DIPLOMAMUNKA

Pálfalvi József

2012

Pálfalvi József (ERKDJB**)**
szigorló informatikus hallgató részére

Kinect alapú ambiens intelligencia AAL alkalmazásokban

Ambiens intelligens környezet egyik jellegzetes és nagy horderejű alkalmazása az un. AAL (Ambient Assisted Living) - az élhetést segítő informatikai technológiák integrálása, főleg idősebb és egyedül élő emberek szolgálatában. Az idősebb, vagy valamilyen módon korlátozott (mozgás, látás, hallás) emberek esetén kiemelten fontos az egészségügyi állapotokat stabilizáló napi rutin – az un. ADL (Activities of Daily Living) a hétköznapi, sokszor ismételt cselekvések – a követése és értékelése. Az egészségügyi és a szociális felügyelet szempontjából egyaránt fontos az előírt gyógyszerszedési, folyadék beviteli, táplálkozási, higiéniás, rehabilitációs gyakorlatozási, stb. rutin mind felügyelete és a betartásának támogatása, mind a rutintól való eltérések észrevétele, minősítése, és a megfelelő kezelése.

A probléma megoldása nagyon nehéz, tekintettel arra, hogy a szándékos (pl. gyógyszerszedés kerülése), ill. a spontán, vis major jellegű (pl. elesés) abnormális viselkedések spektruma igen széles, a következményei sokrétűek, és az egyes fontosabb viselkedések szenzorikus észrevétele, „megmérése", az információk hiánya és bizonytalansága miatt sokszor igen kérdéses. A diplomaterv alapvető célja egy új és potenciálisan hasznos szenzorikus elem – Microsoft Kinect – igényes kivizsgálása az AAL alkalmazások szempontjából. Kinect szenzorplatformtól elvárható bizonyos AAL/ADL követési és diszkriminálási funkciók hatékonyabb megvalósítása.

A feladatok részletes leírása:

1. Irodalom alapján foglalja össze az AAL/ADL rutinfeladatok körét, azok modellezése, azonosítása, diszkriminálása, követése, valamint az azokról való eltérések minősítése szempontjából. Irodalom alapján adja meg a Kinect eszköz tartalmas funkcionális és rendszertechnikai ismertetését.

2. Adjon áttekintést arról, hogy egyes AAL/ADL rutinok támogatását hogyan tudná megvalósítani Kinect funkcionalitásainak felhasználásával (esetleg más szenzorokkal társítva).

3. Válasszon kritikusan egy (néhány) rutinfeladatot (gesztikulálás, felkelés székről, felkelés ágyról, stb.), aminek felismerésével foglalkozni fog a továbbiakban. Tervezze meg a felismerés algoritmusait és implementálja azokat tesztelhető hardver/szoftver eszközökkel.

4. Laborkörnyezetben tervezzen meg alkalmas teszteket, hajtsa azokat végre és dokumentálja. Összegezze a kísérletek eredményét. Fogalmazza meg a kísérletek tanulságát és a kialakított Kinect alkalmazás erős es gyenge pontjait. Adjon kitekintést a Kinect AAL jellegű használhatóságáról.

Tanszéki konzulens: Dr. Dobrowiecki Tadeusz, docens
Külső konzulens: -
Budapest, 2011. október 3.

……
Dr. Jobbágy Ákos
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Pálfalvi József

Kinect alapú ambiens intelligencia AAL alkalmazásokban

KONZULENS

Dr. Dobrowiecki Tadeusz
BME MIT

BUDAPEST, 2012

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Pálfalvi József**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2012. 05. 11.

.....
Pálfalvi József

Kivonat

A diplomamunkában a Microsoft által 2010 végén bemutatott Kinect szenzor használhatóságát mértem fel AAL (Ambient Assisted Living) alkalmazásokban. A szenzor - infra-kamerájának köszönhetően - képes a 3D mélységérzékelésre, és a hozzá tartozó szoftver keretrendszer segítségével az emberi alakok megtalálására és tartós követésére a háromdimenziós térben. A Kinect alapú valósidejű háromdimenziós követés az AAL rendszerekben még kihasználatlan, így a munka célja az alkalmazhatóság kivizsgálása, és egy alapfeladatokat megoldó AAL alkalmazás elkészítése.

A diplomamunka első felében az AAL alkalmazások alapvető feladatait és tervezési aspektusait mutattam be, előzetes tesztekkel felmértem a szenzor képességeit, majd a szakirodalomban található, meglévő AAL rendszerek áttanulmányozása után egy Kinect alapú AAL rendszer tervét vázoltam fel. A diplomamunka második felében mutattam be a végső rendszer tervezésének és megvalósításának lépéseit, mely a Kinect szenzorból érkező adatokra alapozva old meg alapvető AAL feladatokat, többek között általános gesztusfelismerést, elesés detektálást és ADL (Activities of Daily Living) aktivitás felismerést. Az általános gesztusfelismerés egy működő gesztusfelismerő megoldás alapján lett kialakítva, mely a DTW (Dynamic Time Warping) algoritmust használja a gesztusok felismeréséhez. Az ADL aktivitás felismerés alapja egy többrétegű modell, melynek első szintjét a szenzor által megfigyelt tér és a benne található objektumokból (asztal, szekrény, ajtó… stb.) kialakított virtuális térkép (fizikai kontextus) képezi. A modell második szintjén a gesztusok és mozgási adatok vannak, míg a legfelső szint reprezentálja a tényleges ADL aktivitásokat. A rendszer a szenzor által követett ember valós térbeli pozícióját levetíti a virtuális térképre, majd az alany pozíciója, sebessége, nézési iránya, és az elvégzett gesztusok alapján képes az ADL aktivitásokat felismerni. Az elesés detektálás az AAL alkalmazásokban egy megoldott probléma, de a meglévő rendszerek képességei különbözőek. Munkámban egy általános, megbízható elesés detektáló rendszert mutattam be, amely az alany sebessége, pozíciója és további temporális (időbeli) paraméterek alapján működik, és képes kezelni a legtöbb szélsőséges esetet is, beleértve a majdnem teljes kitakarásba esést, a tartós földön fekvést, és a félig ágyra, székre való esést. Az ADL aktivitás felismerés és elesés detektálás számos paraméter alapján működik, melyek fuzionálására fuzzy következtető rendszert használtam. A végső Kinect AAL rendszer így képes a felhasználónak visszajelezni a követett alany ADL aktivitásit, és vészhelyzet esetén riasztani.

Abstract

The Microsoft Kinect motion sensor was released at the end of 2010. The infrared projector and camera of the sensor make it possible to sense the observed scene and track the movement of individuals in three dimensions. Kinect-based, real-time, three dimensional tracking is a new approach in AAL (Ambient Assisted Living) applications, therefore the main objectives of my thesis was to survey the applicability of the Kinect sensor in AAL systems, to design an application which can solve basic AAL tasks, such as fall detection, gesture recognition and ADL (Activities of Daily Living) activity recognition.

In the first part of the work, I described the main aspects of designing AAL systems. Next, I examined the general capabilities of the sensor by preliminary experiments. Finally, I studied the available AAL systems and presented the preliminary design of a possible Kinect AAL system.

In the second part of my work, I presented the steps of system design and implementation. The gesture recognition is based on an available system which uses the DTW (Dynamic Time Warping) algorithm to recognize gestures. The base of ADL activity recognition is a multilevel model, of which the first level is a manually defined virtual map (context) that is created from the real world scene. The second level represents the gesture and movement information extracted from the sensor, and the third level represents the actual ADL activities. The system tracks the user, and maps his movement to the virtual map, which works like a filter for selecting the possible ADL activities. After determining the user's velocity, position, orientation and gestures, the system is able to recognize the real ADL activities.

Fall detection is a general AAL problem, which is solved by many AAL applications, but with quite different capabilities and precision. The fall detection system presented in this work measures the velocity, the height of the center of mass, and other temporal parameters, and makes it possible to recognize falls, even in extreme conditions, such as falling behind objects (being almost completely occluded), falling on objects like chairs, and long lasting motionlessness. As a technique, the Kinect AAL system uses a fuzzy inference engine for data fusion, as well as, ADL activity and emergency event generation.

Bevezető

Társadalmunk egyre növekvő tempóban öregszik, ugyanakkor az idős emberek is ragaszkodnak az önálló élethez. A korral összefüggő mozgásszervi, mentális és érzékelési betegségek miatt az önállóan (sokszor egyedül) élő idős emberek sok veszélynek, kockázatnak vannak kitéve, még a saját élőkörnyezetükben is. Ahhoz, hogy biztosítani lehessen az önálló életet, célszerű olyan rendszereket bevezetni, amik folyamatosan megfigyelik, monitorozzák az adott élőkörnyezetet, időben felismerik a szokatlan mozgásokat, mozgássorozatokat, egészségügyi krízishelyzeteket (pl. elesés), és megbízható módon képesek ezt jelezni.

Sok ilyen intelligens rendszer került kifejlesztésre, melyek elsősorban több szenzorból nyert adatok együttese alapján próbálnak következtetéseket levonni. A rendszerek többségének alapja egy, vagy több kamera, melyek képeinek feldolgozásával az adott környezetben élő embereket követik nyomon. A személyi jogok tiszteletben tartása miatt ezeknek a módszereknek az alkalmazása sok akadályba ütközik, legtöbbször csak emberi sziluetteket használnak fel a kameraképek alapján, ezzel nagyban korlátozva a rendszerek képességét. Más szenzorok, mint például gyorsulásmérők, gait (lépésdinamika) érzékelők, infravörös érzékelők sokszor képtelenek megkülönböztetni egy ember elesését egy tárgy leejtésétől.

Az élhetést támogató megfigyelési rendszerek másik fontos feladata a begyűjtött adatok feldolgozása, az információk szűrése, és a végső következtetések levonása. Ehhez különböző tanuló és következtető rendszereket, heurisztikákat használhatunk, de fontos tisztában lenni az AAL (Ambient Assisted Living) és ADL (Activities of Daily Living) alapvető kihívásaival, igényeivel.

A Microsoft által 2010 végén bemutatott Kinect szenzor, mely eredetileg a szórakoztatóipar számára lett kifejlesztve, új lehetőségeket tárt fel az emberi alakok felismerése, és az ambiens élhetést támogató rendszerek kialakítása terén. Az eszköz és a hozzá tartozó szoftver architektúra képes olyan adatokat szolgáltatni, melyeket ma még semmilyen más eszközzel – ilyen kompakt formában - nem lehet előállítani. Ez a technológia olyan lehetőségeket tár fel, melyeket AAL alkalmazásokban még nem alkalmaztak, és kiaknázásuk nagyban elősegíthetné az AAL feladatok megoldását.

A diplomamunka első fejezetében ismertetem az AAL és ADL témaköröket, és rámutatok az AAL problémákat megoldó alkalmazások fontosságára. Bemutatom az AAL

rendszerek tervezésekor felmerülő legfontosabb szempontokat, és az AAL témakörben készülő alkalmazások mai technológiai állását.

A második fejezetben bemutatom a diplomamunka központi elemét képző Kinect szenzort, ismertetve a szenzor működésének alapjait, a szenzorhoz használható szoftver keretrendszereket, és alapvető teszteseteken keresztül egy előzetes felmérést végzek a szenzor használatának lehetőségéről AAL alkalmazásokban.

A harmadik fejezetben bemutatok és elemzek néhány meglévő AAL alkalmazást, majd az előző fejezetek információira építve felvázolom a Kinect AAL rendszer előzetes tervét, megfogalmazva a problémafelvetést, kitérve a megoldandó feladatokra.

A negyedik fejezetben mutatom be a Kinect AAL rendszer komponensek fejlesztésének lépéseit, elemezve a tervezés és fejlesztés közben felmerülő fontosabb kérdéseket.

Az ötödik fejezetben bemutatom a Kinect AAL szoftver felépítését, és működését, kitérve a fontosabb tervezési megfontolásokra, majd ismertetem az egyes komponenseket megvalósító osztályokat.

A hatodik fejezetben tesztelem a végső Kinect AAL rendszert. Alkalmas tesztekkel demonstrálom az elkészült szoftver komponenseket, és értékelem az eredményeket.

A diplomamunka végén összefoglalom az elvégzett munkát és tapasztalatokat, majd értékelem a teljes rendszert. Zárszóként javaslatokat adok a Kinect AAL rendszer bővítésére és javítására.

