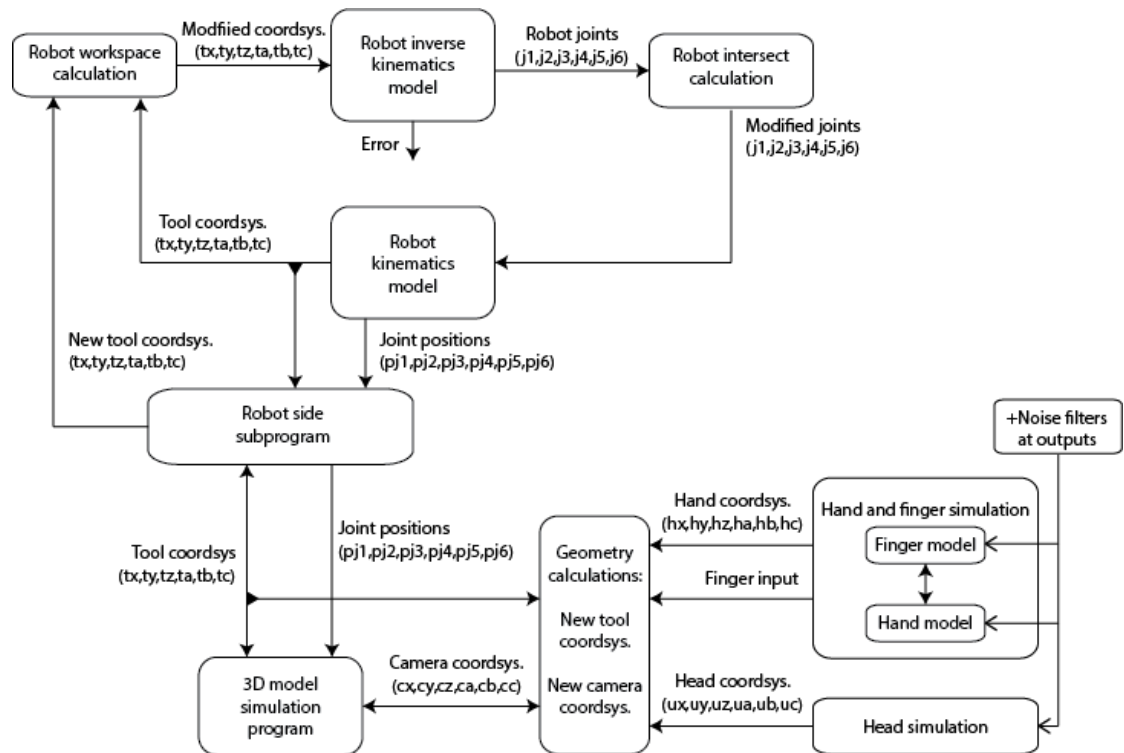
Az irányító kesztyű szimulációjához egy jelenleg még fejlesztés alatt lévő prototípus, a japán Chuo University egyik laboratóriumi telemanipulációs kesztyűje alapján lett leprogramozva. A kesztyű analóg módon képes érzékelni azt, hogy az operátor melyik

ujja van behajlítva és melyik nincs, egymástól teljesen függetlenül. Továbbá a kesztyű ujjain el van helyezve öt kisméretű excenter, amely visszajelzést ad az operátor számára. Ezeknek a motoroknak a kezelése még nem lett beillesztve a szimulációs modellbe.

Az Oculus VR Inc. által támogatott, jelenleg is fejlesztés alatt álló Oculus Rift DK1 lett kiválasztva virtuális szemüveggént. (1) Ez képes a fej orientációját valós időben eljuttatni a vezérlési programba, valamint monitorként is funkcionál. Ahhoz, hogy abszolút pozíciót is érzékeljen, markereket kell elhelyezni a szemüveg külsején, amit webkamera segítségével lehet detektálni és a látott képet feldolgozni. Az első verziójú szoftvercsomag, a szimulációs modell még nem tartalmazta a virtuális szemüveg megoldását, ez csak később került bele a gyakorlati alkalmazásba.

Hasonlóan a kesztyűhöz, a kinect kamera is egyszerű bemenetekkel lett helyettesítve a szimulációs modellben. A kinect kamera képes a kéz, a könyök és a fej abszolút pozíciójának érzékelésére egyedileg elkészített markerek alkalmazása nélkül is. Ezekről még később lesz szó, a szimulációs modell még nem tartalmazta a kinect-et, mint bemenetet, ehelyett billentyűzettel történő irányítás volt megvalósítva.

A robot program és a vezérlési program szimulációja is LabVIEW 2013-ben készült el 2014 nyara elején. Ez ugyan még nem tartalmazta az internetes kommunikációt, viszont annak kisebb- nagyobb késését képes valós időben szimulálni. A további fejlesztések mind erre a szimulációs modellre épülnek.



3. ábra: A szimulációs modell blokk-diagramja

Forrás: Saját készítés

2.3 Valós modell

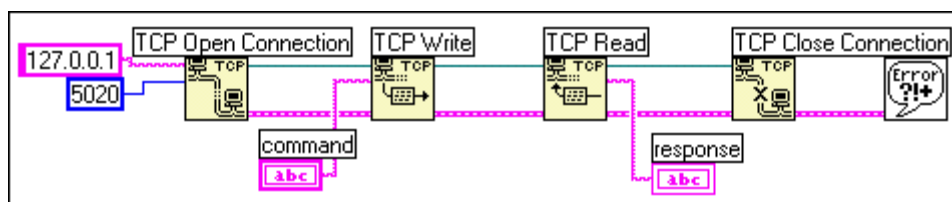
2.3.1 Kommunikációk

Miután a szoftvercsomag kezdeti modellje elkészült és stabil működést mutattak a számolások, a hiányzó – elsősorban kommunikációs és hardveres – elemek beillesztése következett. Ezek közül is a megvalósítás szempontjából az egyik legfontosabb elem az internetes kapcsolat és kommunikáció megbízható működésének leprogramozása. Ehhez a szimulációs modellt olyan módon kellett módosítani és tesztelni, hogy valóban különböző, operátor és robot oldalon egyéni felhasználói felülettel rendelkező programok keletkezzenek.

A valós modellben TCP/IP kapcsolat van megvalósítva az alapvető kommunikációhoz. Ezen a protokollon keresztül küldi és fogadja a kritikus adatokat a vezérlési program, mint például a valódi modell aktuális pozícióját, orientációját, vagy a csuklók koordinátáit. Ezeknek az adatoknak az átvitele fontos, hogy gyors és megbízható legyen akár az internetes kapcsolat ideiglenes megszakadása esetén is. Éppen ezért a kapcsolat

felépítése és újraépítése automatikusan történik, vagyis ha a robot oldalon kialakított szerver nem találja egy megadott időn belül a csatlakozni kívánt operátor oldalt, újból inicializálja az egész kapcsolatot. Mivel ez a folyamat gyorsan végbemegy, a felhasználók csak azt tapasztalják, hogy ha megszűnik ideiglenesen a két program között a kapcsolat, pár másodpercen belül automatikusan újra csatlakoznak egymáshoz.

A LabVIEW több megoldást is kínál a VI-ok közötti távoli kapcsolatra. Egy esetleges web szerveres megoldás helyett a beépített, kommunikációs protokollokra épülő eszköztár elemeiből raktuk össze a TCP/IP-nek megfelelő kapcsolatot, egyszerűsége és megbízhatósága miatt. Ezt indokolta a feladat jellege is, hiszen a valós idejű robotirányítás miatt kritikus szempont, hogy az adatsomagok hiánytalanul és hibamentesen célba érjenek. A Communication Toolboxból elérhető VI-ok segítségével az alábbi sémára épülő kapcsolat kiépítése vált lehetségessé:



4. ábra: Egyszerű TCP/IP kapcsolat LabVIEW alapon

Forrás: National Instruments

Amire figyelni kellett, hogy a fenti modellen túl nem mindegy, hogy a küldés-fogadás során milyen hosszú adatról van szó, ugyanis ha rövidebb adatot küldünk, mint amit a fogadónál beállítunk, akkor a kiolvasott és küldött adatok elcsúsznak egymáshoz képest. Megoldásként mindig elküldjük a küldendő adat hosszát, majd aztán magát az adatot, és fogadó oldalon a kiolvasott hossz alapján történik az adat megfelelő kiolvasása.

A kliens program „Searching” állapotban ciklusonként zárja a kapcsolatot, és megpróbálja megnyitni a megadott IP cím és port szám alapján. Amennyiben nem sikerül, ismétli a műveletet, sikeres csatlakozás esetén „Connected” státuszba állítja magát. Ebben az állapotban a kommunikáció folyamata írás (hossz), írás (adat), olvasás (hossz), olvasás (válasz). A küldött adat a robot csuklóinak pozíciója.

Szerver oldalon hasonlóan működik a kommunikáció. Az IP címet külső internetes oldalt felhasználó szubVI segítségével ellenőrizzük. „Searching” állapotban itt is lezárjuk a kapcsolatot, de ha a TCP/IP listen parancs beazonosítja a kliens kapcsolódását, átállunk „Connected” állapotba. Ezen az oldalon a kommunikációs folyam olvasás (hossz), olvasás (adat), írás (hossz), írás (válasz) lépésekből áll.

Az IP kamera képeit az operátor szintén interneten keresztül látja, viszont ennek a kommunikációnak kisebb a prioritása, mint egy vezérlő jelnek, amely közvetlenül avatkozik be a robotmozgásba. A kamerák által küldött adatokat az operátor oldal UDP segítségével kapja meg.

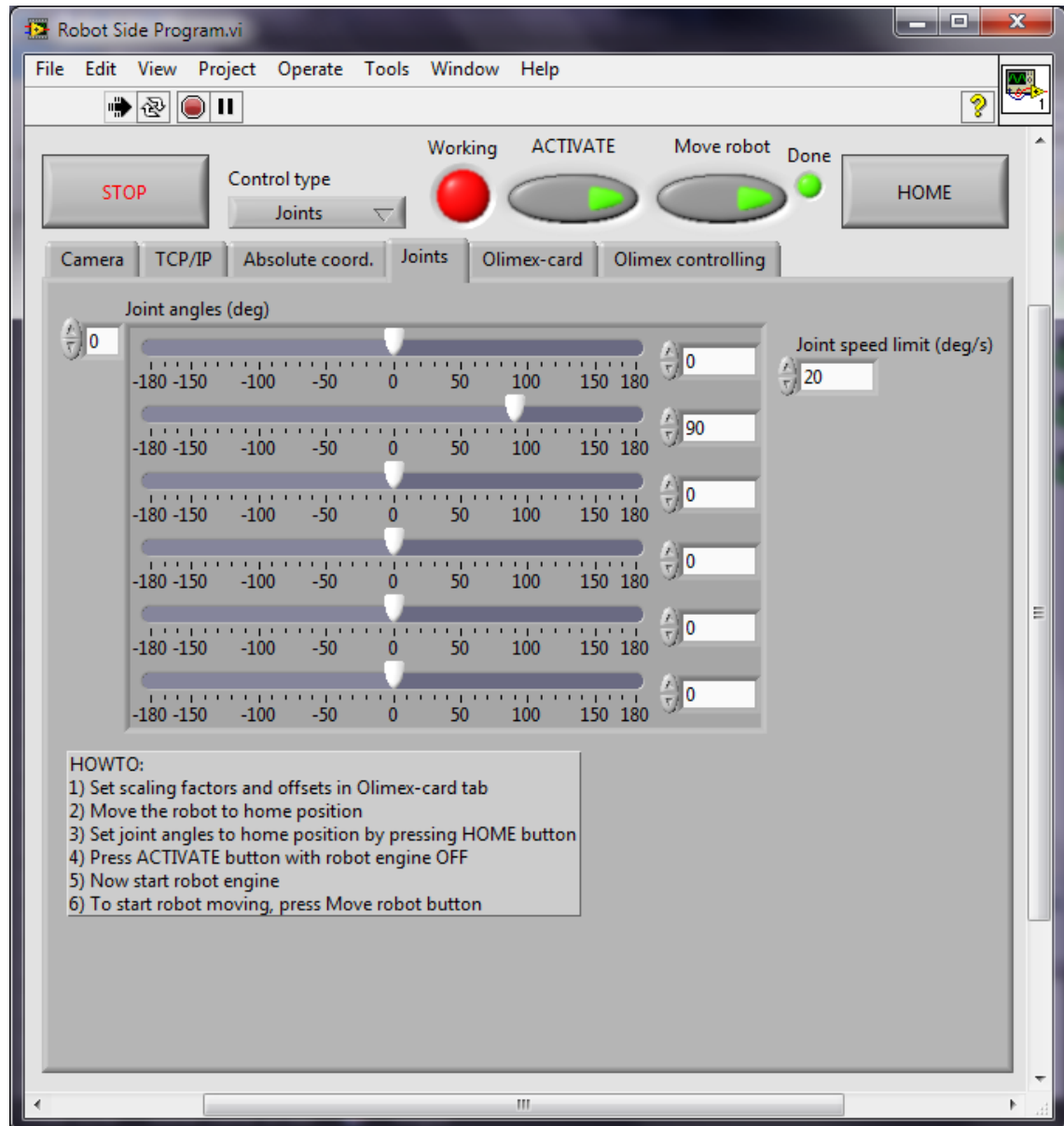
A robot oldali program a robot controller-el egy olimex kártyán keresztül kommunikál. Ez az egység a robot CPU-ból a szervo motoroknak kiadott kódolt adatokhoz és ugyanezen a vonalon visszafelé menő adatokhoz képes hozzáférni és értéket hozzáadni vagy kivonni. Ilyen módon lehetséges online programozni a robotot anélkül, hogy a controller szoftveresen letiltaná a hozzáférést. Mindössze a robot szervo motorjainak hibáit (például a túlságosan nagy ugrást két pozíció érték között, vagy túl nagy nyomaték fellépését) képes detektálni a controller, ilyenkor azonnal leállítja a robotot. Ezt az esetet nem csak a robot és a környezete biztonsága érdekében érdemes elkerülni, de automatizálás szempontjából sem előnyös, mivel egy ilyen hiba fellépésekor manuálisan kell újraindítani a robotot.

Esetünkben az olimex kártya helyett egy virtuálisan megvalósított program lett elhelyezve a controller és a robot között. Ezt a modellt két számítógép alkotta, egy a fogadott adatokhoz, egy a küldött adatokhoz. A virtuális kártya a robot programmal UDP kapcsolaton keresztül kommunikál, ezt a protokollt itt a nagy mennyiségű, nagy sebességű adatátvitel miatt alkalmazzák.

2.3.2 Robot program

A robot vezérlése közvetlenül a robot programon keresztül történik az olimex kártya segítségével. Ennek azért van jelentősége, mert ez a program végzi a robot mozgásához és biztonságához kapcsolódó számításokat. A program rendelkezik egy kezelői

felülettel, amelyen keresztül a felügyelő bármikor beavatkozhat az operátor és a robot kapcsolatába. Továbbá képes a felületen keresztül akár közvetlenül is irányítani a robotot.