Célkitűzés

A munkám célja a Microsoft Kinect szenzor részletes bemutatása, és a felhasználás lehetőségeinek feltérképezése AAL alkalmazásokhoz. Bemutatásra kerül maga a szenzor, minden technikai specifikációja, publikációk illetve az általam végzett tesztek alapján egy általános kép a szenzor képességeiről.

Munkám során be fogom mutatni az AAL és ADL problémakörét, a legfontosabb feladatokat, és az azokat megoldó meglévő megoldásokat, összehasonlítva a Kinect-tel való megoldás lehetőségeivel.

Célom a Kinect szenzor ismeretében egy olyan alkalmazás kifejlesztése, mely képes megoldani az AAL problémakör egyik feladatát. Elsősorban a Kinect mélységérzékelő infra projektorának és kamerájának képességeire támaszkodó megoldást mutatok be, ami a későbbiekben kiegészíthető a Kinect audió szenzorral, más szenzorokkal, vagy akár több Kinect szenzorral is. A rendszer képes lesz egy szobát és egy benne élő embert modellezni, és akár több alapvető AAL feladatot megoldani (például elesés detektálás, gesztusfelismerés).

A célom egy olyan rendszer kiépítése, és használható célszoftver fejlesztése, mely a Kinect használatával képes:

- felismerni az emberi alakokat, azokat tartósan követni az adott térrészben
- alapvető aktivitások, események közül, néhány felismerésére:
    - be- és kilépés a szobából
    - elesés
    - ivás, egyéb gesztus
- az aktivitások alapján reagálni a kritikus szituációkra

Az elkészült rendszer tervezésekor és megvalósításakor célom az adott részfeladatokat megoldó legpraktikusabb megoldások megtalálása, de nem célom az egyes részfeladatokhoz használható algoritmusok vagy létező megoldások részletes összehasonlítása, vagy elemzése.

1.Az AAL témakör

Az öregedő társadalom rohamos növekedése egyre nagyobb igényt mutat az olyan megfigyelő- és beavatkozó rendszerek kialakítására, melyek képesek az idős emberek *önálló, mindennapi* élhetését támogatni, segíteni. A magyarországi 2001-es népszámlálási statisztika szerint a 60 évesek és idősebbek arány 20,4%, ami 30 éven belül akár 30% fölé is emelkedhet [1]. Egyre szükségszerűbb az olyan rendszerek bevezetése, melyek költséghatékonyan, és az idősek jogait és igényeit szem előtt tartva képesek a fontosabb - gondozók által sokszor nem megoldható - feladatokat megoldani, mint például az elesés detektálás, szokatlan mozdulatok, cselekvéssorozatok detektálása (rutintól való szokatlan eltérések). Mindezt úgy, hogy minimálisan, vagy egyáltalán ne avatkozzanak be az élőkörnyezetbe.

1.1 AAL (Ambient Assisted Living)

Az AAL célja, hogy informatikai és kommunikációs technológiák segítségével javítsa az idős, vagy valamilyen kognitív funkciók hiányában szenvedő emberek életszínvonalát azáltal, hogy biztonságosabb, kényelmesebb és önállóbb életet biztosít. Alapvető feladati közé tartoznak:

- minél jobban kinyújtani azt az időtartamot, amíg az idős vagy korlátozott emberek a saját otthonukban élhetnek, növelve ezzel önállóságukat, önbizalmukat, és mobilitásukat
- idős vagy korlátozott emberek egészségügyi és funkcionális támogatása (levenni a terhet az idős emberekről; kognitív funkciók hiánya)
- egészségesebb, komfortosabb életmód biztosítása
- biztonság növelése beteg embereknél, kritikus helyzetekre való felkészülés
- a család, a gondozók és az egészségügy segítése, tehermentesítése, és plusz információk szolgáltatása a biztonságosabb, önállóbb, egészségesebb életvitelhez
- az öregedő társadalmakban előforduló problémák kezelésére fordított erőforrások jobb kihasználása, hatékonyság növelése

Az AAL (Ambient Assisted Living) (és a később bemutatott ADL) alapfeladati közé tartozik, hogy valamilyen módon képesek legyünk egy adott térben (ház, szoba) a megfigyelni kívánt embereket észlelni, azok mindennapi aktivitásairól valamilyen módon olyan releváns adatokat gyűjteni, amik segítségével támogatni, segíteni tudjuk őket, kritikus helyzetekben pedig a megfelelő módon beavatkozni, jelezni. Ehhez legtöbbször különböző szenzorok hálózatát használják, melyek sok esetben kamerákat is tartalmaznak. A kamerás rendszerek legnagyobb hátránya, hogy a személyi jogokat és a magánszférát sok esetben sértik (hiszen 24 órás kamera felvétel készül a megfigyelt alanyokról), legtöbbször – érthető módon - az idős emberek kellemetlenül érzik magukat, ha tudatában vannak annak, hogy megfigyelés alatt állnak. Ezt a problémát elemezik ambiens környezetben élő emberek bevonásával a [2] munkában. Érdekes módon rávilágítanak az AAL témakört átható problémára, miszerint az idősek (és általában minden ember) szeretnék a modern technológiák nyújtotta előnyöket kihasználni (lehetőleg minél észrevehetetlenebb módon, automatikusan), de nem szeretnének semmilyen módon megfigyelve lenni. Ez a kettősség jellemzi az összes AAL rendszert, de jól interpretált megoldással kiküszöbölhető a legtöbb ilyen probléma. Sok kutató és felhasználó véleménye szerint az AAL rendszerek negatívan befolyásolják a szociális kapcsolatokat, és a rokonok törődését és látogatásait fogják kiváltani. Ezért fontos hangsúlyozni, hogy a meglévő rendszerek sem, és a Kinect által megvalósított AAL megoldás sem egy teljes „ellátó rendszer”, hanem egy olyan monitorozó rendszert valósít meg, amely képes az emberi gondozást plusz információkkal és funkciókkal kiegészíteni, amik más módszerekkel nem biztosíthatóak, továbbá képes kritikus helyzetekben azonnali visszajelzést küldeni a gondozóknak. Ezt legegyszerűbben egy konkrét példán keresztül lehet szemléltetni: ha egy idős embert akár napi 2-3 órában is rendszeresen látogatnak, a hirtelen eleséseket, szokatlan időkben való tartós mozdulatlanságokat (pl. ebédlőasztalnál mozdulatlanul ül, miközben a főzőlap melegszik) csak teljes (24 órás) megfigyelés mellett lehet észlelni. Ehhez viszont az emberi erőforrás önmagában nem elegendő, de információs technológiák bevonásával a kockázat csökkenthető.

Az európai unióban az AAL Joint Programme (AAL JP) foglalkozik aktívan az idős, vagy korlátozott emberek segítését megoldó projektek összegyűjtésével, támogatásával. Rendszeresen rendeznek konferenciákat, és publikálják az aktuálisan futó projekteket [3]. A 2011-es katalógusban publikált rendszerek legtöbbje összetett szenzorfúzión és célhardverek használatán alapszik. Az AAL problémák megoldására készülő alkalmazások habár már nem gyerekcipőben járnak, de még korántsem érték el a technológia, és az igények csúcsát. A Berlinben évente megrendezett AAL kongresszus alkalmára eddig két könyv jelent meg,

melyek közül a 2012-es kiadvány ismerteti a legújabb elméleteket, eljárásokat, technológiákat és alkalmazásokat az AAL problémakörben [4], a könyv első fejezetében [5] pedig speciálisan csak video, hang és 3D szenzorok adatait használó alkalmazást mutatnak be, mely egy valósidejű, vészhelyzet jelző rendszert valósít meg. A legújabb alkalmazásokról szóló cikkek és bemutatók az előbb említett 2012-es kiadványban jelentek meg, január végén, így alig 3 hónaposak. Általánosan is elmondható, hogy az AAL témakörben a legfrissebb információk és az AAL alkalmazások technológiája szinte napról napra változik, így a legtöbb forrás és beszámoló csak az interneten keresztül érhető el, sokszor nem is hivatalos publikációként. Habár az AAL alapfogalmai és problémaköre viszonylag jól körülhatárolható, mégsem lehet egységes képet kialakítani, hiszen az interaktív szórakoztató szoftvereken keresztül, a teljesen autonóm megfigyelőkön át, a robotikus segédeszközökig minden beletartozik.

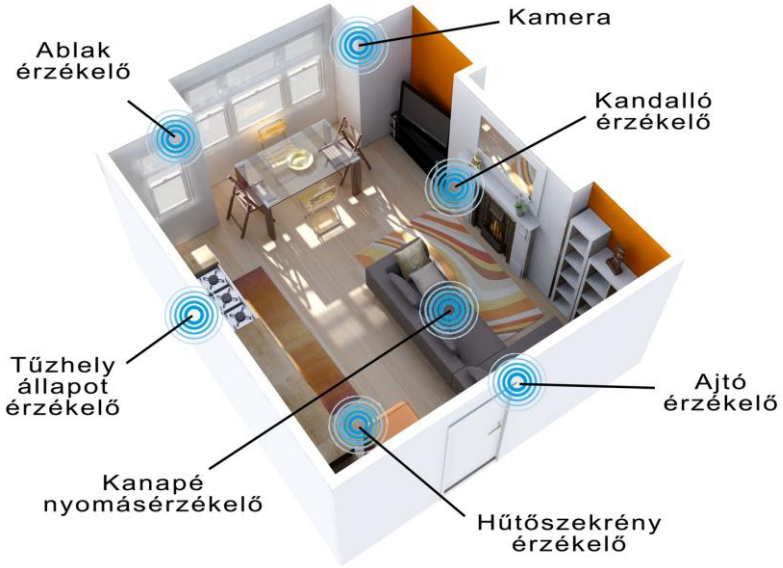
1.1.1 Az AAL legfontosabb feladatai

Az AAL legfontosabb feladatai közé tartozik:

- lokáció meghatározás (ember helye, helyzete a térben)
- elesés detektálás
- élelmiszer- és italfogyasztás monitorozása
- gyógyszerszedés monitorozása
- kritikus helyzetek azonosítása (pl. főzőlap melegedése alvás közben; ablak nyitva hagyása távozáskor)
- cselekvések azonosítása, diszkriminálása, és cselekvési minták elemzése
- „szórakoztatás” (pl. társalgó robot)
- egészség biztosítás, rehabilitáció

A lokáció meghatározás a legegyszerűbb problémák közé tartozik, és számos megoldás létezik már ennek megoldására. Az elesés detektálás egy kritikus szituáció felismerését és kezelését jelenti, emiatt nagy biztonsággal kell tudni meghatározni (specificitás maximalizálása), és a téves pozitív esetek számát is minimálisra kell csökkenteni (érzékenység maximalizálása). Ezt a problémát már számos alkalmazással megoldották, teljesen különböző szenzorokat használva (kamera, gyorsulásmérők, mozgásérzékelő), de az első benyomás ellenére ez egy nehezen megoldható probléma, elsősorban az esemény bekövetkezésének ismeretlen helyzete miatt.

A legnehezebb feladatok közé az élelmiszer, ital és gyógyszer fogyasztás monitorozása tartozik, hiszen ezeket csak összetett, vagy nagyon speciális szenzorok segítségével lehetne megoldani. A gyógyszerszedés monitorozására több megoldási terv is készült, melyek egyike a kézfej mozgásának megfigyelése (amennyiben a gyógyszerek egy adott helyen vannak). A probléma így redukálódik a gyógyszer elvételének, majd bevételének mozdulatsorának azonosítására, de ez még mindig számos problémát megoldatlanul hagy, hiszen nem biztosítható, hogy a helyes gyógyszert, a megfelelő mennyiségben vette be, illetve a „rosszindulatú" alany könnyen megtéveszthetti a rendszert.



1.1. ábra Általános AAL rendszer; forrás: [53]

A kritikus helyzetek felismerése a legtöbb esetben több szenzor használatát igényli az események és helyzetek térbeli és jellegbeli különbségei miatt. Az 1.1. ábrán egy 1 szobát lefedő általános AAL rendszer látható. Megfigyelhető a szenzorok teljesen heterogén összetétele, a térbeli elhelyezésük és funkcióik teljesen különböznek, ugyanakkor a rendszer egészét tekintve kiegészítik egymást.

Az AAL feladatok közül a „szórakoztatás" azon feladatokat foglalja megába, mely az idős emberek szociális életét igyekszik megtartani azzal, hogy a megszokott környezetben interaktív kommunikációs interfészeket alakítanak ki. Ennek segítségével az idős emberek könnyebben tudnak kapcsolatot tartani a hozzátartozókkal, vagy akár egy mesterséges intelligenciával is kommunikálhatnak, aki további információkat tud ez által szolgáltatni a megfigyelő rendszernek.