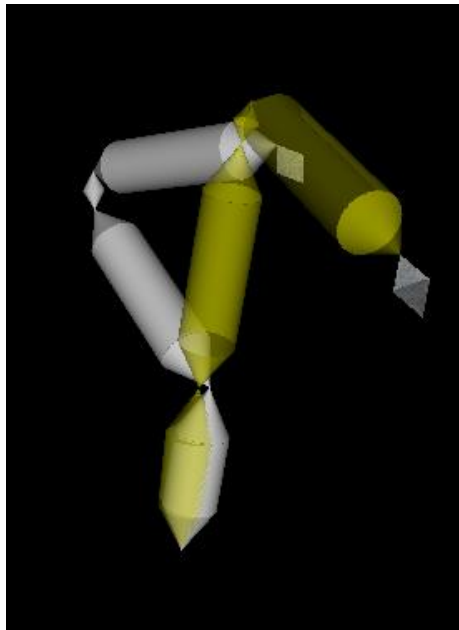
5. ábra: Robot program működése közben

Forrás: Saját készítés

A kezelőfelület több különböző ablakot is tartalmaz attól függően, hogy mire szeretnének használni a robot programot. Négy féle irányítás közül lehet kiválasztani egyet, amelyek sorban: olimex kártyára adatok közvetlen írása, csukló típusú adatok írása, abszolút pozíció típusú adatok írása (robot koordináta rendszerben), valamint TCP/IP-n keresztül fogadott adatok továbbítása. Ezen irányítási módok, mint modulok épülnek be a

programba, ezért a későbbiekben ezek tetszőleges módon kiegészíthetők. Típustól függetlenül mindegyik működéséhez szükség van az olimex kártya konfigurálására, hiszen minden esetben ezzel kommunikál végül a program. Ezeken a beállításokon túl további, irányítás specifikus paramétereket is lehet módosítani, mint például abszolút pozícióon keresztül történő irányításkor a lineáris interpolációval mozgó robot maximális sebességét, a csuklók forgásának maximális szögsebességét, vagy a TCP/IP kapcsolathoz használt portok számát is meg lehet adni.

Hasonlóan az operátor oldali programhoz, a robot program is képes megjeleníteni a robotot 3D grafika segítségével. Ezt a megjelenítést jelenleg a LabVIEW beépített funkciója biztosítja, amely segítségével a robot hat szabadságfokának változását nyomon lehet követni. A hatodik szabadságfok változását a fej textúrájának forgása érzékelteti.



6. ábra: A robot program 3D-s megjelenítése

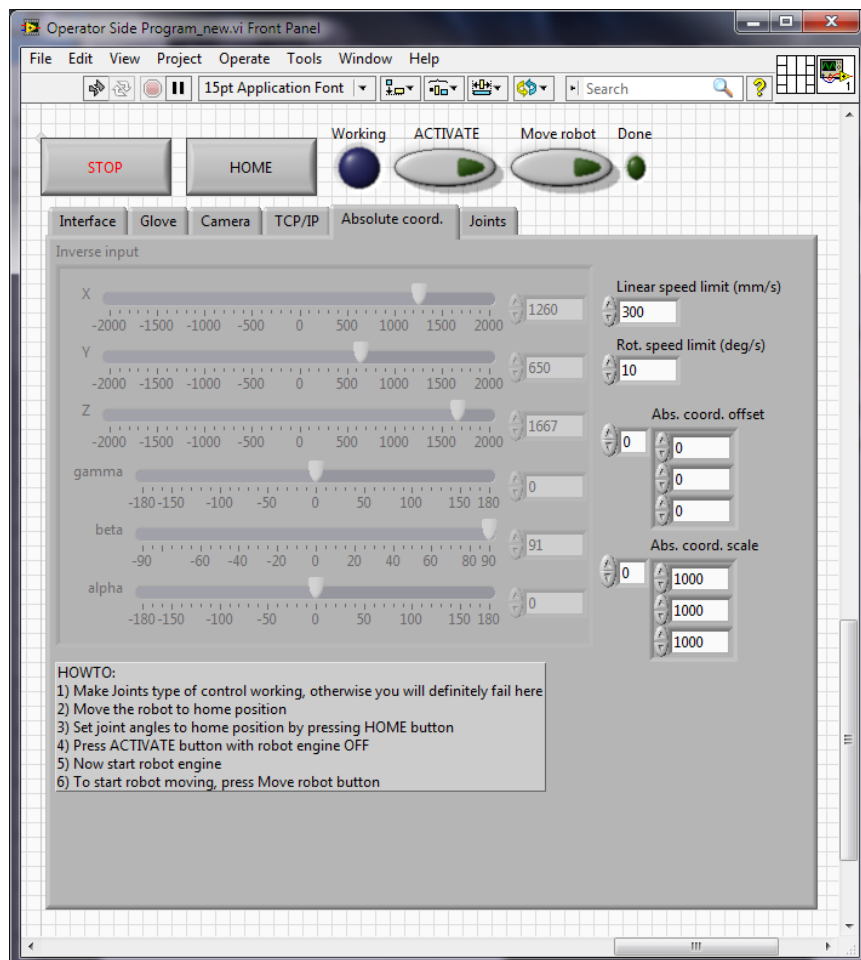
Forrás: saját készítés

A megjelenítő program tömör testként ábrázolja a robot aktuális pozícióját és áttetsző modellel jeleníti meg a cél helyzetet. A beállításoktól függően a robot csak egy adott sebességgel képes megközelíteni a célt, a mozgásának befejezésének jelzésére a programnak külön kimenet lett definiálva későbbi felhasználáshoz.

2.3.3 Vezérlési program

Az operátor oldalán a robot programhoz hasonló szoftvernek kell futnia, amely feladata elsősorban kommunikálni a robot programmal, valamint a hozzá illesztett perifériákat kezelni.

Kommunikációhoz a korábbiakban részletezett TCP/IP kapcsolatot használja. Ebbe a programba is be van építve egy 3D-s grafika megjelenítő modul, amely képes a valódi robot pozícióját és a kívánt pozíciót kirajzolni valós időben. Ezen kívül fontos szerepe van a programnak az egyes hardverek kezelésében, amelyről részletekbe menően a perifériák bemutatásánál lesz szó.



7. ábra: A vezérlési program felhasználói felülete

Forrás: saját készítés

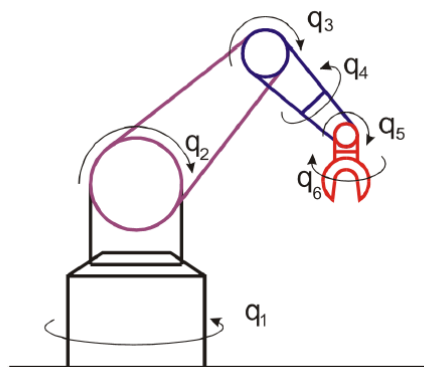
3 Kinematikai rendszer

Alapvetően szükséges a robot kinematikájával foglalkozni, hiszen csak ennek pontos ismeretével leszünk képesek a térben a kívánt pozícióba irányítani. Maga a kinematika leegyszerűsítve tulajdonképpen egy konverzió a különböző koordináták között, attól függően, hogy mit ismerünk és mire van szükségünk. A mi feladatunk egy hat szabadságfokú ipari robot esetén a geometriai paraméterek ismeretében mindkét rendszer felépítése.

3.1 Robot kinematika

A direkt, vagy idegen kifejezéssel forward kinematika célja, hogy egy kinematikai lánc elemein végighaladva meghatározzuk a szerszámközpont, vagy TCP (Tool center point) helyzetét. A kinematikai láncok lehetnek translációs és rotációs csuklókkal összekötve, esetünkben a robot csak rotációs csuklókkal rendelkezik, következésképpen a csuklókoordináták vektora, $\underline{q}$ hat elfordulási szögből áll. Ha ezek mellett ismerjük a lánc elemek hosszait, és a TCP egyértelműen meghatározható a világkoordináta rendszerben, melyet az x, y, z abszolút pozíció koordináták, és a φ, θ, Ψ (roll – csavarás, pitch – billenés, yaw – forgatás) szögek jelentenek. Tehát a feladat:

$$\underline{x} = f(\underline{q})$$



8. ábra: A kinematikai feladat
Forrás: (2)

Egyetlen lánc elemre a transzformáció a következőképpen néz ki:

$$T_i^{i+1} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & d_{i,x} \\ R_{21} & R_{22} & R_{23} & d_{i,y} \\ R_{31} & R_{32} & R_{33} & d_{i,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_\alpha & -c_\beta s_\alpha & s_\beta s_\alpha & Lc_\alpha \\ s_\alpha & c_\beta c_\alpha & -s_\beta c_\alpha & Ls_\alpha \\ 0 & s_\beta & c_\beta & D \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ahol α a hajlás, β az elcsavarodás szögei, L az elem hossza, D pedig egy offszet távolság a forgástengelyen. c_α és s_α a szögek koszinuszait, szinuszeit jelölik. A teljes lánc ezek alapján:

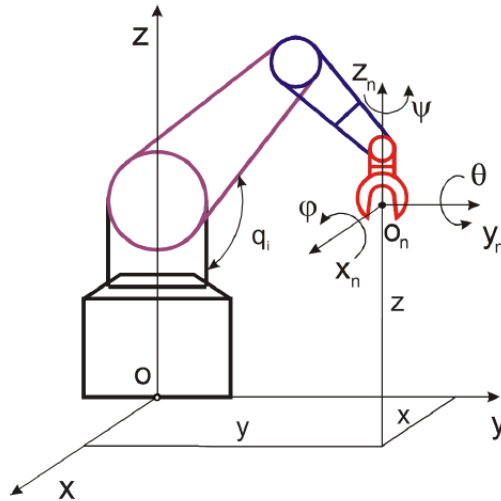
$$T = B(\underline{x}_0) T_1^2(q_1) T_2^3(q_2) T_3^4(q_3) T_4^5(q_4) T_5^6(q_5) T_6^{TCP}(q_6)$$

A teljes transzformációból ezek után kiolvasható az eredő eltolási vektor és eredő forgatási mátrix, amiből kiszámíthatóak az eredő világkoordináták. Természetesen az alkalmazásoknak megfelelő szerszám felhelyezése egy, a szerszámot jellemző további transzformációval hosszabbítja meg a fenti láncot.

3.2 Robot inverz kinematika

Az irányításhoz persze nem elegendő, hogy a csuklók (szervó motorok) adott szögei esetére ismerjük az abszolút pozíciót, hiszen a felmerülő feladatokban csak világkoordinátákkal tudjuk megadni a robotnak, hova szeretnénk irányítani és ekkor a csuklókoordináták a keresett ismeretlenek. Az inverz feladat ennek megfelelően:

$$f^{-1}(\underline{x}) = \underline{q}$$



9. ábra: Az inverz kinematikai feladat
Forrás: (2)

Hat szabadságfok esetén e feladat is egyértelmű, de az analitikus megoldás problematikusága miatt a legtöbb esetben numerikus módszerekkel szokás eredményhez jutni.

A kiindulási ötlet, mivel f nem-lineáris függvény (a szögfüggvények miatt), hogy keressük f^{-1} lineáris approximációját. Így jutunk el a Jacobi mátrixhoz.

$$\underline{x} = \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} f_1(\underline{q}) \\ f_2(\underline{q}) \\ f_3(\underline{q}) \\ f_4(\underline{q}) \\ f_5(\underline{q}) \\ f_6(\underline{q}) \end{bmatrix}$$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \dots & \frac{\partial f_1}{\partial q_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial q_1} & \dots & \frac{\partial f_6}{\partial q_6} \end{bmatrix}_{6 \times 6}$$

Ennek használata, hogy a Jacobi mátrixot kiértékeljük valamilyen $\underline{\hat{q}}$ helyen, ez lehet a robot aktuális pozíciója csuklókoordinátákkal. Ezt követően élünk a következő közelítéssel:

$$\underline{x}(\underline{q}) \approx \underline{\hat{x}}(\underline{\hat{q}}) + \underline{J}(\underline{\hat{q}}) \cdot (\underline{q} - \underline{\hat{q}})$$

Más szavakkal $\underline{J}$ azt mondja meg, $\underline{x}$ hogyan fog megváltozni, ha megváltoztatjuk $\underline{q}$ egyes elemeit. Átrendezve az egyenletet:

$$\Delta \underline{x} \approx \underline{J}(\underline{\hat{q}}) \Delta \underline{q}$$

Kis elmozdulásokra ez az összefüggés jó közelítés, így láthatóan:

$$\Delta \underline{q} \approx \underline{J}(\underline{\hat{q}})^{-1} \Delta \underline{x}$$

Mivel az inverz számítás ekkora (vagy nagyobb) mátrixok esetén már komoly erőforrást igényel, gyakran kerülő módszerekkel szokás eredményhez jutni. Ilyenek például a Jacobi-transzponált módszer, melyben az elvégzett munka elvére alapozva elegendővé válik a transzponált mátrix számítása (jóval egyszerűbb számítási művelet), és az alapján következtetünk a pozícióra, illetve ilyen még a gömbkoordináta-származtatás módszere is, melyben a feladatot felbontjuk 1 szabadságfokú problémák sorozatára, melyek szintén könnyen számíthatóak.

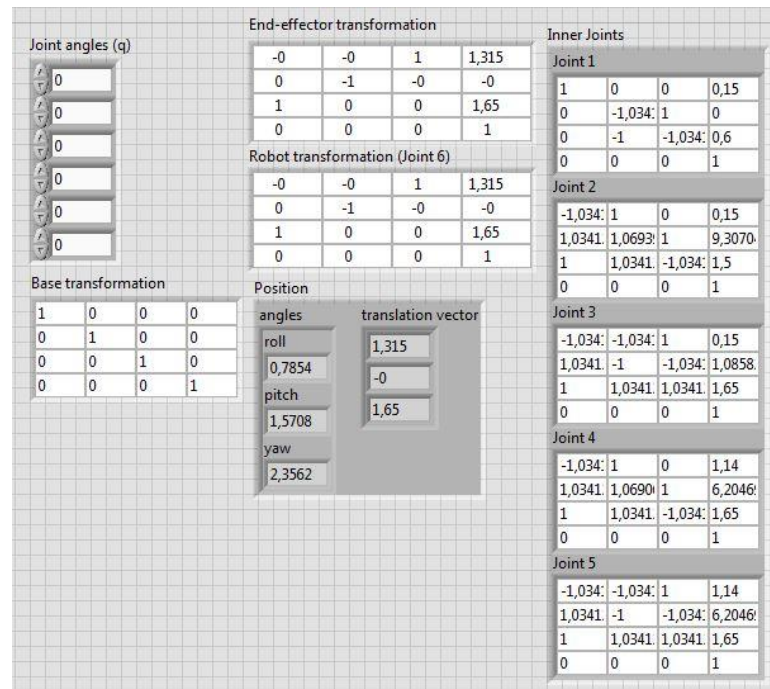
Minden módszernek vannak előnyei és hátrányai, ebben a feladatban a megoldást a LabVIEW kiterjedt beépített funkciókönyvtárában található, a lineáris egyenletrendszereket megoldó VI jelentette. Ez lényegében egyezik az Jacobi-inverz számításának módszerével. (3) (4)

3.3 Implementálás LabVIEW segítségével

A LabVIEW program bemenet-kimenet jellegű szemlélete nagyban megkönnyítette a feladat szimulációját. Ezzel az eszközzel a programunkat modulárisan tudtuk felépíteni, ami átláthatóvá tette a rendszert és lehetővé tette a hatékony párhuzamos munkavégzést.