Az egészség biztosítás és rehabilitáció egy olyan AAL alapfeladat, amely megoldására készülő alkalmazások a felépülőben lévő betegeket segítik, vagy az egészségesebb életmódot követni kívánó emberek igényeit elégítik ki. Általában a céljuk, hogy a rendszer képes legyen torna és rehabilitációs programokat végrehajtatni az emberrel és monitorozni azok végrehajtásának módját és minőségét.

1.1.2 AAL rendszer tervezése

A témában való elmélyüléshez szükséges megérteni az AAL rendszer tervezésének aspektusait általános esetekben, és megismerni a rendszer kialakításában és használatában résztvevő ágenseket (emberi, gépi, vagy virtuális entitás). Több szemszögből kell a témát megvizsgálni:

- végfelhasználói szemszög (igények, kritériumok, korlátozások)
- tervezői szemszög (technológiák, környezet tulajdonságai)
- kivitelezői szemszög (környezet, kialakítás)

A felhasználói igények közvetlen definiálása helyett célszerű a *célközönséget megjelölni*, tehát meghatározni, hogy milyen korú, mentális állapotú, kognitív képességekkel rendelkező emberek tartoznak a célcsoportba. A felhasználó szemszögéből fontos figyelembe venni, hogy milyen helyen lesz alkalmazva a rendszer, így milyen módon kell a *magánszférát* figyelembe venni (nappali, hálószoba, publikus hely).

A tervezőnek és kivitelezőnek fontos számba venni a telepítési környezetet, hogy a szenzorrendszer elrejtet (átlátszó) legyen, *épüljön be teljesen a meglévő környezetbe*, vagy az adott térrész legyen már eleve úgy megépítve, hogy fel legyen készítve a rendszer működtetésére. Általában az emberek meg akarják tartani meglévő élőkörnyezetüket, és minden látható módosítás nélkül kell a szenzorokat elhelyezni. A technológiák és szenzorok számbavétele a tervezők részéről tehát azért, mert ezek nagyban befolyásolják a végső rendszer képességeit és a környezetbe ágyazhatóság mértékét. Az 1.1 ábrán látható szenzorok közül a kamerát kivéve mindegyik könnyen elrejthető a felhasználó elől. A kamera is beépíthető bútorokba, amint az a később elemzett egyik AAL alkalmazásban is látható, de így pozíciója nagyban függ a helyszín kialakításától, és így funkcionalitása is módosulhat.

Az AAL alkalmazásokban a célszemély perzisztens (folyamatos) figyelése miatt nagy adatmennyiség generálódik. Az adatok feldolgozásánál ezért arra kell törekedni, hogy minél alacsonyabb szinten (már a szenzorban) *végezzünk előszűrést*, szegmentálást, és a

szenzoradatokat feldolgozó következtető rendszer lehetőleg már *kompakt események sorozatával* dolgozzon (magasabb absztrakciós szinten).

Ahogy a fejezet elején bemutattam, napjainkban is számos megoldás készül az AAL problémák megoldására. Az alkalmazások funkcióit és algoritmusait tekintve is nagyban eltérnek egymástól, de közös bennük az AAL témakör fogalomtára, és általában a megoldandó feladatok is jól definiálhatók. Erre építve számos alkalmazás [6] nem egy adott problémát old meg, hanem egy platformot nyújt (middleware) más alkalmazások integrálásához, összekapcsolásához és fejlesztéséhez az AAL témakörön belül. Ezek alapja a közös „nyelv”, tehát komoly hangsúlyt fektetnek az alkalmazások által használt fogalmak egységesítésére, ezáltal egy közös ontológiát kialakítva.

1.1.3 Ambiens tér

Ambiens térnek nevezzük azt a környezetet, amiben az ágensek élnek, mozognak. A megfigyelt tér általában *nem hozzáférhető*, tehát nem áll rendelkezésünkre minden pillanatban minden információ, csak a szenzorokra lehet hagyatkozni. Egy emberek által lakott környezet *dinamikus*, hiszen tetszésük szerint mozognak benne, változhatnak a tereptárgyak is, és az ember által végzett cselekvések gyakorlatilag a tér minden paraméterét képesek befolyásolni, függetlenül a megfigyelő rendszertől (ablakok nyitása, redőny lehúzása, kisebb bútorok mozgatása). A térben a folyamatok mindig folytonosak (fizikailag és temporálisan is), így a megfigyelés is folytonos kell, hogy legyen, de az adatokat események sorozatára érdemes alakítani, a könnyebb kezelhetőség érdekében.

1.1.4 Ágensek az ambiens térben

A humán ágens (ember) a fizikai térben mozog, cselekszik, és elképzelhető, hogy a fizikai térben más ágensek (pl. robotok) is jelen vannak. Az emberi ágens a teret saját, nem ismert céljaira használja, ebből kifolyólag a legnagyobb nehézséget az okozza, hogy a megfigyelő rendszer nem tud információt szerezni az *ember szándékairól* (hacsak ez az információ csere nem történik meg explicit módon), terveiről. A humán ágensek (ellentétben a virtuális ágensekkel, robotokkal) *nem szisztematikusan terveznek*, sokszor ad-hoc módon cselekszenek, legalábbis a megfigyelő rendszer számára. Egy emberi ágenstől elvárható, hogy bizonyos interfészeket kezeljen, de ezek fajtája mindig az adott szituációtól,

célcsoporttól függ, hiszen például a halláskárosultakat nem célszerű hangjelzéssel egy esetleges veszélyhelyzetre figyelmeztetni. Az AAL rendszer tervezésekor a humán ágensek kooperációját nem szabad specifikációs tényezőnek tekinteni. A felhasználó alap esetben *nem professzionális*, tehát csak jól definiált, jól megválasztott interfészeken keresztül avatkozhat be a rendszerbe. Mindig azt kell feltételezni, hogy a felhasználó tudatosan, vagy akaratlanul, de „rosszindulatú", tehát minden szélsőséges helyzetet számításba kell venni.

1.2 ADL (Activities of Daily Living)

Az AAL témakör másik leghangsúlyosabb feladata a napi rutinhoz tartozó cselekvések (ADL) felismerése, a cselekvések alapján, autonóm módon a napi rutin megállapítása, és a szokatlan események (outlier) azonosítása. A cselekvések diszkriminálása és egyedi azonosítása önmagában nehéz feladat. A legtöbb cselekvésnél vagy a vizuális információra kell hagyatkozni (pl. kamera, Kinect), vagy több különböző szenzor adataiból kell következtetni. Például egy „reggelizés cselekvést" azonosítani lehet a pozíció, és a kézmozgás alapján, vagy következtetni a különböző háztartási eszközök állapotából (mikró, főzőlap be- és kikapcsolása), és a pozíció és aktvitás (pl. ülés asztalnál) jellegéből.

Az ADL olyan aktivitások összességét foglalja magába, melyeket benti vagy kinti környezetben, ismételve (általában napi rutin szerűen, sokszor, periodikusan) hajtanak végre az emberek. Ezek közé tartozik (a teljesség igénye nélkül):

- az étkezés
- a személyes higiénia (fogmosás, zuhanyzás, WC)
- a takarítás, házimunka
- az alvás
- a hely- és helyzetváltoztatás (felkelés az ágyból, leülés az asztalhoz)

Két csoportra lehet osztani ezeket az aktivitásokat: rutinszerű és kritikus. Az ADL-t megfigyelő rendszerek legnagyobb problémája (hasonlóan az AAL témakörnél leírtakhoz), hogy a legtöbb aktivitás nehezen jósolható, mind idejét, mind jellegét tekintve, és sokszor nehezen észlelhető. A rutinszerű aktivitásokat az idősebb emberek általában sokkal pontosabban betartják, mint a fiatalabb korosztály, így az ismétlődő szekvenciákban való eltérések is fontos információt hordoznak. Például új gyógyszer bevezetése után a székelési

és alvási aktivitások változása sokszor a gyógyszer miatt váltódik ki, ami fontos információ lehet az orvosok számára.

Több féle modellt használnak az ADL aktivitások leírására, modellezésére, ilyen például Roper–Logan–Tierney model of nursing [7], de számos más könyv is kiadásra került a témában. Emellett kialakítottak olyan mérőszámokat (Katz ADL Index, Lawton iADL Index), melyek segítségével egy ember önállóságra való képessége jellemezhető. [8] Katz hat szempont (és kritérium) alapján pontozza az egyéneket: fürdés, öltözködés, székelés, helyváltoztatás, önuralom (vizelés, székelés), étkezés. Lawton az instrumentális (eszközhasználati) aktivitásokat osztályozza: telefonhasználat, bevásárlás, főzés, házimunka, mosás, utazás módja, gyógyszerszedés, pénzügyek kezelése. A mérőszámok meghatározása egy adott emberre segít a személyre szabott igények meghatározásában.

Az ADL alapvetően az AAL alkalmazások alapját (háttértudását) képezik, hiszen ezeknek az aktivitásoknak az ismerete és statisztikái az AAL rendszerek heurisztikáiként működnek. Az AAL rendszerek célja az ADL aktivitások végrehajtásának segítése és azok ellenőrzése, továbbá a kritikus aktivitások (pl. elesés detektálás) felismerése.

2. **Kinect szenzor**

A Kinect a Microsoft által fejlesztett kamera és mélységérzékelő szenzorikus eszköz, mely a Microsoft Xbox 360 játékkonzol egyik kiegészítőjeként, 2010. november 4-én került elérhetővé a felhasználók számára. Lehetővé teszi, hogy a konzollal és az általa futtatott programokkal teljesen újszerű módon tudjunk interakcióba lépni. A Kinect a teljes testünk és a hangok (hangutasítások) megfigyelésével képes egy sokkal felhasználóbarátabb, természetesebb felhasználói interfészt (Natural Interaction) nyújtani. Számos fejlesztő rögtön meglátta benne a potenciális lehetőségeket, és a megnövekedett igényeknek és nyomásnak engedve a Kinect magját képező szenzorok gyártói (PrimeSense), illetve a Microsoft is lehetővé tette az eszközzel való fejlesztést PC-re is.

2.1. ábra Kinect szenzor

A szenzor egy RGB kamerával, hangérzékelővel, és egy mélységtérképet szolgáltató IR (infravörös) kamerával van ellátva. Az integrált megoldásnak köszönhetően egyszerűen lehet objektumokat (elsősorban emberi alakokat, végtagokat) azonosítani a 3D térben, ami leegyszerűsíti a sok témakörben (elsősorban képfeldolgozásban) felmerülő igen komplex problémát: jellegzetes pontokat 3D térben azonosítani és tartósan követni. A hangérzékelő (6 mikrofon) lehetővé teszi a hangvezérlést, és a hangforrások irányának meghatározását a szenzorhoz viszonyítva, de ezek a funkciók a felhasználói programoktól nagyban függenek. A Kinect pontos technikai specifikációját az 1. függelék tartalmazza, a továbbiakban nem részletezem.

2.1 A szenzor működése és képességei

A Kinect szenzor alapja az IR kamera együttes, mely egy infravörös fényt kibocsájtó fényforrásból, és egy infravörös fényt érzékelő kamerából áll. Más IR fényt használó mélységérzékelők általában azon az egyszerűsített fizikai modellen alapszanak, mely a kisugárzott fény által megtett utat (távolság a szenzortól) a kibocsátás és érzékelés között eltelt idő és a fénysebesség által határozzák meg.

A Kinect -ben használt PrimeSense chip más megoldást használ [9,10]. Ennek lényege, hogy az IR kibocsátó egy pszeudorandom mintázatot (sűrű ponthalmazt) vetít ki a megfigyelendő objektumra (lásd 2.2 ábra). A kivetített mintázatokat először ismert, de különböző távolságban lévő referencia felületekre vetítik ki (egymás után, egyenként), majd ezeket a képeket, és a mintázatokban lévő pontok távolságát elmentik referencia adatokként. Az ismeretlen távolságban lévő objektumra ezután ugyan azt a pszeudorandom mintázatot vetítik ki. Az előzőleg referencia felületekre kivetített mintázatok közül, különböző korrelációs technikákat alkalmazva, kiválasztják a legvalószínűbb referencia felületet (felületeket), és az ahhoz eltárolt referenciapontok távolságát. Ezután az ismeretlen objektumra (objektumokra) kivetített mintázat pontjait keresztkorrelációs technikákkal megfeleltetik a referencia felület pontjainak, és a pontok távolságából meghatározható a referencia felület és az ismeretlen objektum felület pontjainak z-tengely (3. dimenzió) menti különbsége. Az így meghatározott mélységkülönbségek képezik tehát a mélységkép (mélység térkép) alapját.

2.2. ábra Kinect által kibocsátott infravörös fény (pontháló); ábra forrása: [54]

A szenzor előnye, hogy az RGB (színes) kamerakép mellé szolgáltatja a mélységképet, ami a felületek, mozgó objektumok azonosítását nagyban megkönnyíti. Már ezek az adatok is sokkal több információt szolgáltatnak, mint az egyszerű RGB kamerák (vagy akár több

kamera együttes használata), de a Kinect képességeit nagymértékben kiterjeszti az a technológia, mely ezeknek az alacsonyszintű adatoknak a felhasználásával további magasabb szintű adatot szolgáltat. Az absztrakciós szint eggyel magasabb fokán már az emberek felismerését, kézfejek azonosítását, és az emberre illeszthető, a gerincet, végtagokat és fejet szimuláló úgynevezett szkeletont (angol *skeleton*, azaz csontváz) értjük. Jelenleg két rendszer is létezik: OpenNI és a Kinect SDK, melyeket a következő szakaszban ismertetek.

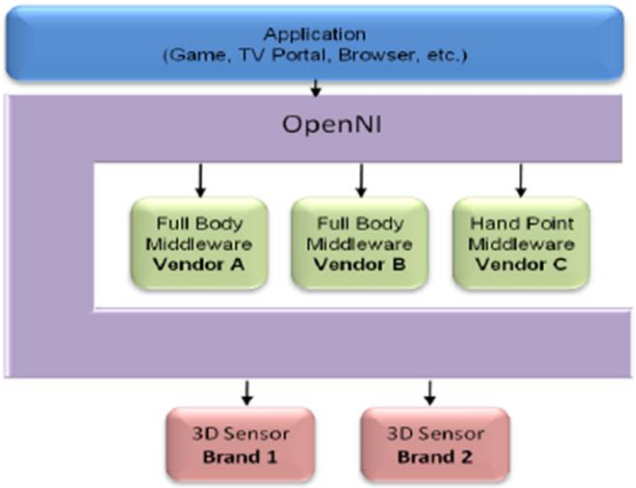
2.2 Szoftver keretrendszerek

Röviden bemutatom a két szoftver keretrendszert, melyek magát a szenzort képesek működtetni (driver), és a szenzor által mért adatokat (mélység térkép, színes kép, audió jel) felhasználható formára transzformálni, továbbá elérhetővé tenni olyan származtatott adatokat, mint például a szkeleton.

2.2.1 OpenNI és NITE

Az NI (Natural Interaction) egy olyan elgondolás, ami az ember-gép interakciót elsősorban a látásra és hallásra akarja alapozni. Alapvető céljai közé tartozik a hangfelismerés és hangvezérlés, kézjelek felismerése, vagy akár a teljes emberi alakfelismerés, követés és interpretáció további felhasználásra. Az OpenNI egy cross-platform, több programozási nyelven elérhető (C, C++, C#, Java) framework, ami különböző API-kat (*Application Programming Interface*, azaz alkalmazásprogramozási felület) szolgáltat az olyan alkalmazások készítéséhez, amik az NI (Natural Interaction) alapelvekre épülnek. Az API-kat interfészeken keresztül lehet elérni, amik azt a célt szolgálják, hogy egységes felületen lehessen kommunikálni a hardver elemekkel, amik „látnak és hallanak”, és a szenzoradatokat feldolgozó middleware-rel (pl. NITE). A middleware elemek olyan szoftverkomponensek, amik a szenzorból érkező adatokat feldolgozzák, és olyan magasabb szintű információkat szolgáltatnak a nyers adatokon felül, mint például egy felismert kézfej pozíciója a 3D térben, vagy az emberi alakra illesztett szkeleton. A koncepciót a 2.3-as ábra mutatja be. A legalsó réteg reprezentálja a hardver réteget, ide tartozik maga a KINECT eszköz is. A középső (zöld) réteg azokat a middleware (pl. NITE, *Natural Interaction Technology for End-user*) alkalmazásokat jelöli, amik kielégítik az OpenNI API-t és valamilyen hardverre támaszkodva magasabb szintű adatokat szolgáltatnak (pl. nyers pixel

adatok helyett egy kézfej pozícióját a térben). Az ezeket összekötő réteg maga az OpenNI, mely ez által egy egységes interfészt kínál a legfelsőbb szinten elhelyezkedő felhasználói alkalmazásnak.



2.3. ábra OpenNI architektúra; ábra forrása: [15]

A NITE az OpenNI-vel használható egyik modul (middleware), mely képes az OpenNI által szolgáltatott (Kinect alacsonyszintű adatfolyamából kinyert) magasabb szintű adatok alapján az emberi „csontváz” (szkeleton) emberi alakokra illesztésére, követésére, továbbá kézfejkövetésre és egyszerű gesztusfelismerésre [11]. Egy *gesztus* minden olyan mozgást magába foglal, ami a végtagokkal és testtel végzett egyszeri cselekvés. A NITE gesztusfelismerése korlátozott képességű, csak kézfejjel végzett mozgást képes felismerni, többek között: integetést, nyomó mozdulatot (kézfej előre-hátra mozgatása) vagy oldalra léptető mozdulatot (kézfej oldalra mozgatása). Saját gesztusok felismerésére nem képes, így ennek megtervezése és implementálása egy AAL rendszer esetében a végfelhasználói programban szükséges.

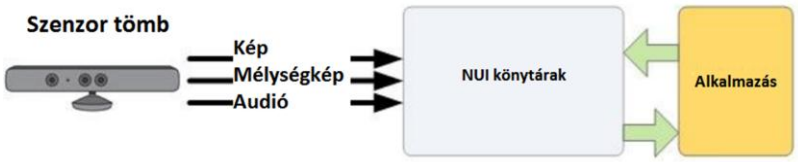
2.2.2 Kinect SDK

A Microsoft által 2011 júliusában kiadott keretrendszer egy sokkal kompaktabb API-t nyújt a Kinect funkcióinak kihasználására. Ez egy zárt rendszer abban az értelemben, hogy nem komponens alapú, csak a Kinect-tel kompatibilis, és nem is nyílt forráskódú, illetve használható fel szabadon (az ezzel elkészült alkalmazásokra a Microsoft igényt tarthat). Jelenleg második béta állapotban van, de már képes a következő funkciókat biztosítani:

- mélységkép, RGB kép és audió adatok kinyerése

- szkeleton illesztése (NUI)

A belső működése (adatfolyamhoz való hozzáférés) nem annyira befolyásolható, mint az OpenNI esetében [12], de azt tudni lehet, hogy a szkeleton illesztés és követés más algoritmust használ [13] mint az OpenNI-hez használt middleware, a NITE.



2.4. ábra Kinect SDK architektúra; ábra forrása: [12]

2.3 Szoftver keretrendszerek tesztelése

Első lépésként mindkét rendszerrel teszteltem a Kinect-et, hogy a gyakorlatban is egy felületes, előzetes képet kaphassunk a keretrendszerek képességeiről. Alapvető teszteseteket állítottam össze, melyek nagyjából lefedik azokat az eseteket, amik egy Kinect szenzort használó AAL rendszernél előfordulhatnak. A két rendszert egyenként teszteltem egy Windows 7 operációs rendszeren (csak ezen a konfiguráción futtatható mind a kettő). Mindkét rendszer nem lehet egyszerre feltelepítve, a driverek összeakadása miatt, ezért a két rendszer képességeit kombinálni nem lehet. A teszteléshez a keretrendszerekhez mellékelt példaprogramokat használtam, melyek képesek az emberi alakok felismerésére és követésére, illetve a szkeleton illesztésére is (külön programként). A tesztelt keretrendszerek és aktuális verzióik:

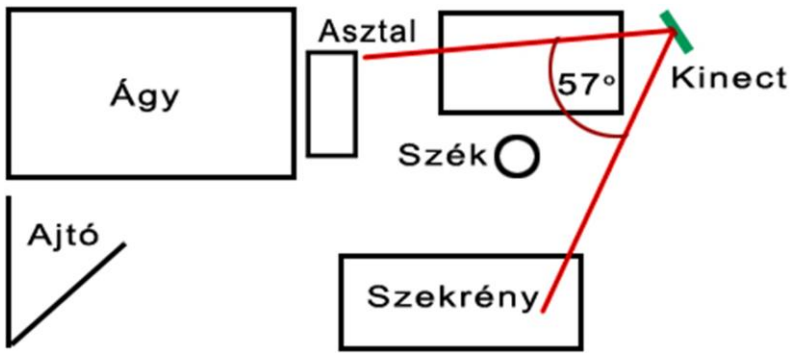
- Kinect SDK: v.1.0 beta2 x86
- OpenNI: 1.5.2.23; NITE: 1.5.2.21

2.3.1 Tesztkörnyezetek

Minden tesztet két külön teszthelyiségben végeztem el. A szobák méretei, és a látószög úgy lettek beállítva, hogy ha egy emberi alak az ajtóban áll, akkor a szenzor éppen azonosítani tudja, de a 4,5-5m-es távolságon kívül már nem. Az alább bemutatott tesztkörnyezeteket használom majd a végső rendszer teszteléséhez is.

Tesztszoba 1

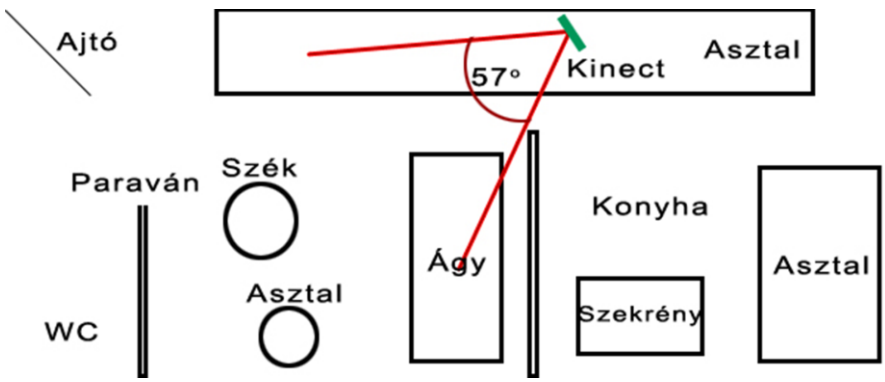
- Valós lakóhely (szoba), ahol a Kinect látóterében van egy asztal, ágy, szekrény, és a bejárati ajtó.
- Méret: 5m x 3m
- Kinect szenzor elhelyezése: 170cm magasságban, a 2.5 ábrán látható pozícióban



2.5. ábra Teszt szoba 1

Tesztszoba 2 (Ambiens Intelligens Labor, BME MIT)

- Laborkörnyezetben szimulált lakóhelyiség, ahol a Kinect látóterében van egy szék, ágy, szekrény, asztal, a bejárati ajtó, és egy falat, és külön helyiséget (wc) szimuláló paravánnal elválasztott helyiség.
- Méret: 6m x 4m
- Kinect szenzor elhelyezése: 170cm magasságban, a 2.6 ábrán látható pozícióban

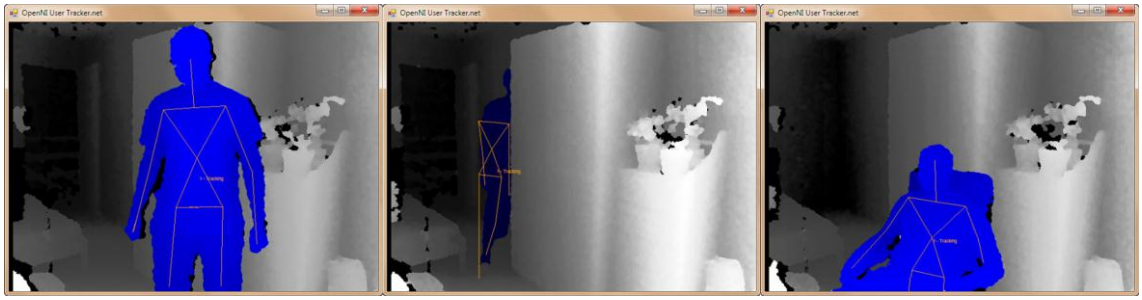


2.6. ábra Teszt szoba 2 (Ambiens Intelligencia labor, BME MIT)

A szenzor szűk látószöge miatt a szobáknak csak egy részét fedi le a szenzor. Törekedtem a szenzor optimális elhelyezésére, hogy a főbb bútorok (ágy, szék) a látótérben maradjanak. Mindkét helyiségben van elég hely a térben való mozgásra, az elesések szimulálására, és néhány ADL aktivitás elvégzéséhez (asztalhoz ülés, székbe ülés, ágyra fekvés… stb.).