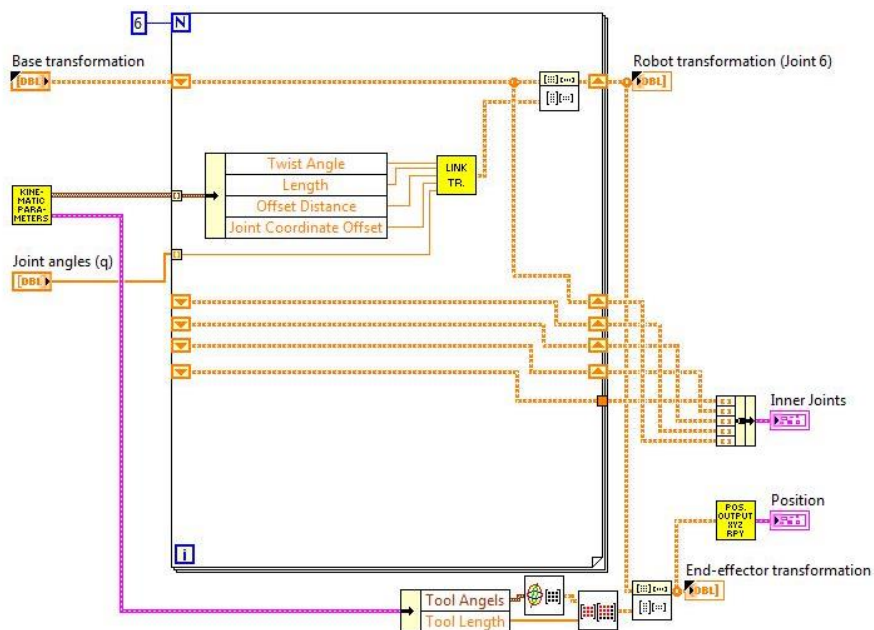
3.3.1 Forward kinematics

Ez a VI a fent említettek szerint a bemenő csuklókoordinátákat (kiegészítve a bázis transzformációval és a szerszám transzformációval) a számításhoz hasznos transzformációkká és a könnyebben kiolvasható pozíció adatokká alakítja.



10. ábra: Direkt kinematika - front panel

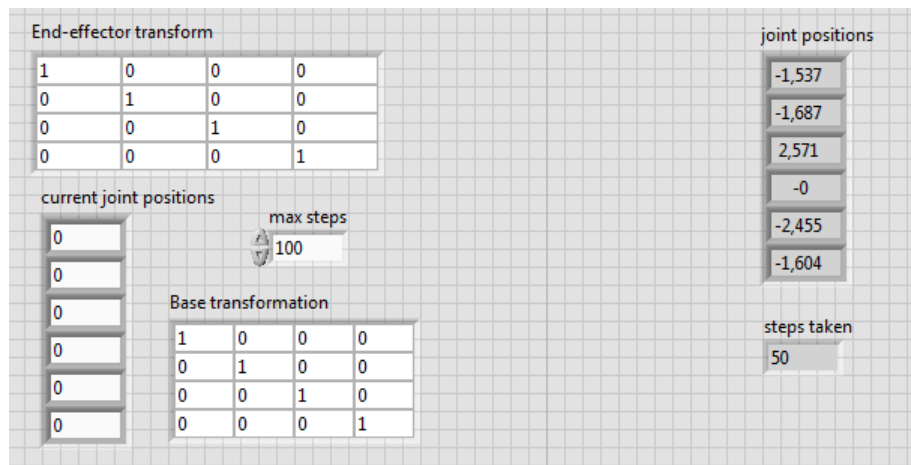
Forrás: saját készítés



11. ábra: Direkt kinematika - blokk diagram

Forrás: saját készítés

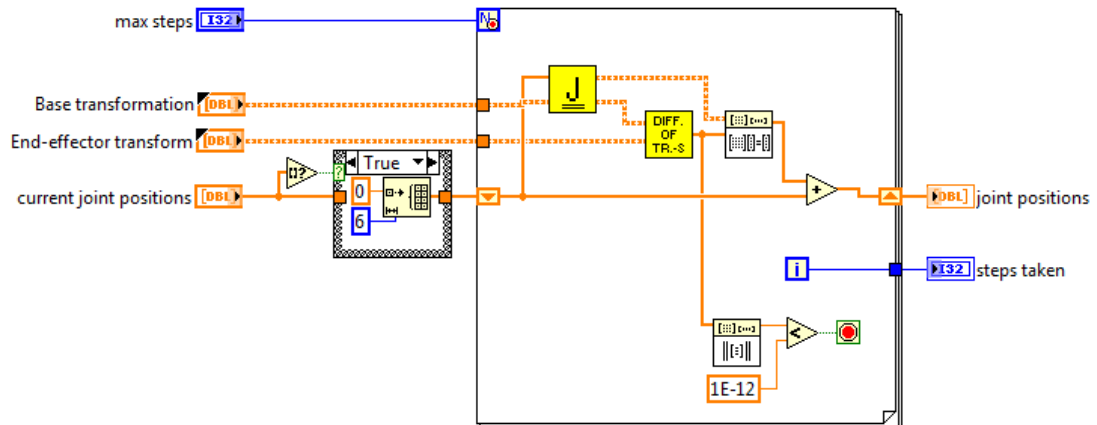
3.3.2 Inverse kinematics



12. ábra: Inverz kinematika - front panel

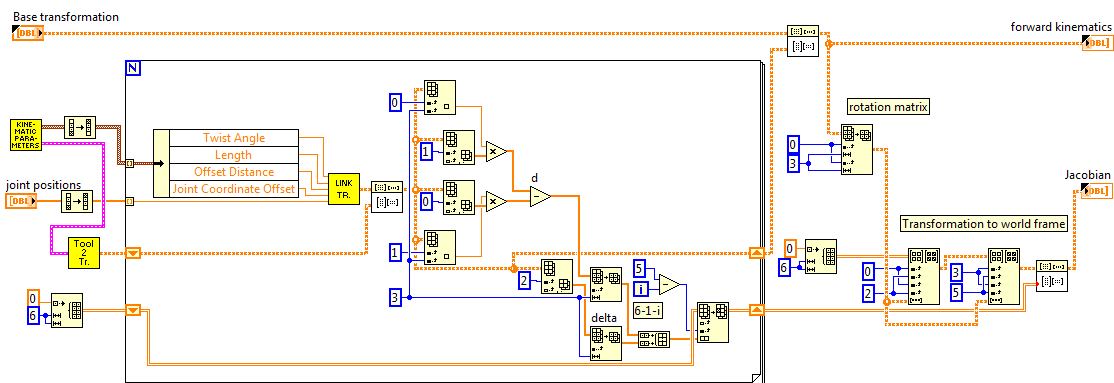
Forrás: saját készítés

Ez a jelenlegi pozíció ($\underline{\hat{q}}$) és a kívánt pozíció ($\underline{x}$) alapján kiszámítja a Jacobi mátrixot, és azzal a szükséges új csuklókoordinátákat ($\underline{q}$). Az eredményeket iteratív módon közelíti az ideális megoldáshoz, és akkor áll meg, ha kellő pontosságot elért, vagy ha az iterációs időkorlátot felhasználta.



13. ábra: Inverz kinematika - blokk diagram

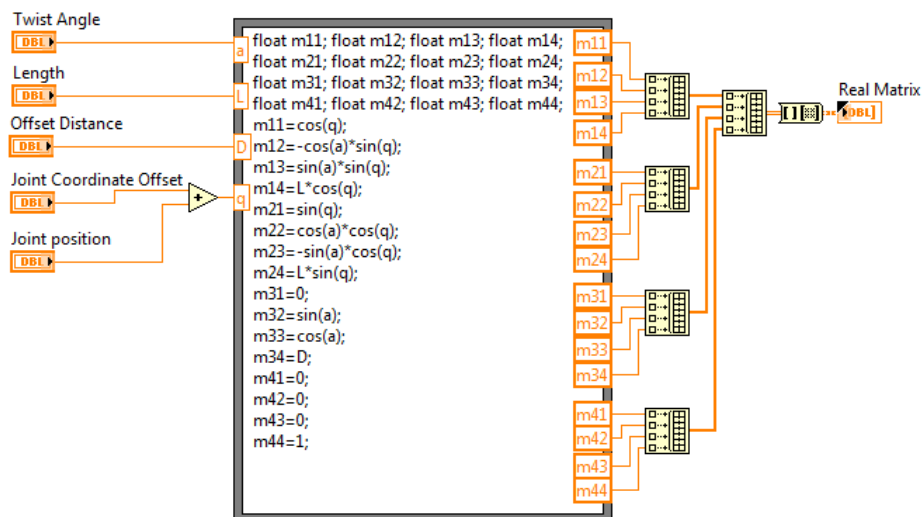
Forrás: saját készítés



14. ábra: Jacobi mátrix számítása

Forrás: saját készítés

A Jacobi mátrixot a fenti blokkdiagram alapján számítottam ki. A felhasznált link transzformáció a kinematikai lánc egyes transzformációit számítja ki, az alábbi elven:



15. ábra Link transzformáció

Forrás: saját készítés

4 Kinect szenzor

4.1 A szenzor működése

A szenzor egy RGB kamerával, hangérzékelővel, egy mélységtérképet szolgáltató IR kamerával és chippel van ellátva. Az integrált megoldásnak köszönhetően egyszerűen lehet objektumokat azonosítani a 3D térben, amely lehetővé teszi számunkra kitüntetett térbeli pontok felismerését és követését.



16. ábra: Microsoft Kinect szenzor

Forrás: (5)

A Kinect szenzor alapja az IR kamera együttes, mely egy infravörös fényt kibocsájtó fényforrásból és egy infravörös fényt érzékelő kamerából áll. Az eszközben található mélységérzékelő chip a szokásostól eltérő módon állítja elő a mélységi adatokat. Az IR kibocsátó egy sűrű ponthalmazt vetít ki a megfigyelendő objektumra. A kivetített mintázatokat először ismert, de különböző távolságban lévő referencia felületekre vetítik ki, majd ezeket az képeket, és a mintázatokban lévő pontok távolságát elmentik referencia adatokként. Az ismeretlen távolságban lévő objektumra ezután ugyanazt a mintázatot vetítik ki. Az előzőleg referencia felületekre kivetített mintázatok közül, kiválasztják a legvalószínűbb referencia felületet, és az ahhoz eltárolt referenciapontok távolságát. Ezután az ismeretlen objektumra kivetített mintázat pontjait kereszt korrelációs technikákkal megfeleltetik a referencia felület pontjainak, és a pontok távolságából meghatározható a referencia felület és az ismeretlen objektum felületének z tengely (3. dimenzió) menti különbsége. (6)

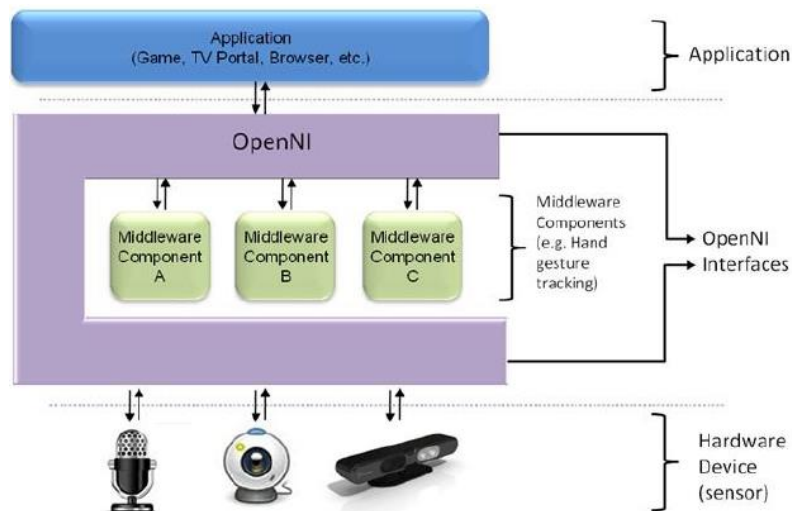
A Kinect szenzor precizitását tekintve alulmarad más, elterjedten használt mozgáskövetési rendszerrel összehasonlítva, mint például a több kamerás markeres rendszerek, azonban bizonyos ipari feladatokra kiválóan alkalmas, megfelelő

szoftverekkel használható telemanipulációs feladatok input eszközeként. További előnye, hogy az integrált kamerarendszeren kívül nem igényel további eszközöket, markereket, így magában hordozza az NI elv előnyeit is, kényelmes és intuitív használatot eredményezve.

4.2 Middleware

Az NI (Natural Interaction) egy olyan elgondolás, amely az ember-gép interakciót a látásra és a hallásra korlátozza. Az NI elvek alapján tehát az olyan eszközök, mint a távirányító, egér vagy billentyűzet szükségtelenek az ember-gép kapcsolat megteremtéséhez, továbbá ezen eszközök használatának eliminálása sokkal intuitívabb felhasználást eredményez. Tipikus kutatási területei közé tartozik a hangfelismerés, hangvezérlés, kézjelek felismerése, alakfelismerés, illetve az előbbieket implementálása a mindennapi életbe, ipari felhasználásba. (6) Néhány, a hétköznapiakból ismert példa lehet a Natural User Interface-re a telefonok és tabletek esetén alkalmazott érintőképernyő vagy hangvezérlés. Ezen kívül a szórakoztató iparban széles körben elterjedt mozgáskövetésen alapuló játékok, mint a Nintendo Wii és a Microsoft Kinect is hasonló elvekre épít.

Az OpenNI egy cross-platform, több programozási nyelven elérhető framework, ami API-kat szolgáltat az olyan alkalmazások készítéséhez, amelyek az NI alapelvekre épülnek. Az API-kat interfészekon keresztül lehet elérni, amik azt a célt szolgálják, hogy egységes felületen lehessen kommunikálni a hardver elemekkel, és a szenzoradatokat feldolgozó middleware-rel. A middleware elemek olyan szoftverkomponensek, amelyek a szenzorból érkező adatokat feldolgozzák, és olyan magasabb szintű információkat szolgáltatnak a nyers adatokon felül, mint például egy felismert kézfej pozíciója a 3D térben. (7)



17. ábra: OpenNI platform
Forrás: (7)

Az általunk felhasznált ingyenesen elérhető program a Leeds University által fejlesztett Kinesthesia, amely lehetővé teszi a Kinect szenzorból származó RGB, mélységi, és akusztikus információk felhasználását LabView környezetben, illetve a nyers adatok mellett alkalmas szkeleton kirajzolására és az ízületek mozgásának követésére is. (8)



18. ábra: A Kinesthesia motion tracking rendszere
Forrás: (9)

Az ízületek 3D adataiból kinyertük a számunkra fontos csukló, könyök és váll térbeli pozíció-adatokat, kísérleteket végeztünk a jelek felhasználhatóságával kapcsolatban, illetve a simítás és szűrés után kidolgoztunk egy olyan parancsrendszert, amellyel a

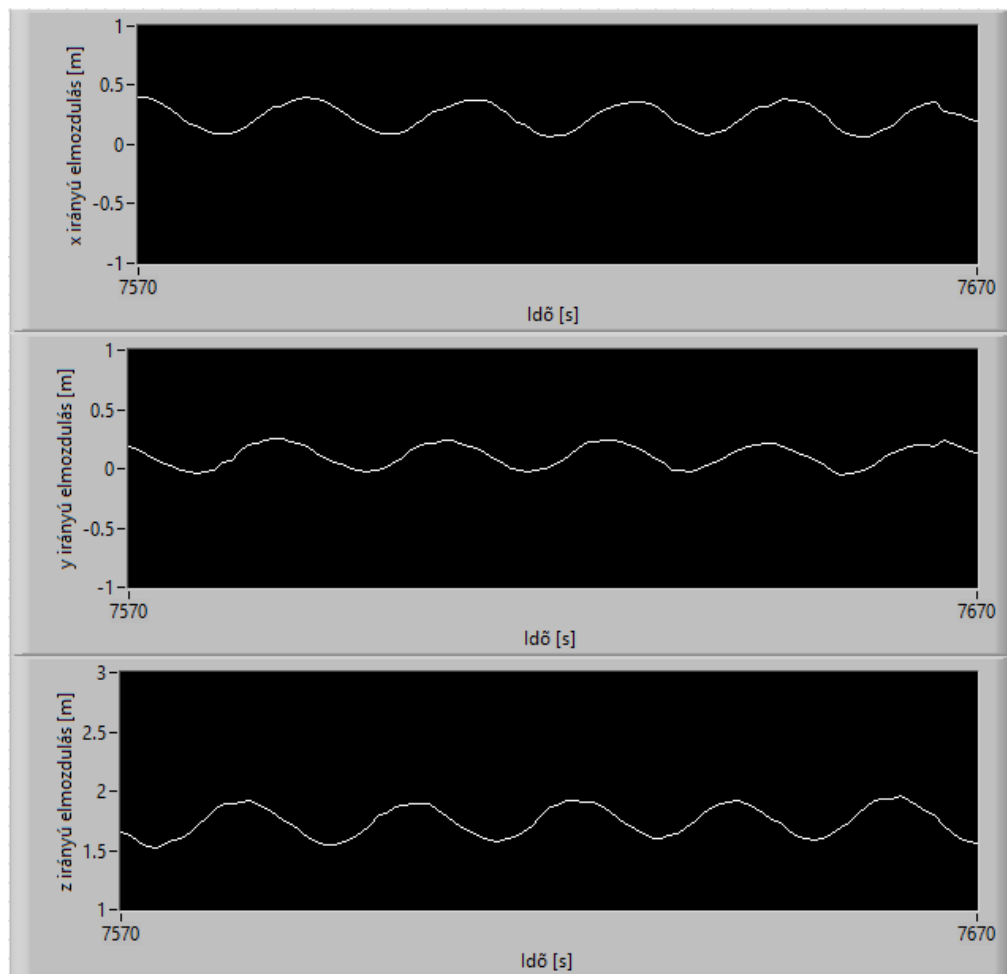
kapott adatok minőségét figyelme véve a távoli vezérlés egyszerűen és megbízhatóan megvalósítható, egyféle alternatívaként a billentyűzeten történő vezérlés mellett

4.3 A Kinect szenzorból származó jelek feldolgozása

4.3.1 Követelmények

Az általunk kidolgozott – és későbbiekben bemutatásra kerülő – parancsrendszer bemenő adatként az operátor egyik karjának csukló-, könyök- és vállízületének térbeli pozícióját igényli. Az adatokkal szemben támasztott legfontosabb követelmény, hogy azok valós időben, meglehetősen kis késleltetéssel rendelkezésre álljanak. Ez azért fontos, mert a nagy késleltetés kevésbé megbízható irányítást eredményez. Előfordulhatnak nem szándékos túlmozgások, túllövések, hiszen az operátor csak később veszi észre a problémát, mint ahogy az felmerült, tehát a vezérlő jeleket már a probléma észlelése előtt kiadta. Emellett természetesen fontos tényező a viszonylagos pontosság is, ez azonban a gyorsasággal ellentétes követelmény, így ilyen értelemben másodlagos szerepet kap a gyors működés mellett. Ez az állítás a jelfeldolgozás során kapott nagyobb hangsúlyt. A probléma úgy szemléltethető, hogy amennyiben a szenzor által szolgáltatott jel kis mértékben eltér az ízületek abszolút térbeli helyzetétől, az operátor további pótlólagos mozgásokkal a robotkart a kívánt helyzetre tudja irányítani. Ilyen értelemben a szenzor hibája korrigálható. Ebből következik, hogy finompozicionálást igénylő feladatok kivitelezésénél elengedhetetlen, hogy a jel stabil legyen, vagyis ne ugráljon az időben. Ha a zajokból kifolyólag a hibatag véletlenszerű nagyságú és előjelű, akkor gondoskodnunk kell a megfelelő zajszűrésről. Ez a jelenség nem csupán a külső zaj megjelenésével magyarázható, hanem az emberi mozgás tökéletlenségeivel is. Előfordulhat kézremegés, amelyet lekövetve egy pontos szenzor is ugráló jelképet ad eredményül.

A jelfeldolgozás során a Kinectből származó, csuklóra vonatkozó térbeli információkat teszteltük. Az adatokat az idő függvényében ábrázolva az alábbi ábrát kaptuk. Először mindhárom irányba megvizsgáltuk a kapott jeleket, hogy meggyőződjünk róla, az RGB kamera által kapott jelek és a mélységi jelek között nincsen lényeges különbség.



19. ábra: A Kinesthesia feldolgozatlan jelei

Forrás: saját készítés

Látható, hogy az érzékelés hasonló jósági tényezőkkel rendelkezik mindhárom irányban, ezért a továbbiakban csak egy kitüntetett, x irányban végzünk kísérleteket a jelszűrésre vonatkozóan.

4.3.2 Jelszűrés, simítás

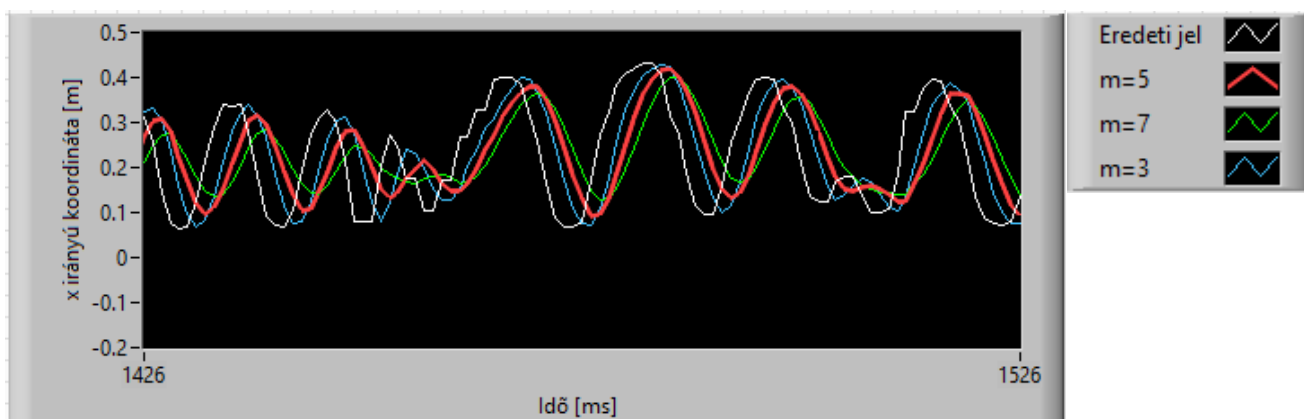
A simító, szűrő eljárások között az egyik leggyakrabban alkalmazott módszer a mozgó átlagolás. Lényege, hogy az időfüggvény egy t pontbeli függvényértékét a t pont környezetében lévő elemek függvényértékeinek átlagával közelíti. Legegyszerűbb esetben, ha csak a t -edik függvényértéket megelőző és – szimmetrikusan – követő elemek egyszerű átlagát vesszük, azonos súlyozással, akkor az egyszerű mozgó átlagot kapjuk.

$$\hat{y}_t = \frac{y_{t-n} + y_{t-(n-1)} + \dots + y_t + \dots + y_{t+n}}{2n+1} \quad (1)$$

$$\hat{y}_t = \frac{\frac{1}{2}y_{t-n} + y_{t-(n-1)} + \dots + y_t + \dots + \frac{1}{2}y_{t+n}}{2n} \quad (2)$$

Az (1) és (2) formulák a lehető legegyszerűbb mozgó átlagolású adatsort eredményezik. Az első egyenletben $m = 2n + 1$ páratlan, míg a második egyenletben m páros, ennek megfelelően az aktuális elem kiszámítása különbözik. (10) Ha a rendszerről semmi további lényeges információnk nincsen, akkor indulásként ennek használata célszerű, hiszen ez a lehető legegyszerűbb módon simítja ki az időfüggvényt.

A feladat m optimális meghatározása. Az átlagszámítással csökkentjük a véletlen tag, másként zaj szerepét, ugyanakkor a t mozgatásával biztosítjuk az alaptendencia megmaradását. Tudjuk azonban, hogy m növelésével – ahogy egyre több elemből számolunk átlagot – egyre csökken a hiba szerepe, azonban egyre nagyobb lesz a késleltetés, hiszen a mindenkor pillanatban számolt érték n mintavételezéssel később kerül kiszámításra. Továbbá a túl nagy m értékkel a mozgásról is értékes információkat veszíthetünk. Az alábbi grafikonokon az eredeti értékekhez viszonyítva jelenítettük meg a simított jelet, három, öt és hét tagú mozgó átlagolást alkalmazva.



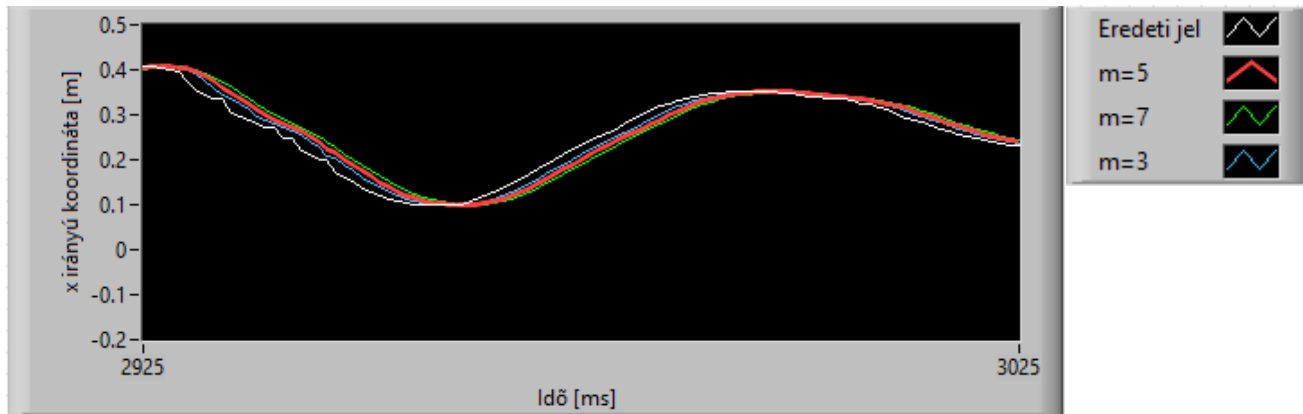
20. ábra: Mozgó átlagolás

Forrás: saját készítés

Az fenti grafikonon olyan jelet generáltunk, amelynek a változása meglehetősen gyors és sok zajjal terhelt. Tehát gyors kézmozgás és imitált kézremegés mellett rögzítettük a pozícióértékeket. Megvizsgálva a három szűrési típust megállapíthatjuk, hogy a három

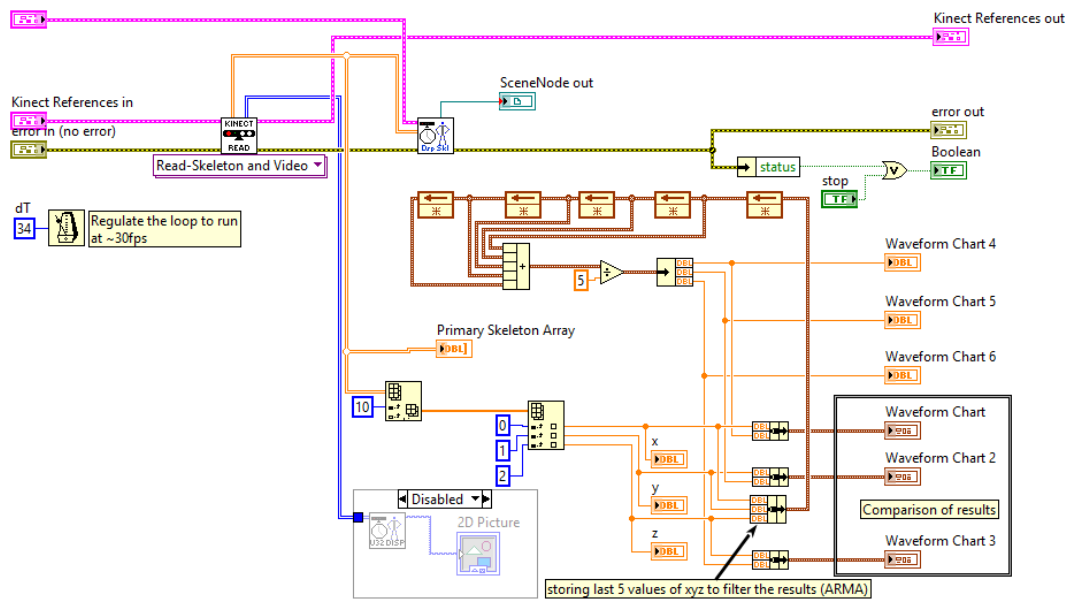
tagú mozgó átlag az élesebb változások esetében nem reagál megfelelően. Ez azt jelenti, hogy a három tagú átlagolásnál esetleg előfordulhat, hogy egy hirtelen, véletlenből származó kézmozdulatot követ az irányítandó robotkar, ami nem megengedhető, a hét tagú átlagolás azonban már egy szinte tökéletes, sima görbét eredményez. A késleltetések között viszonylag nagy különbségek vannak, ez további megfontolásokat kíván.

A következő grafikonon egy valósághoz sokkal közelebbi mozgást szemléltet. Ez alatt értendő, hogy a valós ipari körülmények között egy robotkar kézi irányításakor kisebb sebességekre lehet számítani, mint az előbb demonstrált mozgás.