2.3.2 Az elvégzett tesztek és értékelésük

A tesztprogramok olyan információkat is közölnek a felhasználóval, hogy mikor sikerült bemérni az ember alakot, és mikor működik a szkeleton illesztése, így ezek alapján elvégezhető a tesztek kvalitatív értékelése. A tesztelés során egy ember mozgott a tesztszobákban, míg magam végeztem a tesztprogramok kimenetének elemzését, pusztán a vizuális információra és a tesztprogramok kimenetére hagyatkozva, ahogy ez a 2.7 ábrán is látszódik.



2.7. ábra OpenNI tesztprogram kimenete ideális, félig kitakart helyzetben és székben üléskor. A kék szín jelzi az embernek felismert pixeleket, míg a sárga vonalak az illesztett szkeletont.

A következőkben csak az első pár tesztesetet mutatom be, a többi részletes elemzés a 2. függelékben olvasható, és a DVD mellékleten található *OpenNI_test.mp4* videón is megtekinthető.

1. Kalibrálás, távolságok tesztelése
   a) OpenNI: 0,5m –en belül is képes volt csak a felsőtest alapján azonosítani az emberi alakot. 5m-en kívül a felismert emberi alak követését elveszti, de a közeledő alakot rögtön azonosítja.
   b) Kinect SDK: Csak felsőtest alapján nehezen tudta azonosítani az emberi alakot, de különben u.a. mint az OpenNI.

2. Ember felismerés
   a) OpenNI: Érzékenység: 5-10cm-es, lassú mozgástól már felismeri az emberi alakot 2,5m távolságban.
   b) Kinect SDK: u.a. mint az OpenNI.
3. Ember követése karosszékbe ülés közben
   a) OpenNI: Leülés után a két kontúr összeolvadt (lásd 6-os teszt és 2.7 ábra 3. képe). Felállás után a fotelt sokszor megtartotta külön mozgó entitásként (nem háttér), és megpróbált rá szkeletont illeszteni.
   b) Kinect SDK: Hasonlóan az OpenNI-hez, itt is hibásan észlelete az emberi alakot, ha a székben ülve, azzal együtt mozgott, de felállás után szeparálta, és eldobta a székhez tartozó részeket.
4. Ember követése lefekvés közben
   a) OpenNI: Fekvő pozícióban az emberi alak követése pontatlanabb, de a felismerő algoritmus nem veszítette el az emberi alakot. Nem felismert helyzetből kiindulva a fekvő embert egyáltalán nem érzékelte, sem mozgás (forgolódás) közben, sem felüléskor, hanem csak felálláskor észlelete az emberi alakot.
   b) Kinect SDK: u.a. mint az OpenNI.

Az előzetes tesztek alapján az derült ki, hogy számottevő teljesítménybeli különbség nincs a két rendszer között. A két rendszer közötti legnagyobb különbséget a szoftvere, hardver támogatottság és a jelenlegi (2011.11) szoftverek képességei jelentik. A 2.1-es táblázatban foglaltam össze a két rendszer legfontosabb tulajdonságait. Az információk egy része az OpenNI központi fórumáról [14] származik, ahol más kezdő és tapasztalt Kinect fejlesztők osztják meg tapasztalataikat [15,16,17], másik része pedig az elvégzett tesztekből és személyes tapasztalatból származik.

Konklúzió

A két rendszer közül az OpenNI használata javasolt hosszútávra, a későbbi más eszközökkel való kompatibilitás, illetve a nagyobb támogatottság és nyílt forráskódú platform miatt. Elképzelhető, hogy a közeljövőben a Kinect SDK is bővülni fog olyan funkciókkal, amit a NITE nyújt, tehát beépített gesztusfelismerő algoritmusokkal, és modulárisabb felépítéssel. A dolgozatban elvégzett munka célja, hogy a Kinect-tel megoldjuk a kitűzött célfeladatokat, funkciókat, tehát a köztes technológia kiválasztása másodlagos szempont.

Szempont	OpenNI (1.5.2.23)	Kinect SDK (beta2)
Forráskód	nyílt	zárt
Felhasználhatóság	szabadon felhasználható, kereskedelmi célokra is	minden KinectSDK-val létrehozott megoldás publikus használatához a Microsoft-tal kell szerződni
API	általánosabb, moduláris, bővíthető	rögzített, nem moduláris, nem bővíthető
Platform	cross-platform (windows, linux, osx)	windows 7
Hardver	bármi, ami kielégíti az API-t	csak Kinect
Ember és szkeleton felismerés	automatikus, gyors	automatikus, gyors
Gesztusfelismerés	alapvető vezérlési gesztusok	nincs
Mikrofon	használható	használható (irány meghatározás is)
Több szenzor használata	lehetséges	nem lehetséges
Visszafele kompatibilitás	igen (külön verziókezelés lehetséges)	nem (béta állapotban)
Egyéb előnyök	idősor adatok rögzítése, szimulálása	kitakart végtagok következtetése pár másodpercig

2.1 táblázat Szoftver keretrendszerek összehasonlítása

Mindkét keretrendszer nagyjából ugyan olyan képességekkel bír, és a feladatnak nem célja, hogy ezeket a technológiákat mélyre menően összehasonlítsuk, hanem *egy* olyan rendszer kialakítása, amely képes a Kinect-tel a célfeladatot megoldani. Lehetséges, hogy később a köztes technológiát ki kell cserélni, így a szoftver tervezésénél erre is hangsúlyt kell fektetni.

Az OpenNI megbízhatóbban tudta követni az emberi alakot, és képes volt csak a felsőtestet is külön kezelni. A legtöbb helyzetben sokkal stabilabban határozta meg a szkeleton ízületi pontjait, nem volt tapasztalható szignifikáns ugrálás (térbeli illesztési pontatlanság gyors változása) a nem meghatározható pontok esetében.

Az elvégzett tesztekből az derült ki, hogy a Kinect alkalmas lehet az AAL egyes feladatainak megoldására, de mindenkép célszerű kezdetben egy „ideálisabb” környezetben tesztelni, tehát úgy, hogy az emberi alak végig teljes mértékben a látótérben marad. A tesztekből az is kiderült, hogy a kamerának háttal, vagy oldalt álló ember kezeit, amennyiben kitakarásban voltak, nem volt képes követni. Ezt fontos majd számításba venni a különböző AAL feladatok megoldásánál, hiszen gesztusfelismerésnél sokszor a kezek pozíciója a mérvadó.

Az OpenNI dokumentációja alapján [15] az OpenNI képes több szenzor párhuzamos használatára, és a szenzoroktól érkező adatok szinkronizálására is. Ezzel kiküszöbölhető lenne a távolság és látószög miatti korlátozás, illetve a kitakart térrészek sem okoznának szkeleton illesztései problémákat, hiszen az egyik szenzorból érkező adatokat kiválthatnák a másik szenzor adatai, ha az előbbi illesztési pontatlansága egy adott határ fölé esik. A Kinect SDK beta2-ben több szenzor használata még nem támogatott.

A tesztekből továbbá az derült ki, hogy az emberi alak követése nagy pontossággal elvégezhető, de a más mozgó tárgyaktól való szeparálás néha nem működik rendesen. A tereptárgyakkal való sűrű interakció (székbe ülés, székkel való mozgás, ajtó mozgatása, lefekvés és takaró használata) erősen megnehezítik az ember felismerését és követését. Ilyen helyzetekben más heurisztikákra is szüksége lehet (például utolsó látott pozícióban való érzékenyebb megfigyelés). A szkeleton illesztése (általánosan) sokkal pontatlanabb, mint az emberi sziluett azonosítása és követése, de bizonyos fenntartásokkal (hibafaktorok bevezetésével) az ebből származó információ is felhasználható, így a fej és végtagok pozícióját is meg lehet határozni körülbelül 5-10cm-es pontossággal.

A szenzor legnagyobb hátránya a szűk látótér (57^o). A megfigyelhető maximális terület körülbelül $12.4m^2$ (a körcikk területe), amit természetesen a falak tovább csökkentenek. A távolság növekedésével a pontosság is csökken (a Kinect távolságmérő algoritmusából következik), és ez okozza lényegében az emberi alak követés és a szkeleton illesztés elvesztését a kb. 5m-en kívül. Ez a probléma többféle képen is megoldható:

- a látótér szélén lévő embert a Kinect-et mozgató motor segítségével követni kell
- több kamera használata, akár átfedésben lévő látótérrel, de közös intelligenciával.

Az első nagy hátránya, hogy ha bármilyen kalibrálást végzünk, ami a Kinect pozíciójától és beállított látóterétől függ, akkor azt dinamikusan kell változtatni. Mivel munkám során tervezem a szoba egy modelljét elkészíteni egy adott kalibrációhoz, ezért a Kinect mozgatása

nem megoldható (pontosabban új kalibrációt igényelne, ami valós környezetben nem kivitelezhető). A második megoldás egy bonyolultabb logikát igényel, hiszen a két szenzortól érkező adatokat fuzionálni kell. A diplomamunka során a szenzor látóteréből eredő nehézségeket azzal hidalom át, hogy mindig egy ideális pozíciót fogok használni, ami a tesztszobák ábráin is látszódik, és ezen a csökkentett területen végzem el az AAL feladatok tesztelését.

3.KINECT használata AAL alkalmazáshoz

Az 1. fejezetben bemutattam az AAL és ADL témaköröket, és az AAL rendszerek főbb kihívásait. A 2. fejezetben megismertettem az olvasóval a Kinect szenzort, és alap tesztesetek elvégzésével próbáltam az eszköz képességeit feltérképezni.

A Kinect szenzor segítségével követni tudjuk a teljes emberi testet a háromdimenziós térben (pontos 3D adatokkal), ami az AAL alkalmazásokban eddig használt szenzorokhoz képest teljesen újszerű. A célom a szenzornak ezt az erősségét kihasználva egy olyan alkalmazás fejlesztése, mely képes alapvető AAL feladatokat megoldani. Habár sok AAL probléma más szenzorokkal és technológiákkal már megoldott, mégis fontos egy új, potenciális szenzor megvizsgálása a témakörben.

Az alapfunkciókon kívül a Kinect-tel a szórakoztató, rehabilitációs és torna programok használata is megoldható, hiszen erre számos alkalmazás létezik már. Továbbá kiterjesztett és virtuális valóság rendszerekhez, természetes interfész kialakítására, és robotvezérlésre is használják [18,19], amiket AAL alkalmazásokba beépítve a megfigyelt emberek és a megfigyelő rendszerek között kommunikációs csatornák kiterjeszthetők lennének.

3.1 Problémafelvetés

Az alap probléma a következőképpen fogalmazható meg. Egy zárt térrészben (szoba) egyedül mozgó embert kell megfigyelni, és a kritikus eseményeket (pl. elesés) és bizonyos gesztusokat, aktivitásokat (pl. ivás) felismerni. A rendszer a Kinect szenzort használja elsődleges adatforrásként, de rendelkezik a térrész, és esetleg a megfigyelt ember, egy modelljével. A problémát megoldó alkalmazásnak képesnek kell lennie az előre definiált eseményeket felismerni, és a rendszerben rögzíteni, vagy valós időben visszajelezni. Továbbá be kell tartani az AAL alkalmazásoknál leírt alapvető igényt, a magánszféra tiszteletben tartását. Nem szükséges az események naplózása, sem hosszú távú következtetések levonása az eseménysorozatok alapján, de a rendszer teszteléséhez szükséges eszközök és interfész biztosítása igen.

3.2 Meglévő AAL rendszerek

A következő szakaszban néhány, AAL feladatokat megoldó alkalmazást mutatok be, egy általános körképet adva a megoldott és megoldatlan feladatokról, azok nehézségeiről, tehát a technológia mai állásáról. Az AAL katalógusban [3] számos rendszer kerül bemutatásra, amik képesek különféle AAL feladatok megoldására, de ezek legtöbbje szenzorfúziót és speciális célhardvereket használ. Elsősorban olyan összetettebb rendszereket vizsgáltam meg, melyek kameraképek, mikrofonok és 3D szenzorok használatával (audiovizuális szenzorokkal) próbálnak meg AAL feladatokat megoldani, ezzel is feltérképezve egy Kinect szenzort használó AAL alkalmazás létjogosultságát. A megvizsgált rendszereket a Kinect szenzor képességeit szem előtt tartva kritikusan elemeztem, hogy egy előzetes képet kapjunk a Kinect szenzort használó (elképzelt) AAL rendszer képességeiről.

A real-time system for in home activity monitoring [20]

A munka egy általános AAL rendszert mutat be egy idős ember alap aktivitásainak követésére. A szenzoradatok (videokamera kép) alapján aktivitásanalízist végeznek, amiből további következtetéseket lehet levonni. Nagy hangsúlyt fektetnek a rendszer élőkörnyezetbe épülésére, illetve ügyelnek a magánszféra megóvására, ezért csak sziluetteket használnak, hogy a kamera képe ne sértse a személyiségi jogokat. A megoldás legnagyobb hátránya, hogy RGB kameraképek alapján működik, így változó fényviszonyok mellett (pl. éjjel) nem működik. A képi információ feldolgozása igen összetett, és nehézséget okoz a háttérdetektálás algoritmusának finomhangolása, illetve olyan problémák megoldása, hogy egy mozdulatlan embert egy idő után ne tekintsen háttérnek a rendszer. Az emberi alak felismerése körülbelül 7 FPS-sel (Frames Per Second, azaz képkocka per másodperc) működik, amihez még hozzáadódik a további analizáló algoritmusok futási ideje. Összességében ez a feldolgozási sebesség bizonyos AAL feladatok megoldásánál elfogadható, de hirtelen mozdulatoknál, ahol a hirtelen mozdulat jellege és dinamikája is fontos információ (pl. elesés detektálás), nehezíti a felismerést.

Sebesség adatok alapján következtetnek sokszor az adott ADL aktivitásra, mivel a sziluettekből a finom mozgásokat és végtagokat nehezen lehet kivenni. A rendszer összességében képes sziluettek alapján, két emberre meghatározni, hogy mikor és hol tartózkodtak (hálószoba, konyha, kanapé előtt, hálószoba előtt), és erről statisztikát készíteni.

A Kinect a mélységadatoknak és a szkeletonnak köszönhetően az alakfelismerést elvégzi, és a háttérdetektálás problémája se a végfelhasználói programra hárul. Ráadásul

mindezt stabil 30 FPS sebesség mellett elvégezni, tehát az AAL alkalmazás többi komponense, a szenzorfúzió, és a következtető rendszer algoritmusai pontos, valósidejű adatokkal tudnak dolgozni, és a rendszer sokkal reszponzívabb. A végtagok mozgását sziluett adatok alapján próbálták meghatározni. Ezen a téren is nagy előnyt nyújt a Kinect, hiszen a teljes szkeletont, és annak egyes részeit külön is képes követni.

Activity Analysis, Summarization, and Visualization for Indoor Human Activity [21,22]

Ebben a tanulmányban kameraképek alapján készített emberi sziluettek kinyerésének algoritmusai kerülnek bemutatásra, amiket később emberek követésére lehet használni. Az érdekessége a munkának, hogy a sziluett megahatározása más, dinamikusan mozgó objektumok mellett igen nehéz, de az általuk kidolgozott fuzzy logikán alapuló rendszer képes volt az emberi alakokat szeparálni. Ugyanakkor a végtagok azonosítása nem megoldott probléma. A Kinect erőssége ezzel a rendszerrel szemben is az, hogy a végtagokat könnyebb azonosítani, és egy nagyobb felbontású, pontosabban követett emberalakot kapunk.

Az események (aktivitások, ADL) azonosításához hierarchikus döntési fát használtak. A legfontosabb adatok közé a sziluett helye és sebessége tartozik, és az adott aktivitás (ADL) felismeréséhez még a kontextust is felhasználták. Konkrét pózok betanításával sikerült összetettebb ADL-ek meghatározása is. A nehézséget az okozza, hogy nagyszámú cselekvés nem diszkriminálható csak a sziluett alapján. Ezért több egy többszintű modellt építettek:

1. szint: pozíció sebesség a sziluett alapján
2. szint: aktivitási szint (végtagok mozgása, mozgolódás) meghatározása
3. szint: tényleges cselekvések meghatározása a 2. szintet használva szűrőként, és a pózok segítségével

Ez a háromszintű modell lényegében a heurisztikák használatát jelenti, hiszen a konkrét aktivitás meghatározásához (ADL, 3. szint) az 1. szintet (pl. „konyhában van, pultnál áll”) és a 2. szintet (pl. „mennyire aktív a konyhában?”) használja. Így például, ha egy egyén a konyhában van, és sokat mozognak a kezei a pultnál, akkor valószínűleg éppen főz valamit.

A többszintű viselkedés és kontextus modellezés egy olyan megoldás, amit egy Kinect AAL rendszerben is célszerű lenne alkalmazni.

Evaluation of an Inexpensive Depth Camera [23]

Ez a munka kiváltképpen érdekes, hiszen összehasonlít három, különböző fajta, elesés detektáló rendszert, köztük egy olyat is, ami Kinect szenzort használ az emberek

felismeréséhez. Mindegyik rendszer képes az emberek felismerésére és pozíciójának követésére. Az elemzés konklúziója, hogy Kinect szenzort használó megoldás is pontos, sőt számos előnnyel is rendelkezik a többi rendszerhez képest: invariáns az ambiens fényre és egyszerűbb algoritmusokat kell használni a mélységszenzor és a szkeleton adatok miatt. Ugyanakkor a hátránya, hogy szűk a látószöge, bizonyos ruhatípusok, amik más módon verték vissza az IR (infravörös) fényt megzavarták a szenzort, és a falhoz, vagy földhöz közeli ember alakok sokszor beolvadnak a felületbe. Sajnos a Kinect pontossága a távolsággal romlik (1 méteren 2cm, 10 méteren már 10cm), amit munkám során is számításba kell majd venni.

További rendszerek [24,25,26]

Olyan rendszereket is megvizsgáltam, melyek több kamerát használnak, és a két kamerakép alapján képesek az emberi alak megkeresésére és követésére a 3D térben. A [22] megoldásához hasonlóan itt is fuzzy logikát használtak a különböző aktivitások megahatározásához, elsősorban az elesés detektálásra koncentrálva. Az audió és videó érzékelők kombinált használatával foglalkozik a [25] munkában bemutatott rendszer. A kamera képeken olyan heurisztikákat használ, melyek segítségével képes szeparálni a fejet, felsőtestet és lábakat, az audió adatokból pedig a sírást, köhögést és kiabálást szűrik ki. Ez azért megfontolandó, mivel a Kinect is rendelkezik mikrofonnal, így adott lenne a két szenzortípus együttes használata. A [26] munkába érdekessége, hogy a Kinect használatával oldják meg az elesés detektálás problémáját, különösen ügyelve az olyan esetekre, amikor a követett ember félig, vagy teljesen kitakart helyzetbe kerül az esés következtében (bútor mögé esik). A megoldás működőképesnek tűnik, az eleséseket nagy biztonsággal detektálja, de jelenleg nem valós időben, hanem csak előre rögzített adatok alapján működik.

További elesés detektáló és célzottan gesztusfelismeréssel foglalkozó rendszereket is megvizsgáltam, melyek módszereit a tervezési fázisban, és a tesztelésénél fogom szem előtt tartani.

3.2.1 Összefoglalás

A megismert AAL megoldások közül a legtöbb egy vagy több kamerakép alapján próbálja meg az emberi alakot felismerni, azokról statisztikákat készíteni, majd különböző következtetők használatával felismerni az ADL aktivitásokat, azok közül is elsősorban a pozíciót és sebességet, illetve az elesést. Magasabb szintű következtetéseket további

heurisztikák használatával és különböző szenzorok fúziójával vonnak le. Kijelenthetjük, hogy a meglévő rendszerekben a helymeghatározás, és az elesés detektálás a két megoldott AAL probléma. A gesztus és ADL aktivitás felismerésre koncentráló munkák [27,28,20] alapján az általános, de részletesebb gesztusok felismerése még viszonylag kezdetleges, és összetettebb aktivitások és pózok felismerését további szenzorok bevonásával oldják meg.

3.3 Kinect AAL rendszer tervezése

A tervezési fázis első lépéseként a fejezet további részében a Kinect AAL rendszer tervezésekor felmerülő alapvető kérdések által felvetett problémákat elemzem:

- Milyen előnyünk származik a Kinect használatából?
- Mit és milyen szinten szeretnénk modellezni? (Kontextus, sebesség és pozíció, mozgásmennyiség, globális statisztika eseményekről, konkrét mozdulatok?)
- Milyen feladatokat akarunk megoldani, ADL aktivitásokat felismerni?

3.3.1 Kinect használatának előnyei AAL alkalmazásokban

Korábbi fejezetekben már bemutattam, hogy a Kinect legnagyobb előnye a harmadik dimenzióról kapott szenzorikus információ az alap kétdimenziós kép mellett. Erre épülve, a szoftver keretrendszerek számos olyan eljárást biztosítanak, amiben a szegmentálás, emberi alak keresése és követése már implementálva van, ráadásul ennek egy része már a szenzorban (szenzor chip) van megvalósítva, így a feldolgozási sebesség is jobb a kamerás rendszerekhez képest. Ez lényegében azt jelenti, hogy itt nem szükséges az emberi alak megtalálásával, és az ehhez kapcsolódó feladatokkal foglalkozni, hanem a magasabb szintű (elesés, gesztus, ADL aktivitás) felismeréssel és modellezéssel kapcsolatos feladatokra lehet fókuszálni.

A Kinect elsősorban a képi információ alapján dolgozik, tehát a kamerákat hivatott kiváltatni, így más szenzorok (ajtónyitás figyelő, hőmérséklet szenzor, nyomásérzékelő… stb.) használata továbbra is fontos tényezője az AAL rendszernek. A szenzorok fúziója, a közös információ felhasználása nem lesz része a rendszer első verziójának, de a dolgozat végén megoldási javaslatokat adok a Kinect felhasználására szenzorfúziós rendszerekben.

A szoftver keretrendszer könyvtárai olyan információt is szolgáltatnak (végtagok pontos azonosítása és szkeleton követése), ami más megoldásokban ilyen szinten még nem jelent, így a rendszer egyik fő feladat a többletinformáció felhasználás lehetőségeinek vizsgálat

AAL alkalmazásokban. Várhatóan az emberkövetés (amiről az 3. fejezet elején bemutatott megoldások ismeretében kiderül, hogy a mai technika szintjén nehéz képfeldolgozási probléma) terén nyert többletinformációra építve (pozíció, végtagok), mely eddig csak összetett markeres, vagy többkamerás rendszerekkel volt elérhető, olyan rendszert lehet építeni, mely sokkal precízebben és gyorsabban képes alapvető ADL aktivitásokat észlelni, és AAL feladatokat megoldani.

3.3.2 Modellezési szintek

A cél egy összetettebb felügyeleti rendszer megalkotása, mely képes egy szobát modellezni (kontextus), és a benne élő, mozgó embert követni. Mivel a Kinect szenzorral egy egész szobát (későbbiekben, több szenzor használatával akár egy egész lakást) akarunk feltérképezni, és a benne mozgó embert követni, ezért a kontextus és modell meghatározása az egyik alapja a rendszernek. Modellezés alatt azt értjük, hogy a valós fizikai világ paramétereit, működését, és folyamatait hogyan interpretáljuk az AAL rendszer (szoftver) számára. A modellezéssel tulajdonképpen egy leképzést valósítunk meg a két világ (valós, virtuális) között, általában egyszerűsítve a valós világ komplex rendszerét.