21. ábra: Jelsimítás
Forrás: saját készítés

Látható, hogy ebben az esetben a késleltetésnek nincs akkora hatása a valós és számított értékek különbségére. Mindezek figyelembevételével a választásunk az öttagú mozgó átlagolásra esett, melynek megvalósítása LabView környezetben alább látható. A programrészlet a főprogram While ciklusában helyezkedik el.

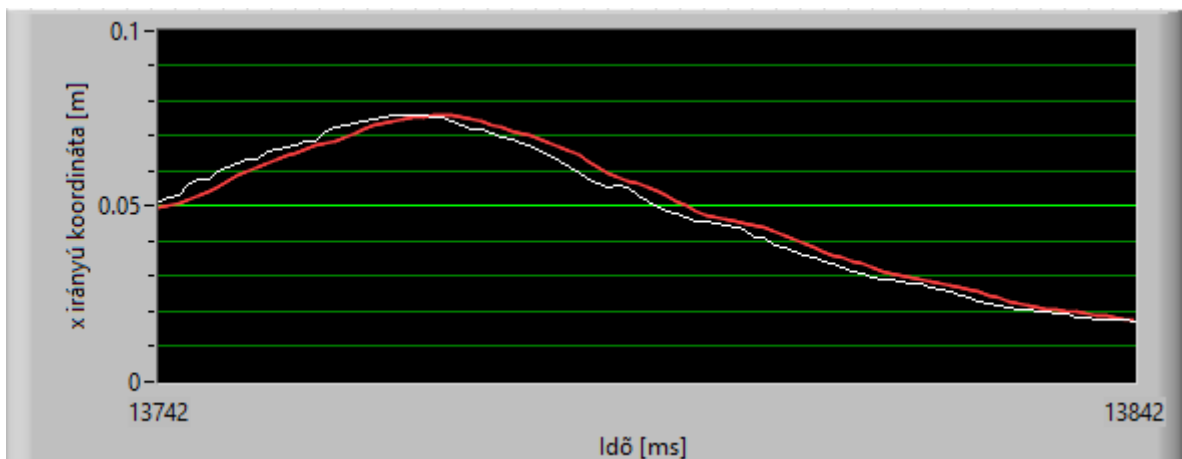


22. ábra: A jelszűrés programrészlete

Forrás: saját készítés

4.3.3 Jósági paraméterek

A szűrés után lehetőség nyílik arra, hogy számszerűen is meghatározzuk a késleltetés és pontosság mértékét. A Kinect programja 30 fps-sel fut, tehát közelítőleg 33 ms telik el két mintavételezés között, így a késleltetés $2 \cdot 33 = 66$ ms.

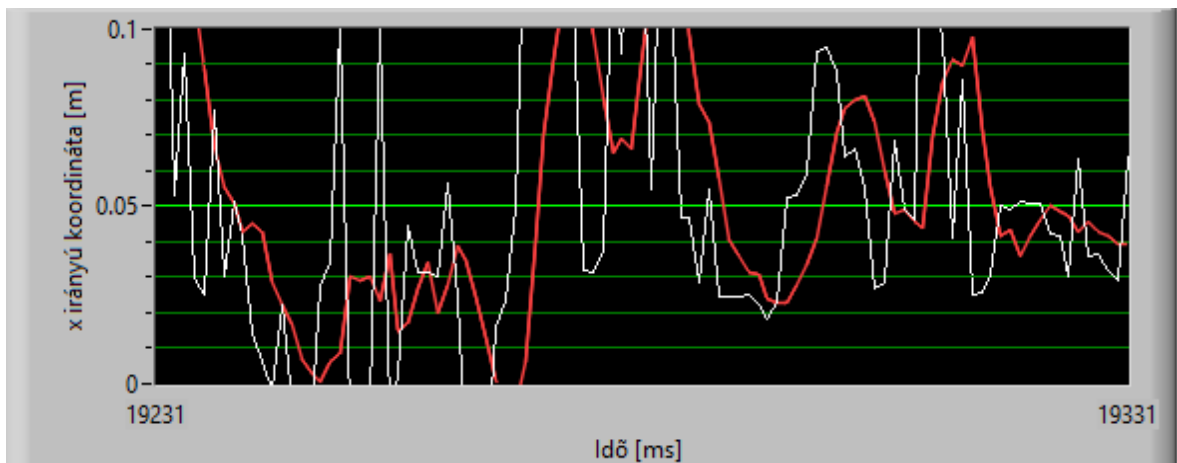


23. ábra: Jelkövetés

Forrás: saját készítés

A pontosságot olyan értelemben használjuk, hogy mennyire pontos a jelkövetés a szűrés után, amely kielégítőnek mutatkozik, lévén közepes sebességű mozgás esetén az adott pillanatbeli értékek különbsége fél centiméteres sávban mozog.

Itt érdemes megjegyezni az általunk használt Kinect szenzor egyik nagy hibáját, miszerint akkor is ad eredményt – tehát egy meghatározott pozícióértéket – amikor bizonyos ízületi pontoknak nem érzékeli a jelenlétét. Tehát takarásban, nem megfelelő elhelyezésben. Ilyenkor viszont a szenzorból érkező jelek extrém módon oszcillálnak, ami használhatatlan input adatokat eredményez. Ez részben azzal magyarázható, hogy a Microsoft Kinect szórakoztatási céllal készült, ahol sokkal fontosabb a folyamatos üzem, mint az adekvát működés. A problémát a jelen feladatban a későbbiekben említésre kerülő sebességkorlátozással oldottuk meg, hiszen ez az üzemállapot a legkönnyebben a jelszintek nagyon gyors változásával ismerhető fel legkönnyebben.



24. ábra: Szabálytalan üzemállapot

Forrás: saját készítés

4.3.4 Orientáció kezelése

A jobb kéz koordinátáit felhasználva a robotpozíció koordináták irányítása szemléletesen megfeleltethető a kéz pozíció koordináta rendszerének. Az orientáció irányítása azonban felvet néhány kérdést. Ugyan a robot és az emberi kéz analógiájából kiindulva felmerülhet a lehetőség, hogy az orientáció-irányítási üzemmódban a szögeket a kéz csuklójának iránya adja, tehát amerre az emberi ujjak mutatnak. Azonban ezt a

lehetőséget a kézfej változatos helyzete, és a Kinect által kevésbé megbízható feldolgozás miatt elvetettük, helyette az elfordulási szögeket az alkar irányával definiáltuk. Ez még mindig elég szemléletes ahhoz, hogy természetes, ösztönös mozdulatokkal rögtön a megfelelő irányítást ériük el egy tanuló operátornál, és emellett a stabilitás szempontjából is jóval megbízhatóbb eredményt ad.

A számítást a csontváz-modell két megfelelő pontja, a könyök és a csukló közé húzott vektor iránya adja. Gömbkoordináta rendszerbe váltva a Descartes koordinátákat egyértelműen és egyszerűen kiadódik a két szög, melyet e módszerrel irányíthatunk. A harmadik szöget ezzel a módszerrel nem lehet befolyásolni, hiszen a rúd-modell saját tengelye körüli forgás a Kinesthesia rendszerében nem megoldott. Ennek következtében ezt a szöveget alternatív módokon lehetne irányítani, jelen helyzetben meghagytuk a számítógépes gombokkal való irányításának, hiszen a fizikai objektumok manipulálása és az erőszabályozás bevezetése előtt ritkán fordul elő a vele szemben támasztott orientációs követelmény.

5 A vezérlés

A vezérlő program tartalmazza mindazokat a funkciókat, amelyek szükségesek a robotkar megfelelő irányításához. Az alapkoncepció szerint a vezérlő kesztyű – tehát az ennek megfelelő szimulációs paraméterek – a kéz pozíciója és orientációja egyértelműen meghatározza a robotkar szerszámának mozgását. Az ujjak helyzetének a mozgítás szempontjából nincsen jelentősége, mivel azonban a hardver lehetővé teszi az ujjak állapotának detektálását is, ezt kihasználva lehetőségünk van a különböző engedélyezett parancsokat a különböző ujj-kombinációkhoz kötni. Ezzel egy jóval egyszerűbb és biztonságosabb vezérlés valósítható meg.

5.1 A vezérlő program részei

A megvalósított funkciókat az előbbieknél megfelelően az ujjak helyzetéhez kapcsoljuk. Öt nyúlásmérő bélyeg, tehát analóg szenzorok szolgáltatják a behajlításból származó jeleket. Ennek megfelelően elvileg a behajlítósság mértékét is detektálni lehet, azonban a kesztyű nem pontosan mozog együtt az ujjakkal, a jelek feldolgozása is komplikált lenne, ezért egy komparátor segítségével az információkat digitális jelekké alakítjuk. Ez azt jelenti, hogy a vezérlés szempontjából csak az a bit információval, hogy az ujjak behajlított vagy kinyújtott állapotban vannak. Mivel ezek feszültségtartományok így szélesebb tartományban lehetnek, sokkal kisebb a valószínűsége a téves parancsoknak kiadásának. A komparálási szint beállításánál azt vettük figyelembe, hogy az ujjak természetes helyzetükben is kissé be vannak hajlítva, a referencijel viszonylag magas értékre állításával biztosíthatjuk, hogy a program csak valóban szándékos ujjbehajlításokat érzékeljen parancsként.

Elviekben minden ujj 0 és 1 helyzetét mérni tudjuk, ezért a lehetséges parancsok, funkciók száma $2^5=32$. A feladat során mi három főfunkciót definiáltunk, ennyi elég a robotkar mozgásához, de természetesen a szabályozás során újabb funkciók is szükségesek lehetnek, illetve a robotkar felhasználási módjából származhatnak további igények.

A program természetesen lehetővé teszi, hogy kesztyű hiányában a kezelőfelületen is lehetőség nyíljon az ujj behajlítottságának megfelelő parancsokat kiadni, ebben az esetben a segítségünkre van egy visszajelző ablak, amely megmutatja az éppen aktuális állapotot.

1. funkció

Az első funkció biztonsági célokat szolgál. A kéz kinyitott állapotában a robotkar nem kap parancsokat, tehát az elindításhoz szükséges az aktív közreműködés, a robot „megfogása”. Emellett a leállításra is igaz, hogy különböző nyomógombok és bonyolult vészleállítási protokollok helyett elegendő az ujjak felengedése, és a kommunikációs kapcsolat megszűnik. Ez az állapot van megadva a program default értékének is, tehát ha nem értelmezett kombinációban tartja az operátor az ujjait, nem történik semmi.

2. funkció

A második funkció szerint minden ujjat szükséges behajlítani, ekkor az ökölfogás hatására létrejön a kommunikációs kapcsolat a kesztyű és a robotkar között, a robot a kéz irányításával mozgásba kezd. A vezérlés a csukló abszolút térbeli koordinátáit érzékeli, amelyeket továbbít a robotnak, így az annak megfelelően képes változtatni a szerszám relatív helyzetét. Ez azt jelenti, hogy ha a kéz nem képes – a skálázási tényezőt is figyelembe véve – akkora távolágba elérni, amely pozícióba a robotkar szerszámát juttatni szeretnénk, lehetőség van arra, hogy a kéz kinyitásával megszakítsuk a kapcsolatot, majd az ismételt megfogással tovább mozgatni a kart. Ehhez persze szükséges, hogy minden esetben, ha elindul a mozgási ciklus, a robotkarnak küldött jelek előző ciklusban tárolt értékei törlődjenek. Vagyis a robotkar koordinátái inkrementálódnak, a kar koordinátái, illetve a küldött jelek törlődnek. Így a mozgás hasonló lesz egy számítógépes egér mozgatásához, tehát ha kiértünk a mozgástartomány határára, megfogjuk az „egeret” és visszarakjuk az eredeti pozíciójába.

3. funkció

A harmadik ujj-kombinációban a hüvelykujj nyújtva van, a többi ujj ökölfogásban. Ebben a helyzetben a robotkar felé nem érkezik lineáris mozgásra vonatkozó parancs, a kéz mozgásától függetlenül. Viszont a kéz orientációjának mérése megkezdődik, és ezeket az adatokat továbbítja a vezérlés. Összefoglalva tehát a szerszám három tengely körüli forgatása ebben a funkcióban lehetséges. A mozgási és forgatási funkciót azért

választottuk külön, mert a szerszám forgatására csak a nagy mozgások véghelyzetbe érése után, a finompozicionálás szakaszában van szükség, a mozgás során nem. Ettől függetlenül természetesen a robotkar egyéb csuklóinak a szöghelyzet változása lehetséges, ez a robotprogram által vezérelt.

5.2 Skálázás és offset beállítás

A fő funkciókon kívül lehetőség nyílik a skálázási faktor megváltoztatására is, ami megadja, hogy a csukló egységnyivel történő elmozdulása a robotkar hány egységnyi elmozdulását eredményezi. Ennek értelmében a pozíció és orientáció vezérlésnek – ujjtartástól függően – három sebességfokozata van definiálva. Egy lassú fokozat a finompozicionáláshoz, egy gyors az átvezetésekhez és egy normál sebességű fokozat általános esetben. Emellett a vezérlő programon is található egy analóg kontroll változó, amely forgatásával az előre definiált skálafaktorok tetszőlegesen változtathatók felhasználási igényekhez igazíthatóan. A skálázás létrehozása során, illetve a funkciók közötti váltás esetén hangsúlyos figyelmet érdemel a szimulációs modellek helyzetének offsetet értékbeállítása.

Az egyik felmerülő probléma a kinect és a billentyű által megadott pozíciók közötti váltás kapcsán a különböző kezdőhelyzet. Szükség volt a robot vezérlő által használt koordináta adatokat összeegyeztetni a robot és a kéz szimulációjában használt koordináta adatokkal. Az így működőképes rendszerhez a kinectet és annak saját koordináta rendszerét illeszteni kellett. A probléma, hogy a kinectel érzékelt kéz pozíció és a robot természetes kezdőhelyzete közötti koordináta érték különbségeket, offseteket a skálázhatóság megőrzése érdekében dinamikusan kell változtatni, a skálafaktorok alapján meghatározható mértékben. A számítás a következőképpen történt:

$$\underline{x}_{robot} = K \cdot \underline{x}_k + (1 - K)\underline{x}_{k0} + \underline{x}_0$$

ahol

$\underline{x}_{robot}$: a robot pozíciója

K : az erősítés vagy skálafaktor a két rendszer között

$\underline{x}_k$: a kéz pozíciója a kinect által érzékelve

$\underline{x}_0$: az egyes irányokban lévő eredeti offset (K=1 esetén)

$\underline{x}_{k0}$: a kéz kezdeti pozíciója (minden K értékváltás esetén rögzítve)

Az ezen elven számított koordináta átváltás segítségével elkerülhetőek a kezdetben tapasztalt ugrások, hiszen a skálafaktor váltással az offsetek is változnak.

5.3 A vezérlő program kezelőfelülete

A kezelőfelületen minden esetben kiválasztható az eszköz, amellyel a vezérlést szeretnénk megvalósítani. Ez a funkciók kiválasztása esetében a kesztyű illetve az öt scrollbar, illetve a mozgáskövetés esetében a billentyűzet és a Kinect lehet.