A Kinect AAL rendszer, a Kinect szenzor adottságai miatt, az elejétől kezdve kontextus érzékeny alkalmazásnak lett tervezve. A kontextus érzékeny kifejezés az AAL alkalmazások témakörében azt jelenti, hogy a megfigyelt felhasználó aktivitásai (aktivitásainak jelentése) függenek a fizikai, mentális, emocionális állapotától, az ember mozgásától, céljaitól, illetve a fizikai környezettől is. Például egy 20 percig földön fekvő ember a konyhában szokatlan jelenségnek minősül, míg a hálószobában relaxációs technikát végző ember (ismerve a szokásait) nem. Az utóbbi mondat zárójeles része jelenti a kontextus érzékeny alkalmazások kulcsát: plusz információkat ismerve, mint például a fizikai környezet és az ember fizikai pozíciója, számos további következtetést lehet levonni a rendszer számára. AAL alkalmazásokban, sok esetben a fizikai tér entitásai (pl. ablakok, konyhai eszközök) és azok állapotai képezik a kontextust. Adaptív (felhasználóhoz igazodó, rátanuló) rendszerek esetében a viselkedés modell, és az abból kikövetkeztethető mentális, emocionális állapot is a kontextus része lehet. Az első Kinect AAL rendszer esetében csak a fizikai tér és annak entitásai (pl. szekrény, szék, asztal) képezik a kontextust (nem modellezünk más viselkedést), és a következő két elemből fog felépülni:

- Valós fizikai tér virtuális megfelelője.

- Kitüntetett entitások azonosítása, melyekkel kontextus információt lehet bevinni a rendszerbe.

A kontextus segítségével az ADL aktivitások könnyebben azonosíthatók lesznek, estenként bizonyos aktivitások teljesen kizárhatóak. Ezzel lényegében a [22] forrásban bemutatott rendszerhez hasonlóan egy többszintű modellel fog rendelkezni a rendszer:

1. szint: Pozíció adatok, kiegészítve a szoba modelljével, azaz a térrészek és a benne található entitások.
2. szint: Végtagok azonosításából származó adatok, információk.
3. szint: Tényleges ADL aktivitások és gesztusok azonosítása.

A szintek között egymásra épülés érvényesül, tehát az alacsonyabb szint adatai a fölötte levő szint „szűrőjeként" szolgálnak, ezzel szűkítve, módosítva a keresési teret, de bizonyos feladatokhoz nem szükséges az összes szint információinak felhasználása.

3.3.3 A rendszer működése

A program alapját egy jelfuzionáló és következtető rendszer fogja képezni. AAL alkalmazások körében számos példa létezik [22,24,29] különböző következtető rendszerek használatára. Az emberek az AAL alkalmazások szempontjából kiszámíthatatlanok, a mért paraméterek folytonos értéktartománnyal rendelkeznek, tehát statikus, szabály alapú rendszer nehezen alkalmazható. A Kinect igen pontosan képes a pozíció meghatározására, ami alapvetően ad egy heurisztikát a lehetséges cselekvések halmazára, de ettől függetlenül minden szinten tudni kell kezelni a bizonytalanságot. Ha a rendszer nem képes kellő bizonyossággal felismerni egy cselekvést vagy eseményt, akkor képes legyen ezt jelezni, és adja meg a legvalószínűbb cselekvéseket, vagy eseményeket. Ez a megközelítés kombinálható a többszintű modellezéssel, így minden cselekvéshez minden modellezési szinten hozzárendelhető egy „valószínűségi" faktor.

Az AAL rendszereknél általában két fajta módon lehet az eseményeket feldolgozni:

1. statisztikák, események regisztrálása, hosszú távú idősor elemzés
2. azonnali feldolgozás, reaktív rendszer

Az első fajta feldolgozásnál az adatok egy globális tárba kerülnek, ami alapján később kimutatások, statisztikák készíthetők, illetve erre építve egy tanuló, következtető rendszer további információkat nyer ki. A másik a reaktív rendszer, ami egyes események

bekövetkezésekor azonnal reagál. Ez alá tartozik a kritikus események azonnali azonosítása (pl. elesés detektálás), és az explicit vezérlés (gesztusfelismerés és gesztussal való vezérlés).

A Kinect AAL rendszer első verziója csak a második fajta feldolgozási mechanizmust fogja követni, tehát hosszú távú statisztikákat nem fog készíteni, és nem fog tanuló rendszerrel rendelkezni.

3.4 Megvalósítandó feladatok

Ebben a pontban definiálom a konkrét aktivitásokat és események, amiket a rendszer első verziója kezelni fog. Ehhez az előző pontban bemutatott modell vázlatot is felhasználom, de az aktivitások definiálásához először fontos megadni az azokat leíró paramétereket.

3.4.1 Paraméterek

A munkám célja a Kinect felmérése az AAL alkalmazásokhoz, így az elsődleges paramétereket a Kinect szenzorral fogom mérni. A jövőben (a rendszer egy későbbi verziójában) más szenzorokból származó paramétereket is lehet használni, de jelenleg csak a Kinect adataira hagyatkozik a rendszer. A Kinect képességei alapján kerültek meghatározásra az alapfeladatok és ADL aktivitások leírásához, felismeréséhez használható paraméterek:

1. Helymeghatározás, alany pozíciója (tömegközéppont) a térben: X, Y, Z koordináta a kalibrált térben.
2. Nézési irány: 3D vektor a kalibrált térben
3. Alany sebessége (iránya és nagysága): X, Y, Z koordináta a kalibrált térben
4. Statikus objektumok pozíciója a térben (fizikai modell része): X, Y, Z koordináta, középpont és dimenziók megadásával
5. Alany távolsága objektumoktól: mindegyik objektumtól való távolság
6. Felsőtest dőlésszöge (modellezett szoba síkjához képest)
7. Végtag adatok (szkeleton adatok):
   a) Felsőtest és lábak pozíciója egymáshoz képest (pl. combok milyen szöget zárnak be a felsőtesttel)
   b) Felsőtest és karok pozíciója egymáshoz képest (pl.: test mellett, fölött;)
   c) Karok behajlításának mértéke (pl.: kinyújtott, behajlított, teljesen behajlított)

d) Kézfej, könyök, váll, csípő, térd, lábfej csuklópontok relatív pozíciója a tömegközépponthoz képest: X, Y, Z koordináta a kalibrált térben

A felsorolt paraméterek kombinációi fogják képezni az egy eseményhez vagy ADL aktivitáshoz szükséges adatokat. A tervezési fázisban ezeket csak iránymutatónak definiáltam, a végső rendszerbe iteratív módon építem be, a teszteléssel párhuzamosan, ezzel megfigyelve a következtető rendszer hatékonyságának és funkcióinak változását a paraméterek függvényében. A paraméterek végső kialakítása majd a 4. fejezetben olvasható.

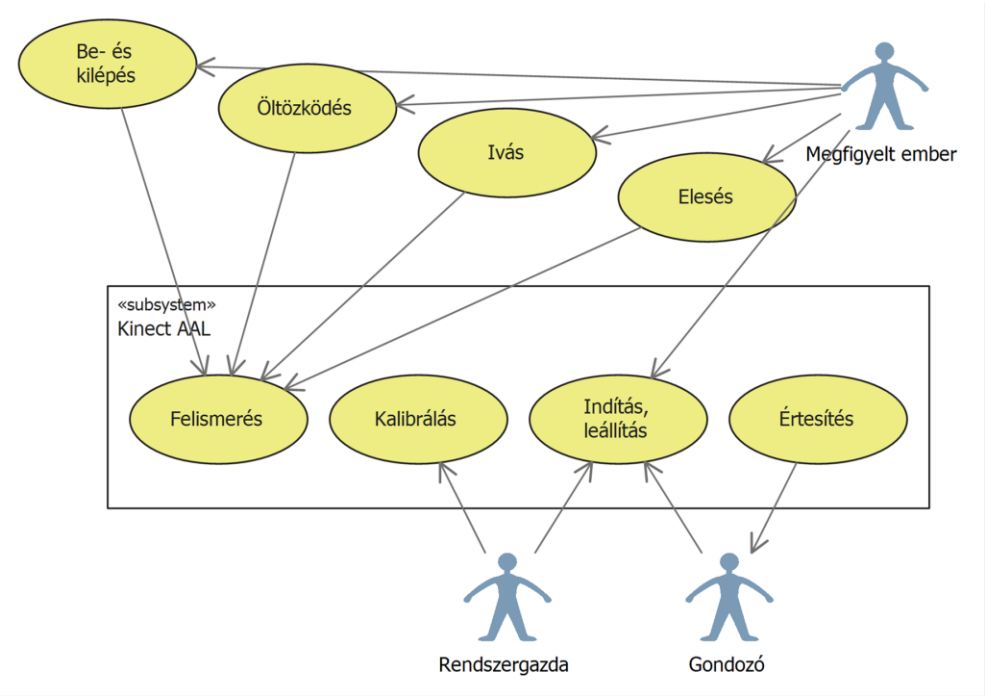
3.4.2 ADL aktivitások és események

Ahogy a célkitűzésnél és a megvalósítandó feladatoknál is leírtam, a dolgozat célja pár ADL aktivitás és esemény felismerése. Továbbá célom kielemezni a Kinect képességeit, és megvizsgálni, hogy milyen aktivitásokat lehet biztosan felismerni a szenzor segítségével, és melyeket nehezen, vagy egyáltalán. A rendszer első verziójában csak alapfeladatok lesznek megvalósítva, de a program már fel lesz készítve az összetettebb feladatok, ADL aktivitások kezelésére is. Más szenzorok adatait nem használja a program első verziója, de a későbbi verziókban külön modulként lehet integrálni őket, ezzel vizsgálva a teljes rendszer pontosságának változását több szenzor együttes használatkor. Az alapvető felismerendő ADL aktivitások, gesztusok és megoldandó feladatok:

1. helymeghatározás (tartós követés és abszolút pozíció meghatározása a térben)
2. sebesség meghatározása
3. be- és kilépés a megfigyelt térrészből
4. elesés detektálás
5. egyedi gesztusok
   a) ivás (vagy gyógyszerbevétel; egyéb egyszerű mozdulat)
   b) öltözködés (vagy egyéb tárgyhasználat; nehezített felismerési körülmények)

Majdnem minden feladathoz szükséges a Kinect kalibrálása, és a megfigyelt térrész fizikai és funkcionális modelljének megadása. A kalibrálás a valós fizikai tér és a modellezett tér megfeleltetését jelenti, tehát a rendszer képes legyen a Kinect által szolgáltatott pozícióadatokat a szoba modelljével együtt kiértékelni. A felismerendő események közül az első kettő, és a szobába való be- és kilépés azt az alapvető funkciót szimbolizálja, hogy a rendszer képes érzékelni az alanyt az adott térrészben, és követni, illetve lekezelni azt az esetet, amikor kilép a kamera látóteréből. Az elesés detektálás olyan alap AAL feladat, amire

különböző szenzorok használatával már számos megoldás készült, de fontos megvizsgálni a Kinect felhasználhatóságát a problémához. Az egyedi gesztusok (és ADL aktivitások) közül az első minden olyan egyszerűbb gesztust szimbolizál, amihez tárgyat egyáltalán, vagy nem zavaró módon kell használni. Ide tartozhat akár a gyógyszerbevétel (ami egy nehezen felismerhető ADL aktivitás), vagy az ivás. A második fajta egyedi gesztuscsoport olyan gesztusokat tartalmaz, amik valamilyen tárgyhasználatot igényelnek. Például öltözködés közben a ruhadarabok mozgása nehezíti a felismerést, így várhatóan ez nehezebben felismerhető mozgás. A rendszer első verziójában az ivás és öltözködés gesztus felismerése a kitűzött cél, de a hosszú távon egy általánosabb gesztusfelismerés beépítése a rendszerbe javasolt.



3.1. ábra Use Case diagram: a rendszerfunkciók és felhasználók

A 3.1 ábrán a tervezett rendszer összesített használati eset diagramja látható az alapvető ADL aktivitásokkal és funkciókkal, három felhasználói szereppel (rendszergazda, gondozó, megfigyelt alany). A rendszergazda képes a rendszer kalibrálására (telepítés, segítségnyújtás a gesztusok betanításában... stb.) és ki-be kapcsolására. A gondozó a rendszer értesítéseit kapja meg, és képes a rendszert ki- és bekapcsolni. A megfigyelt felhasználó saját cselekvéseit tudja elvégezni, amit a rendszer megpróbál felismerni, továbbá (szokatlan módon) a rendszer kikapcsolására is képes. Ennek oka, hogy tesztek alapján kimutatták, hogy

a végfelhasználók sokkal könnyebben fogadtak el egy megfigyelő rendszert, ha tudatában voltak annak, hogy bármikor képesek kikapcsolni azt [2].