Fontos, hogy a program elindításakor a Kinectből érkező jeleknek van egy átmeneti állapota, amikor az eszköz még nem ismerte fel a szkeletont, viszont már ekkor is értékeket ad, helytelenül. A jelenség miatt a kezdeti néhány másodpercben a Kinectet aktiváló menü letiltott állapotban van. A billentyűzettel történő irányítás során a következő billentyűkkel változtathatók a csuklókoordináták:

Q – x koordináta csökkentés, illetve x tengely körüli elforgatás negatív irányba

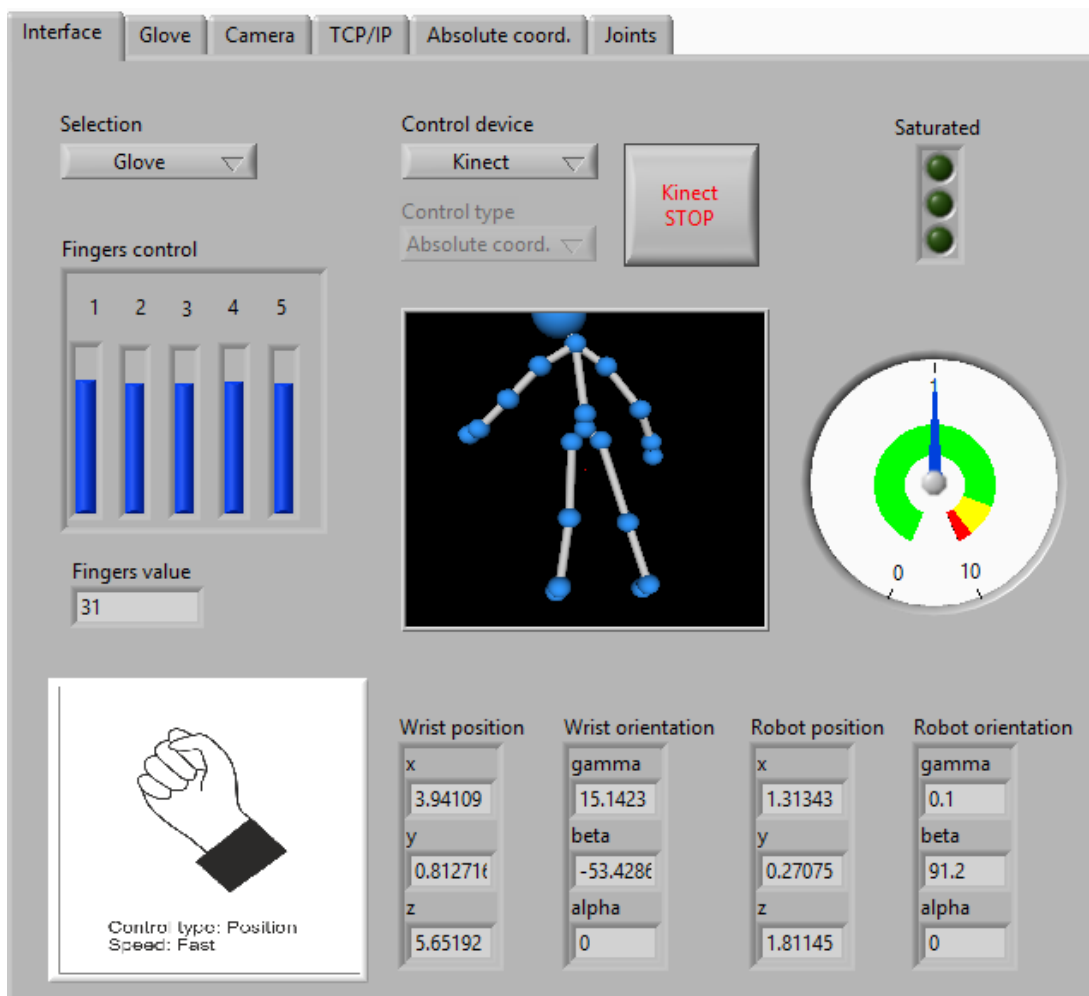
W – x koordináta növelés, illetve x tengely körüli elforgatás pozitív irányba

A – y koordináta csökkentés, illetve y tengely körüli elforgatás negatív irányba

S – y koordináta növelés, illetve y tengely körüli elforgatás pozitív irányba

Y – z koordináta csökkentés, illetve z tengely körüli elforgatás negatív irányba

X – z koordináta növelés, illetve z tengely körüli elforgatás pozitív irányba



25. ábra: A vezérlő program kezelőfelülete

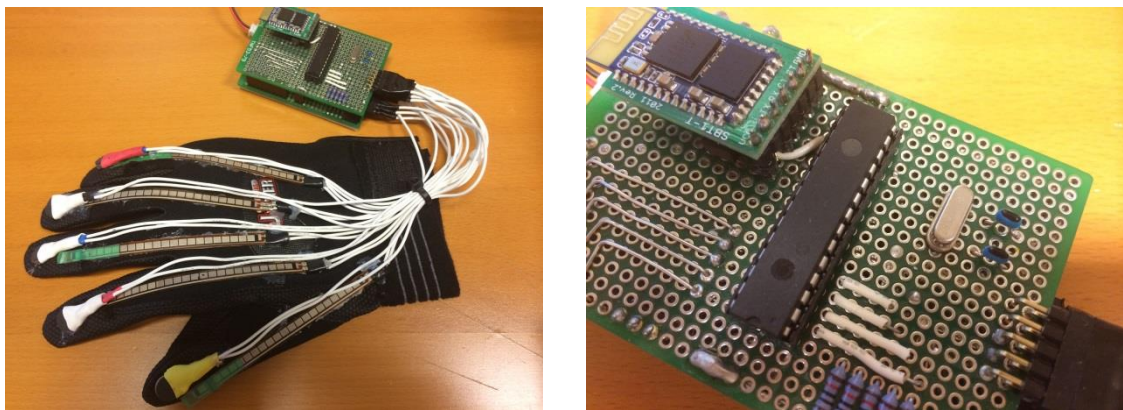
Forrás: saját készítés

Ahogy látható, a vezérlés típusának kiválasztásán kívül főleg visszajelző egységek találhatók a felületen. Ezzel ellenőrizhető, hogy minden egység rendeltetésszerűen működik-e, illetve az aktuális kéz és robot koordináták is fel vannak tüntetve. Természetesen ennél egy jóval szemléletesebb 3D grafikus felület is előhívható, amelynek részleteiről később lesz szó.

5.4 Vezérlő kesztyű

A felhasznált eszköz egy olyan egyedi készítésű kesztyű, ami szenzorokkal és aktuátorokkal felszerelve könnyen használható, intelligens alkalmazások létrehozását teszi lehetővé. A szenzorok az öt ujjra felszerelt nyúlásmérő bélyegek, melyek analóg jelet továbbítanak a kesztyűhöz csatolt tokban lévő mikroprocesszorba, érzékelhetővé

téve az ujjak behajlítottságának mértékét. Az aktuátorok pedig a több ponton felhelyezett rezgető-motorok, melyekkel a felhasználónak egy idáig tipikusan kihasználatlan érzékszervén keresztül tudunk információt visszajuttatni a rendszerről. A kesztyű egy Arduino processzorral van összekapcsolva, mely a jeleket feldolgozza és bluetooth-on keresztül továbbítja a számítógépnek, mely LabVIEW frameworkben fogadja a jeleket. A kesztyűt mint hardvert, mi a projekthez már készen kaptuk, a mi feladatunk az eddigi programokkal való összeköttetése volt, valamint a szenzorok kalibrációjának elvégzése.



26. ábra: A rendelkezésünkre bocsájtott vezérlő kesztyű

Forrás: Saját fénykép

A számítógépes helyettesíthetőség érdekében a főprogramban az irányítás módja választható, így az eszköz hiányában és az alkalmazás fejlesztése érdekében kiválasztható a LabVIEW grafikus felületéről analóg méterekkel való irányítás is. Természetesen a megcélzott hatékonyabb és kényelmesebb megoldást a kesztyűvel való irányítás jelenti, hiszen így a Kinecttel segítségével együtt egy kézzel megoldott az alapfunkciók közötti váltás, a pozíció és orientáció teljes körű irányítása, ami nagy előny hasonló rendszerekhez képest.

5.5 A kar mozgásának szimulációja

A kar mozgásának modellezésére a Kinectes irányítás mellett nincsen szükség, hiszen a szenzor által érzékelt pontok alapján a szimulációs modell kirajzolható. Az animációra magára azért van szükség, mert az operátor számára jó visszajelzést ad arra

vonatkozólag, hogy a szenzor megfelelően működik-e, az operátor mozgásával összhangban mozog. Emellett sokkal nagyobb szerepe van a szimulációs modellnek akkor, amikor áttérünk a billentyűzeten történő irányításra. Ilyenkor az operátor nehezen érzékeli a távolságokat, sebességeket, irányokat, ezért ilyenkor egy irányítást támogató animáció megléte elengedhetetlen a kényelmes működtetéshez. A legfontosabb szempont talán mégis az, hogy a kar modellje lehetővé teszi, hogy azonosítsuk azokat az irányokat, amelyek mentén a paraméterek változtathatók, hiszen a szimulációs modellben rögzített a két modell (robot és kar) egymáshoz képesti szöge, míg a valóságban ez nem igaz.

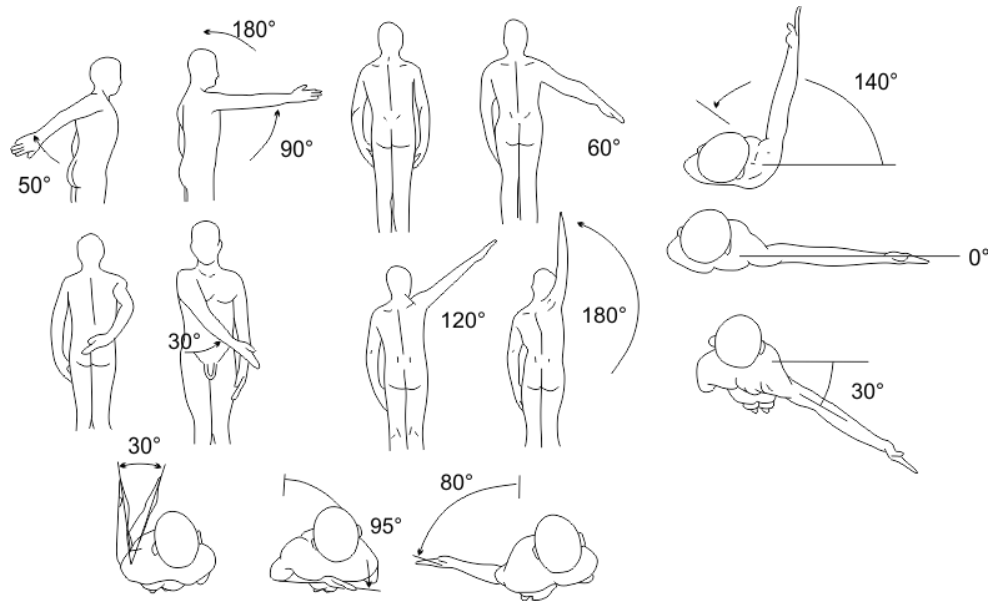
A kar kinematikája meglehetősen összetett, a feladat során alapvetően két koncepció született a modellezésre. Mindkettőnek vannak előnyei és hátrányai is, amelyekre a következő alfejezetekben térünk ki.

5.5.1 Pozícióvezérlés

A pozícióvezérlés során a nyomógombokkal a csukló abszolút koordinátáit változtatjuk. Ennek megfelelően a csukló a kívánt pontba mozog, miközben a kar egy ennek megfelelő helyzetbe kerül. A pontot a nyomógombokkal megadhatjuk a mozgásterjedelmen kívül is, hiszen a robotkar mozgásterjedelme lehet, hogy a kar mozgásterjedelmén kívül esik. Ez esetben a kar ismétlődő mozgásokat végez, míg a robotkar folyamatos egyirányú mozgást végez a csukló által megtett út függvényében. Mivel a csukló végpontját mindenképpen mozgásterjedelmen belül kell tartani, az első feladat a kar mozgásterjedelmének meghatározása volt.

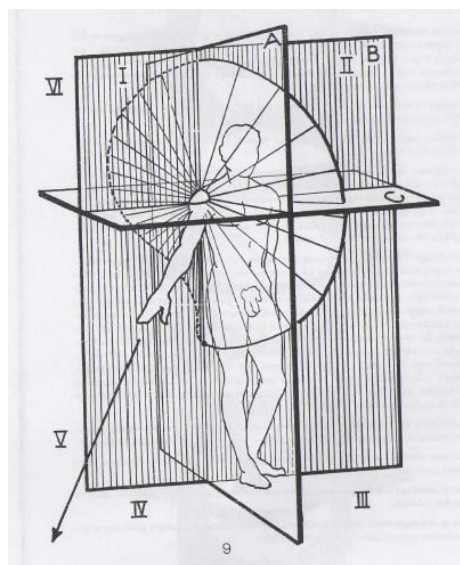
A vállízületi flexió és extenzió a szagittális síkban, a transzverzális tengely körül jön létre. Az extenzióban a mozgásterjedelem kicsi, mindössze 45° - 50° -os, míg flexióban eléri a 180° -ot. A felső végtag törzstől távolodó abdukciós mozgása a frontális síkban, a szagittális tengely körül jön létre, értéke elérheti a 180° -ot. Ezzel szemben az ellentétes irányú addukció a felső végtag neurális helyzetéből kiindulva mechanikailag lehetetlen, viszont minimális flexióban az addukciós mozgás is elérheti a 30° - 45° -ot. Extenzióban az addukciós mozgás ugyan lehetséges, de nem számottevő. A kar axiális rotációja kirotaáció esetén meghaladja a 80° -ot, berotáció esetén pedig akár 100° - 110° is lehetséges. Ilyenkor természetesen a teljes mozgásterjedelem csak úgy érhető el, ha az

alkar a törzs mögé kerül. A horizontális flexió addukcióval maximálisan 140° , a horizontális extenzió ezzel szemben korlátozott, csak 30° - 40° -ig kivitelezhető. Az értékek szemléletesen az alábbi ábrán láthatók. (11)



27. ábra: A kar mozgásterjedelme
Forrás: (11)

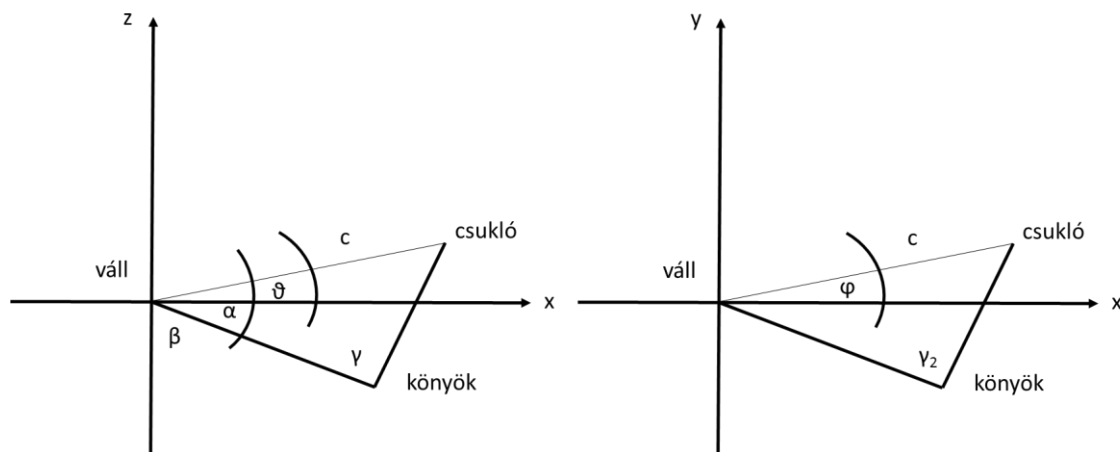
Az előbb felsorolt elemi mozgásokból kirajzolódik a kar teljes mozgástartománya, amely egy szabálytalan cirkumdukciós kúpot rajzol a térben. Ez a kúp egy olyan gömb körülhatárolt része, aminek a középpontja a váll, és sugarának hossza a felső végtag hosszával egyenlő. A cirkumdukciós kúpon belül képes a kar mozogni a törzs elfordulása nélkül.



28. ábra: Cirkumdukciós kúp
Forrás: (11)

A cirkumdukciót figyelembe véve kijelöltünk a mozgásterjedelmen belül egy kisebb területet abból a feltételezésből, hogy a természetes munkavégzés során nem használjuk ki a teljes mozgástartományt, hiszen a végpontok elérése nem kényelmes, nagyobb ciklusszámban pedig a határok feszegetése egyenesen káros. Ez az egyszerűsítés könnyebbé teszi a szimulációs határok beállítását, és közelebb hozza a modellt a valósághoz. Megjegyzésként, a feladatunkban nem feltétel az, hogy a mozgás a törzs mozgása nélkül jöjjön létre, tehát szükség esetén valójában a cirkumdukción kívül is megjelenhet a csukló, de a munkavégzés során egy helyben ülő operátort feltételezünk, illetve a szimulációs modellben az értékek számítása a váll középpontú relatív koordináta rendszerben történik.

A pozícióvezérlés során három eset lehetséges, amelyet a következő részben ismertetünk, az említett szögek és távolságok az alábbi ábra alapján értelmezhetők.



29. ábra: A pozícióvezérlés paraméterei

Forrás: saját készítés

1. eset: A nyomógombok által meghatározott térbeli pont a mozgásterjedelem határán van. Ekkor a kar kinyújtott állapotban van, tehát a γ szög értéke 0. A csukló koordináták egyszerűen átszámíthatóak gömbkoordinátákká, ezzel meghatározva a γ és ϑ szöget. Az összefüggés alább látható.

$$x = R \cdot \cos(\vartheta) \cdot \cos(\varphi)$$

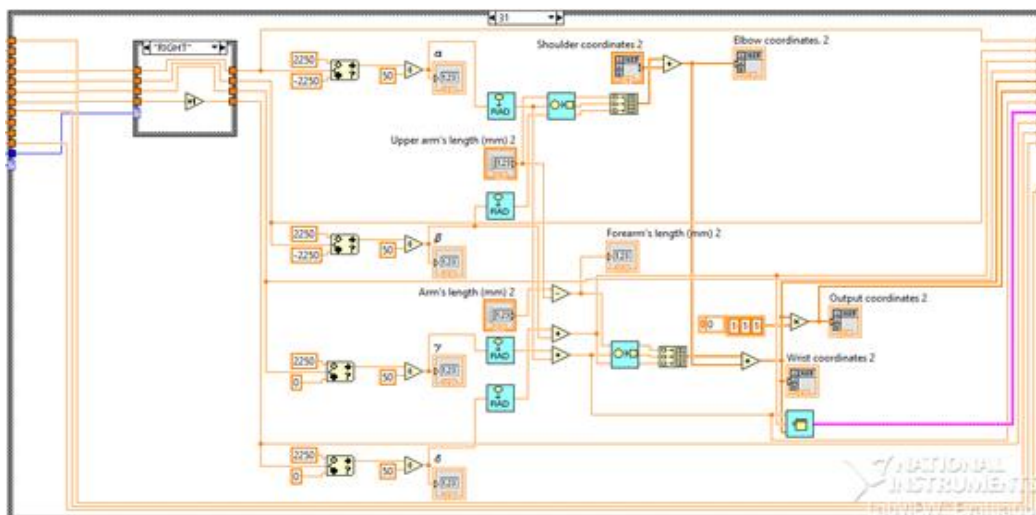
$$y = R \cdot \cos(\vartheta) \cdot \sin(\varphi)$$

$$z = R \cdot \sin(\vartheta)$$

Az R értéke a kar hossza. Miután megkaptuk a gömbkoordinátákat, hasonló összefüggéssel megkapjuk a felkar hosszának ismeretében a csukló koordinátáit is derékszögű koordináta-rendszerben.

2. eset: A térbeli pont a mozgástartományon belül helyezkedik el. Ekkor a koordináták négyzetösszege megadja a c távolságot, amelyből aztán az előbbihez hasonló módon adódnak a szögek a gömb koordináta-rendszerben. Az alkar, felkar hosszából és c távolságból cosinus-tétellel megadható α és γ szög, amelyből β számítható. Megjegyzésként, ez a gondolatmenet csak akkor érvényes, ha a γ_2 szög értéke 0, hiszen amennyiben nem, úgy a cosinus-tételt más távolságokból, a kar hosszainak x - z síkra vett vetületéből kell számolni. Mivel a mozgás minden személyre jellemzően más, és ugyanazt a pontot különböző csuklószögekkel is el lehet érni, a szimuláció során azt az elvet követtük, hogy az x - y síkban a könyök nem mozog. A természetes mozgás tehát ettől kissé eltérő lesz minden esetben, de a vezérlés szempontjából a csuklószög indifferens, így a szimulációban egy megvalósítható mozgást modelleztünk, és nem a valódi trajektória leírására törekedtünk. Megjegyzendő, hogy a csuklószög nagyban befolyásolja a kéz orientációját, ami viszont hatással van a robotkarra, de éppen ezért a forgás és mozgás funkcióit az előző fejezetben tárgyalt módon különválasztottuk.

3. eset: A pont a mozgásterjedelmen kívül esik. Ilyenkor ezt a pontot nem tudja követni a csukló, viszont a robotkar a saját mozgástartományában igen. Tehát a mozgástartomány elérése után a csuklókoordináták törlődnek és default értékről inkrementálódnak, míg a robot koordinátaiban nincs ilyen törlés.



30. ábra: A pozícióvezérlést megadó programrészlet

Forrás: saját készítés

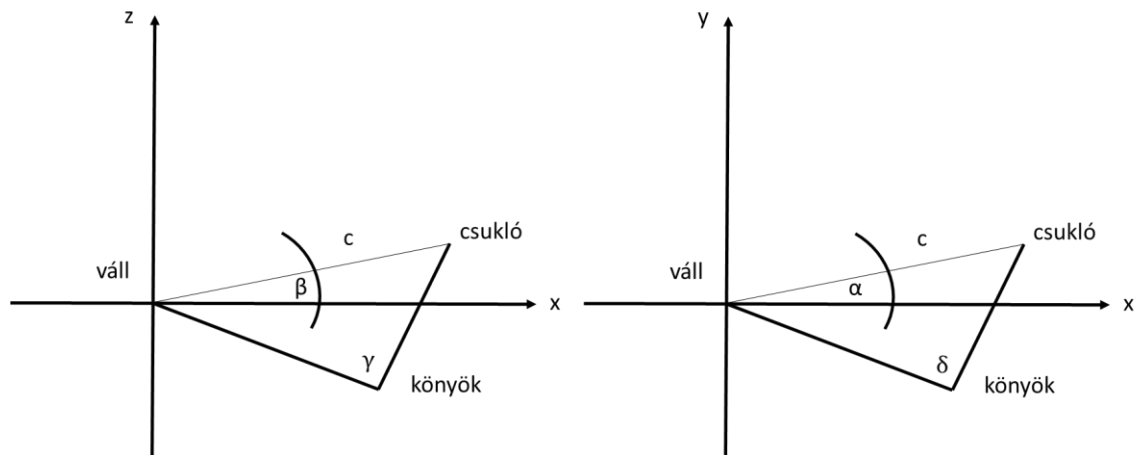
5.5.2 Csuklóvezérlés

A csuklóvezérléses karmozgatás szimulációját azért hoztuk létre, hogy egy természetesebb mozgást kapjunk vissza a programban. Ez abban különbözik az előző esettől, hogy nem adjuk meg a csukló koordinátáit, helyett a kar helyzetét egy gömbkoordinátákkal megadott rendszerben. Ezekből az értékekből pedig számítható a csukló pozíciója derékszögű koordináta rendszerben, illetve a robot bemenő paraméterei. Ebben az üzemmódban a billentyűzet mozgásával a kar szögeit vezérelhetjük, amelyből az előzőekben ismertetett módon minden ízület pozíciója meghatározható. Kérdés, hogy ez a vezérlés miért ad természetesebb mozgást. A válasz, hogy a valóságban is tulajdonképpen a csuklószögeket irányítjuk az izmok segítségével. Az emberi test szegmensei merevek, az összes mozgás az ízületekben jön létre azáltal, hogy a szegmenseket összekapcsoló izmok feszítése és elengedése szögváltozást okoz az ízületekben, vagyis a szegmensek relatív szöge változik.

Látható, hogy ennek a módszernek az előnye az egyszerűbb modellezhetősége, hiszen a gömbkoordinátákkal megadott értékekből egyértelműen meghatározhatók az értékek derékszögű koordináta-rendszerben, míg fordítva nem egyértelmű a hozzárendelés az inverz szögfüggvények miatt, illetve a mozgás is hasonló a valódi mozgáshoz. Viszont így a szimulációban nem tudjuk megadni az elérendő pontot matematikai pontossággal,

csupán „érzésre” nyúlunk az adott pontba. Ez valószínűsíthetően nem fog gondot okozni az operátornak, hiszen a valóságban is így irányítjuk a kezét.

Ebben a rendszerben a szimulációs program is máshogy definiálja a változókat. Az alábbi ábrával értelmezhetőek a használt szögek.



31. ábra: A szögvezérlés paramétereit

Forrás: saját készítés

Ennek megfelelően, ha a vezérlő programban a csuklóvezérlést választjuk, akkor a gombok ezeket az értékeket fogják változtatni

Q és W: α csökkentés és növelés

A és S: β csökkentés és növelés

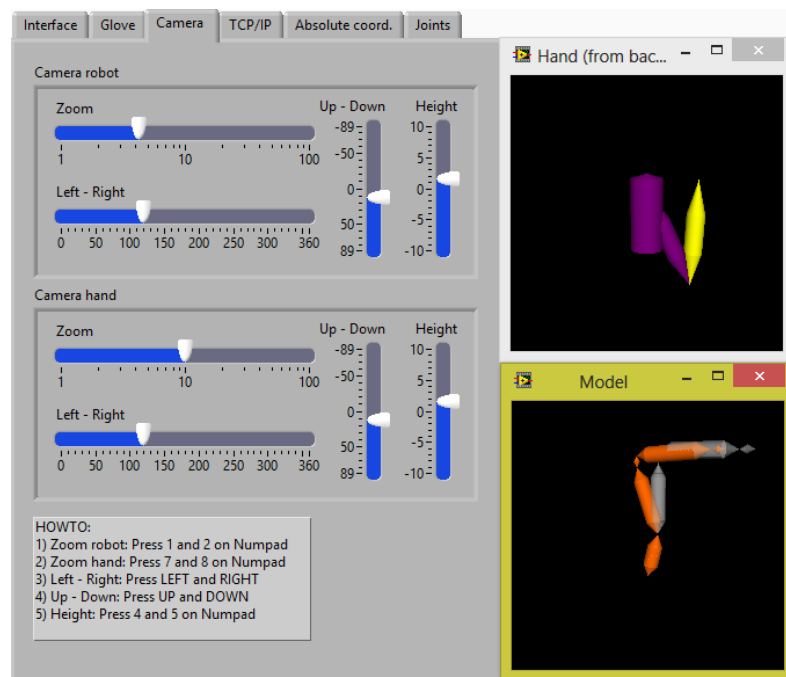
Y és X: γ csökkentés és növelés

E és R: δ csökkentés és növelés

5.6 A fej mozgásának szimulációja

A fej mozgatása azért fontos, mert a programban lehetőség van arra, hogy egy virtuális képet kapjunk a robotkar helyzetéről egyfajta visszajelzésként, illetve a parancsok kiadása miatt is. Ezek az értékeket nem adjuk át a robotnak, a mozgásban nincs szerepe. A fejen lévő Oculus Rift pozíciója és orientációja határozza meg a kamera pozícióját és orientációját, tehát azt az oldalt és szöveget, amelyből rálátásunk van a gépre. Emiatt a fej 3D-s megjelenítését nem kellett elvégeznünk, viszont a koordináták számolására

szükség van a virtuális kép transzformálásához. A fej mozgástartománya minimális, azonban figyelembe véve a törzs mozgásait is, a határértékeket széles intervallumban határoztuk meg. Az Oculus Rift szoftveres szimulációjában szintén nyomógombokkal lehetséges a kamerakép módosítása. A két kamera szinkronban mozgatható, azonban a közelítés-távolítás funkciót szétválasztottuk a könnyebb használhatóság érdekében.



32. ábra: A kamera helyzetének megadása nyomógombokkal

Forrás: saját készítés

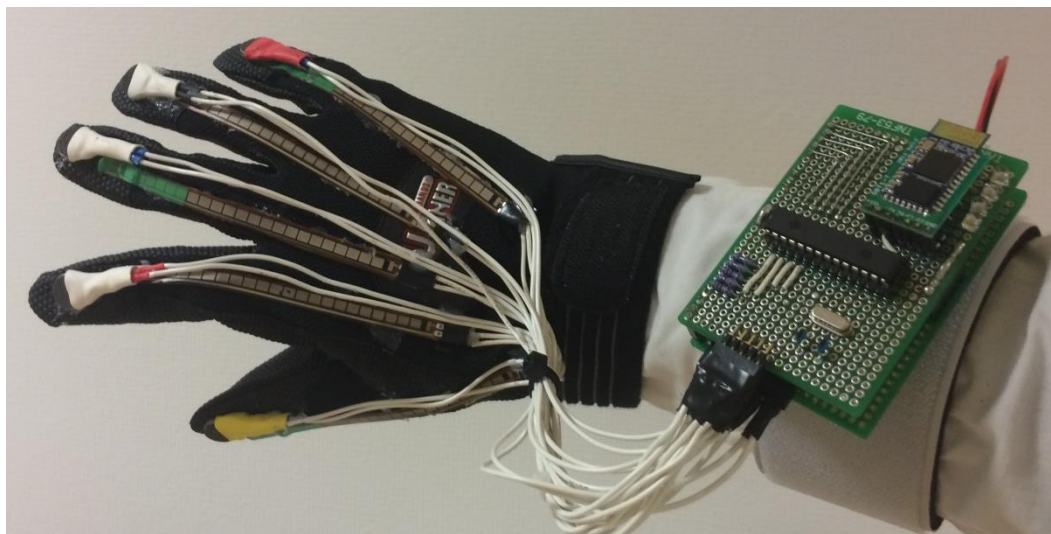
6 Értékelés

Ezen projekt keretében megvalósításra került egy olyan platform, melynek segítségével egy távoli robotkart irányítunk internetes kapcsolaton át, azaz telemanipulálunk. Az első lépcsőben megalkotásra került a szimulációs modell, melyben a robotot és a szenzorokat virtuálisan lemodelleztük és működésüket szimuláltuk. Az eszköz jó alapként szolgált a további fejlesztéshez, hiszen a már meglévő elméletet kellett átültetni a gyakorlatba. Fontos szempont volt az internet okozta késleltetés, melyet a szimuláció során mesterségesen állítottunk elő, a gyakorlati megvalósításban pedig valós problémaként szembesültünk hatásával. Az egyes virtuális bemeneteket lecseréltük hardveresre, és a szükséges illesztések után valóban létrejött az a működés, amit elterveztünk. Az internetes kapcsolatot a gyakorlatban megvalósítottuk, és a két oldalt elhelyezkedő programok sikeresen működtek együtt a gyakorlatban.

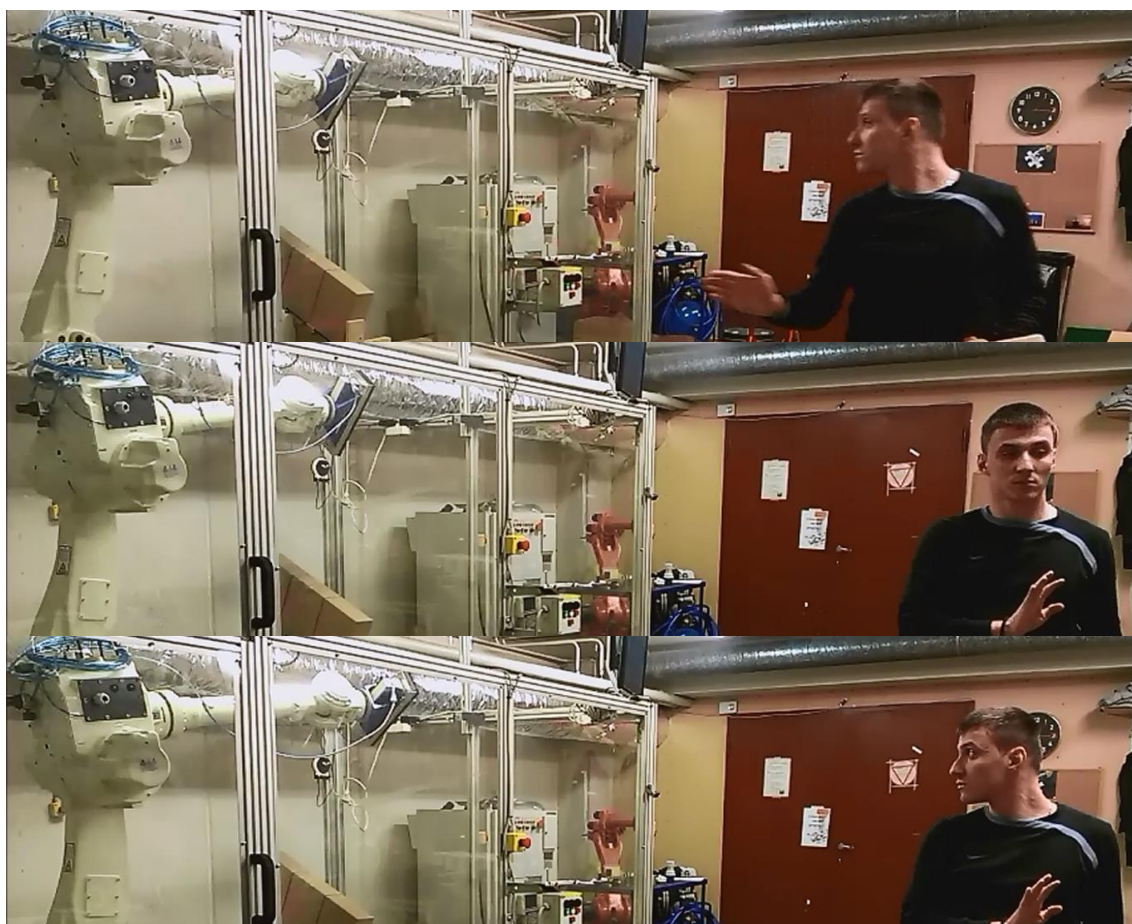
A Kinect használata az operátor kézmozgásának bemenetté alakításához megfelelőnek bizonyult, ám szükséges volt néhány megfontolás: A Kinect használata egyszerű, a felhasználása további ráfordításokat nem igényel, és sokoldalú módon lehet kihasználni a funkcióit. A felhasznált Kinesthesia szoftver akkor működött legjobban, ha az egész alakot fel tudta mérni, hiszen akkor is megrajzolja az alakot, amikor egyes testrészek takarásban vannak. Ez a munkaállomás kialakításakor fontos szempont lehet, jelen jellemzők mellett. A rendszer által érzékelt jelet átlagoló szűrővel simítottuk, a zajmentesebb jel előállításának érdekében, azonban ez hozzá tett a rendszerhez egy minimális (az internet okozta késéshez képest elhanyagolható) késleltetést. A rendszerünk egyik nagy előnye a skálázhatóság, hiszen annak ellenére, hogy a Kinect által érzékelt pozíció felbontása a néhány mm-es-től a cm-es felbontásig ingadozik, az irányítás szempontjából tetszőleges kicsinyítéssel elérhető bármilyen kívánt pontosság.

A programban az egymástól függetlenül megjelenő ablakoknak köszönhetően az egyes modellek eredményei jól láthatóak. Az operátor oldali program és a robot oldali program, illetve ezeknek a modelleknek külső beavatkozás segítségével történő irányítása is mind azt igazolja, hogy a forward kinematics és az inverse kinematics függvények is helyesen, célnak megfelelően működnek. Az felhasználói felületen lévő kijelzők segítségével nem csupán a program bemenete és valós helyzete, de a kéz, az

ujjak mozgása is egyértelműen követhető. A szoftverek tehát a vártak megfelelően működnek és lehetővé teszik, hogy az operátor távolról installálja a robot mozgásokat.



33. ábra: Vezérlő kesztyű használat közben
Forrás: saját fénykép



34. Ábra: Robot mozgásának tesztje
Forrás: saját fénykép

7 További munkák

A projekt blokk-diagramja és a megvalósított funkciók alapján is látható, hogy a projektben számos további lehetőség rejlik. A további fejlesztési munkálatok több irányban is folytatódhatnak akár szoftver, akár hardver terén.

7.1 Erőszabályozás, trajektória-simítás

A távoli irányítás egyik legalapvetőbb problémája az operátor lassabb reakcióideje, mivel az emberi reakcióidőhöz hozzáadódik a távoli kapcsolat válaszügye is. Éppen ezért nagymértékben megnő annak a veszélye, hogy egy ütközés során a túlterhelése a motoroknak nagyobb mértékű lesz, emiatt a károk mértéke is megnőhet. Azért, hogy a motorok ne menjenek tönkre, a robotba be van építve egy automatikus leszabályozó egység, amely túlterhelés esetén azonnal leállítja a robotot. Ha a robot ütközik és megáll, az újraindítását a felügyelőnek kézzel kell elvégeznie. Ez hosszabb távon jelentősen megnövelheti a munka idejét, ráadásul nehézkessé teszi az operátor munkáját. Egy marógép installálása esetén meg egyenesen lehetetlenné válik a beállítás anélkül, hogy a robot ütközne egy felülettel.

Érdemes tehát a robot mozgását hardveres helyett szoftveresen szabályozni, mivel a pálya módosításával nem kell a robotot minden alkalommal újraindítani. Ehhez néhány beépített erőszabályozót kell telepíteni, amely ütközés esetén azonnal újraszámolja a pályát és ütközés helyett a felülethez simulva, de csak kis erőt kifejtve követi a tárgy felületét, amíg szükséges. Mivel a leszabályozó rendszernek az érzékenységet kézzel lehet módosítani, az erőszabályozót is könnyen tudja az operátor távolról installálni, hogy a robot biztonságosan, mégis valós időben vezérelve tudjon tetszőleges körülmények között mozogni.

Szintén biztonsági okok miatt a robot nem csak ütközéskor, de sok hirtelen, nagymértékű irányváltoztatáskor is leszabályoz, mert ilyen mozgásokat normál üzemi esetén nem kell a robotnak végrehajtania. Ezek elkerülésére érdemes a vezérlési program oldalán egy trajektória simítást elvégezni, hogy a különböző remegéseket és nem valós mozgásokat kiszűrje a program.

7.2 Manipulációs kódrendszer továbbfejlesztése

Az operátor feladatának hatékony elvégzéséhez fontos kérdés a telemanipulációs vezérlés ergonomikus kialakítása. A perifériák kényelmi szempontja kevésbé tartozik ide, mivel erre a hardverek gyártóinak kell ügyelni, viszont a szoftver ergonómiája szintén a fejlesztési projekt része. Eszerint az ujjak által vezérelt robot irányítása, az ujjak behajlításának érzékenysége és az esetlegesen felmerülő egyéb kényelmi szempontok kidolgozása és megvalósításának javítása a cél. A meglévő parancsokon túl további parancsok is bevezethetők a rendszerbe, melyekkel egyéb eszközök (pl. megfogó, fűrészsorszám) vagy egyéb funkciók (pl. pozíció elmentése vagy előhozása, tanító vagy automatikus módok közötti váltás) működtethetők a roboton. Az így kialakítható ujj-kombináció alapú irányítási rendszer természetesen bonyolodik és a tanulási folyamatot hosszabbítja, de a funkcionalitást nagy mértékben növeli.

7.3 Visszajelzési mechanizmusok kidolgozása

Az operátor munkájában rendkívül fontos, hogy megfelelő számú ingert kapjon a munkahelyről és a körülményekről. Mivel nincsen jelen, ezért ezeket az ingereket digitálisan, a szoftverek segítségével kell eljuttatni hozzá. Ez jelenleg csupán képek formájában történik, viszont hangok és a kesztyűre szerelt excenterek segítségével jóval több ingert kap az operátor ahhoz, hogy még hatékonyabban tudja feladatát elvégezni.

7.4 Implementálás különböző robottípusokra

A szimulációs program egy konkrét robotra készült el, viszont több különböző típusú robotra történő implementálás esetén lehetőséget kapnak további robotfejlesztő cégek a rendszer szélesebb körben történő alkalmazására. Az egyes modellek közötti áttérés a kinematikai paraméterezés kiterjesztésével és a dinamikai szempontok figyelembe vételével megoldhatóak. Ebben az esetben a telemanipulációs rendszer, mint különálló fejlesztés értendő, amely kutatócsoport a robotfejlesztő cégektől akár független szervezetként is képes lehet működni.

7.5 Kinect és képfeldolgozás magasabb szintű felhasználása

A kinect alkalmazások és a képfeldolgozás fejlődésével a az alakfelismerésben rejlő további lehetőségek is előtérbe fognak kerülni. Jelenleg az egyes ujjak felismerése, elkülönítése és követése még nem áll azon a szinten, hogy megbízhatóan lehessen rá építeni komoly követelmények mellett. Ezen alkalmazások fejlődésével számunkra is új lehetőségek nyílnak majd. Lényegesen olcsóbb lehetne az általunk fejlesztett rendszer telepítése, ha nem lenne például szükség kesztyű használatára, és ugyanazokat a parancsokat egyéb hardver nélkül lehetne valós időben felismerni.

7.6 Beépítés használatban lévő távoli vezérlést alkalmazó rendszerbe

A program modulos felépítése lehetővé teszi, hogy már meglévő távoli irányítást alkalmazó szoftvercsomagokba könnyen beilleszthető legyen. Ennek előnye, hogy a két féle programcsomag egyesítésével alkalmazni lehet mindkettő előnyét. Többek között a magyar fejlesztésű, VirCa elnevezésű szoftverrel lesz érdemes összeilleszteni az itt bemutatott programot, mivel annak grafikai megjelenítése és internetes kommunikációja meghaladja a LabVIEW által nyújtott lehetőségeket. (12) A VirCa bár rendelkezik már néhány egyedileg elkészített perifériát kezelő kiegészítő modullal, a LabVIEW használata és az abban történő összeköttetés nagyban megnövelné a lehetőségeket, hogy végül egy jó hatékonysággal működő telemanipulációs rendszer jöjjön létre. (13)

Irodalomjegyzék

1. Oculus Rift - Virtual Reality Headset for 3D Gaming | Oculus VR. [Online] Oculus VR Inc., 2014. www.oculusvr.com/.
2. **Gyula, Mester.** *Robotika*. Szeged : Szegedi Tudományegyetem, 2011.
3. **Siciliano, Bruno and Khatib, Oussama.** *Handbook of Robotics*. Berlin : Springer-Verlag, 2008.
4. **Siciliano, Bruno, et al., et al.** *Robotics: Modelling, Planning and Control*. London, UK : Springer-Verlag, 2009.
5. **Wikipedia.** Kinect. [Online] [Hivatkozva: 2015. január 5.] <http://en.wikipedia.org/wiki/Kinect>.
6. **József, Pálfalvi.** KINECT szenzor intelligens terekben. [Online] 2011. május 7. [Hivatkozva: 2014. október 10.] http://aigroup.mit.bme.hu/system/files/onlab_feladat/2011/06/palfalvi_kinect_onlab_doc.pdf.
7. **Loriot, Yannick.** Kinect: How to install and use OpenNi on Windows. [Online] 2011. március. [Hivatkozva: 2014. október 26.] <http://yannickloriot.com/2011/03/kinect-how-to-install-and-use-openni-on-windows-part-1/>.
8. **National Instruments.** Kinesthesia Toolkit for Microsoft Kinect. [Online] 2012. [Hivatkozva: 2015. január 5.] <http://sine.ni.com/nips/cds/view/p/lang/hu/nid/210938>.
9. **University of Leeds.** Success for students and researchers at global level. [Online] 2012. augusztus 14. [Hivatkozva: 2015. január 5.] http://www.leeds.ac.uk/forstaff/news/article/3392/Success_for_students_and_researchers_at_global_level.
10. **Hunyadi, László és Vita, László.** *Statisztika I*. Budapest : AULA Kiadó, 2008.
11. **Kapamdji, Ibrahim A.** *Az ízületek élettana*. Budapest : Medicina Könyvkiadó Zrt., 2009.
12. **VirCA.** [Online] Computer and Automation Research Institute Cognitive Informatics Research Group. [Hivatkozva: 2014. October 21.] <http://virca.hu/>.
13. **Sanderud, Audun R. és Reme, Fredrik.** Sensor based control of industrial robots. Trondheim : Norwegian University of Science and Technology, 2011.