3.5 Következtető rendszer

Az AAL rendszer „intelligens” része maga a következtető rendszer, mely a szenzorikus információk alapján, valamilyen algoritmus szerint következtetéseket von le. A kimenete általában események sorozata, amit a rendszer már „beégetett” módon kezel le (pl. felhasználói felületen való kijelzés, vagy automatikus riasztás). A következtető rendszer számos paraméter alapján dolgozik, de véges univerzumban, tehát a paraméterek száma, jellege, dimenziói, és értékkészlete is ismert. A kimenetként generálható események is előre rögzítettek, ezek valójában az előző pontokban definiált felismerendő események és ADL aktivitások. Korábbi fejezetekben is utaltam rá, hogy egy AAL alkalmazásban használt következtető rendszernek tudnia kell kezelni a bizonytalanságot, és képes kell, hogy legyen a bizonytalan kimenteket valamilyen kvantitatív mértékkel jelezni. Erre azért van szükség, mert (főleg a kamerás rendszerek esetében) a nehezen hozzáférhető környezet miatt a mért paraméterek zajosak. Továbbá sok esetben nem akarunk egzakt eredményt kapni a rendszertől, csak egy valószínűségi eloszlást a lehetséges eseményekről. A mérés bizonytalansága és az emberek kiszámíthatatlansága miatt nem célszerű olyan rendszert alkalmazni, ami éles határok szerint végez következtetéseket, ugyanakkor felesleges egy reaktív rendszerhez összetett, betanítandó tanuló algoritmusokat alkalmazni.

A [6] munkában bemutatásra kerül egy olyan rendszer (ZAINGUNE), mely egyszerű, szabály alapú következtetőt használ a szenzoradatok és az események összekapcsolására, illetve egy másik rendszer (PAUL), mely fuzzy logikát alkalmaz. Egy speciálisan kamerakép alapján való elesés detektálással foglalkozó rendszer [24] intelligens következtető rendszere is fuzzy logikán alapszik, és nagy megbízhatósággal képes az elesést detektálni. A fuzzy rendszerek előnye, hogy intuitív módon építhetők fel, kezelik a bizonytalanságot, minden paraméterhez halmazok jelölhetők ki a paraméterértékek fölött, egyéni tagsági függvényekkel, melyek alapján egy paraméterérték akár több halmazba is tartozhat, más-más valószínűséggel. Ezen kívül egyszerű, intuitív módon megadható szabályokkal építhető fel a rendszer, ami a szabályok véges halmaza, és kezelhető mérete ellenére is képes folytonos és differenciált kimenetet adni. További előnyük még, hogy képesek különböző típusú és dimenziójú változók között leképzést megvalósítani, így különböző szenzoradatokat képesek

integrálni. A Kinect AAL alkalmazásban is célszerű lenne a szenzorból érkező különböző adatok integrálására egy fuzzy loginkán alapuló következtető rendszert alkalmazni, mivel a létező megoldások alapján, a kitűzött célfeladatok megoldhatók ezzel.

3.5.1 Fuzzy rendszerek

A fuzzy logikát és fuzzy halmazok elméletét Lotfi A. Zadeh dolgozta ki 1965-ben, és publikálta számos tudományos folyóiratban [30,31]. A fuzzy logika lényege, hogy képes a meghatározatlanságot (bizonytalanságot) is kezelni, és gyakorlatilag egy eszköz annak specifikálására, hogy egy objektum (paraméter, érték) mennyire illeszkedik egy adott leíráshoz. A fuzzy rendszerek lényegében egy nemlineáris leképzést valósítanak meg a bemeneti és kimeneti változók (paraméterek) között, és képesek kezelni a nem számszerűen megadott, hanem címkézett (természetes nyelven megfogalmazott) változókat és a számszerű adatokat is. Rövid összefoglalót és leírást a fuzzy elméletről a Mesterséges Intelligencia Modern Megközelítésben [32] könyv 14.7 fejezetében és a [33] forrásban olvashatunk.

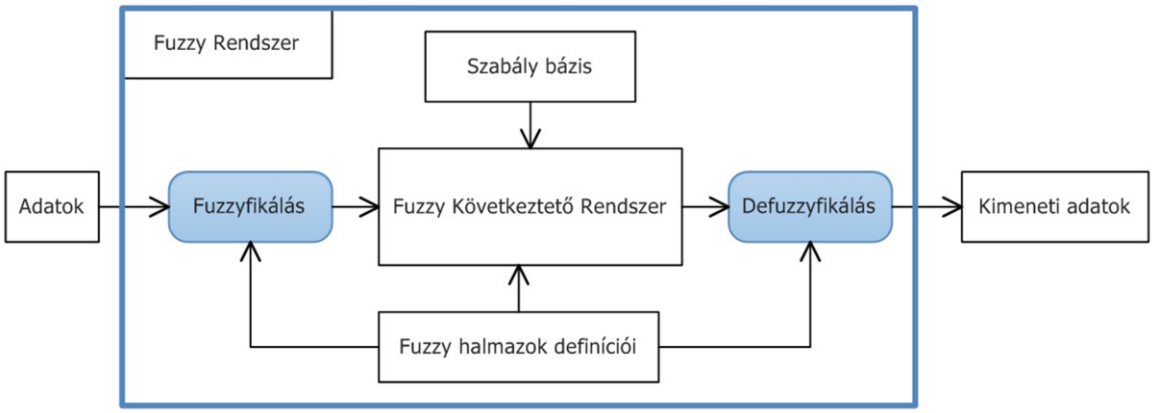
A fuzzy elmélet négy alapfogalma:

1. fuzzy változók (Linguistic Variable)
2. fuzzy halmazok (Fuzzy Set)
3. tagsági függvények (Membership Function)
4. fuzzy szabályok (Fuzzy Rule, Rulebase)

A változók értékkészlete (fuzzy kifejezéssel: Universe of Discourse) általában a valós számok halmaza, és a változókhoz tartoznak a paraméterértékek fölött definiált fuzzy halmazok és tagsági függvények is. A halmazokat, adott értékkészlet fölött, a klasszikus halmazelmélethez hasonlóan függvényekkel kell megadni. Egy adott változó értéke tartozhat több halmazba is, de a klasszikus halmazelmélettel ellentétben itt a halmazba tartozás mértéke a [0,1] intervallumon belül van. Ha x jelöli a paraméterértéket és F függvény a halmazba tartozást, akkor a klasszikus halmazelmélet szerint $F(x) = \{0, 1\}$ értékeket vehet fel (két diszktér érték, melyek jelentése: az adott érték a halmazba tartozik vagy sem). A fuzzy halmazelmélet szerint $F(x) = [0, 1]$ (folytonos érték a zárt intervallumon).

A tagsági függvények segítségével lehet a változó értékkészlete fölött a fuzzy halmazokat definiálni. Egy adott érték felett a tagsági függvények határozzák, hogy az adott érték mekkora mértékben tartozik egy adott halmazba. A halmazokat címkékkel (természetes nyelvi kifejezésekkel, kategóriákkal) jelölik. Függvényeknek általában háromszög, trapéz,

vagy Gauss függvényt használnak, elsősorban az egyszerű tárolásuk és számításuk miatt, de akármilyen tagsági függvény használható. A gyakorlatban egy változót általában 3-7 halmazzal szoktak lefedni, sokkal többel nem érdemes, mert így nehezebben definiálhatók a szabályok.



3.2. ábra Fuzzy rendszerek általános architektúrája

A fuzzy szabályokkal lehet a változókat rendszerbe kötni, tehát gyakorlatilag ezek segítségével lehet a nemlineáris leképzést megvalósítani a bementi és kimeneti változók között. A szabályok természetes nyelvi mondatokhoz hasonlítanak, és két részből állnak:

- premissza (antecedens),
- konklúzió (consequens),

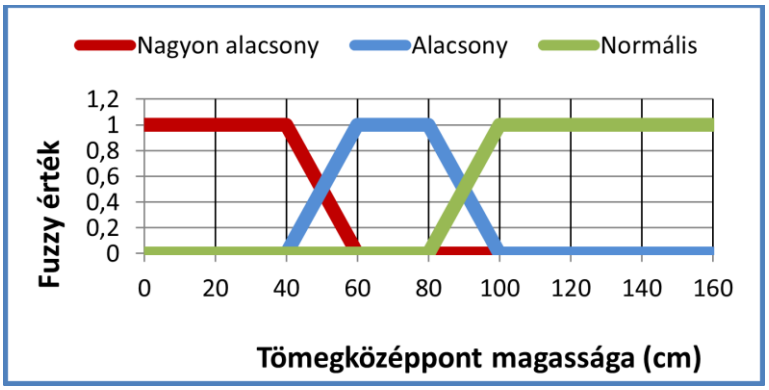
melyeket HA – AKKOR alakban kell megadni, ahol a HA rész a premissza, és az AKKOR rész a konklúzió. A szabályok pontos definiálása a [33] forrásban olvasható, de annyit érdemes róluk tudni, hogy a klasszikus Boole-operátorok (ÉS, VAGY, NEM) fuzzy halmazokra értelmezett változataik (MIN, MAX, 1-X) használhatók bennük, az operandusok pedig a fuzzy változók halmazai.

A fuzzy rendszerek általános struktúrája a 3.2 ábrán látható. A rendszer bemenetére valós mért adatokat kötünk, melyeket a rendszer először a fuzzy következtető által értelmezhető formára alakít, azaz *fuzzifikál*. Ez az jelenti, hogy a bemeneti változó értékékekhez megállapítja a fuzzy halmazokhoz tartozás mértékét (azaz a halmazok tagsági függvényeinek értékét az adott pontban). Ez a folyamat már magán a rendszeren belül történik a változók halmazai, és tagsági függvényei alapján. A fuzzifikált adatokkal és a belső szabály bázis (Rulebase) alapján a rendszer elvégzi a következtetést (inference), majd az eredményeket visszaalakítja a külső rendszer által feldolgozható értékekre, azaz *defuzzifikálja*. Az összes bemeneti változó és szabály kiértékelését a fuzzy következtető

rendszer (Fuzzy Inference System) végzi. A defuzzifikálást többféle módszerrel lehet végezni, melyek közül a leggyakoribb a COG (Center Of Gravity), azaz a súlypont módszer. Ennek lényege, hogy a kimeneti változó értékét a tagsági függvények területének súlypontja alapján határozza meg.

AAL fuzzy példa

A következő szakaszban egy rövid AAL példán mutatom be a fenti fogalmak jelentését. Tekintsük az elesés detektálás problémáját, és egy bemeneti fuzzy változó legyen az ember tömegközéppontjának aktuális magassága (aktuális pozíció Y koordinátája). Az értékkészlete nyilván korlátos, hiszen nem lehet kisebb 0-nál, és átlagosan egy felnőtt ember tömegközéppontja 100-120cm-en van, így felső korlátnak 160cm-et jelölhetünk ki. A fuzzy halmazokat e fölött az értékkészlet fölött kell definiálni adott tagsági függvényekkel. Az egyszerűség kedvéért most a 3.3 ábrán látható módon három trapéz alakú tagsági függvénnyel fedjük le az értékeket (definiáljuk a halmazokat).



3.3. ábra Példa fuzzy változóra és halmazokra

Elképzelhető, hogy a fenti változón kívül még több más paraméter is létezik, melyek dimenziói teljesen eltérhetnek. Definiáljunk egy másik bementi változót is, ami a mozgási sebességet, tehát a tömegközéppont pozíciójának változását reprezentálja, illetve egy kimeneti változót is, ami legyen a vészhelyzet mértékét jelző „vészhelyzet” változó, [0, 10] értékkészlettel. Ez fogja jelezni, hogy mekkora a vészhelyzet mértéke, és halmazai legyenek: „Nincs”, „Közepes”, „Kritikus”. Szükséges a szabályok definiálása is, amik segítségével kapcsolatot teremtünk a bemeneti és kimeneti változók között. Intuitív módon a szabályok legyenek a következők (a teljesség igénye nélkül):

- IF pozíció = Normál AND sebesség = Gyors THEN vészhelyzet = Közepes
- IF pozíció = Alacsony AND sebesség = Gyors THEN vészhelyzet = Kritikus.
- IF pozíció = Normál AND sebesség = Lassú THEN vészhelyzet = Nincs

Ezzel megadtuk, hogy ha normál magasságban van, de gyorsan változik a sebessége (elkezdett zuhanni), akkor közepes legyen a vészhelyzet. Ha már alacsonyan van, és (még mindig) gyorsan zuhan, akkor kritikus a vészhelyzet, hiszen ilyen pozícióban ilyen sebességgel normális esetben nem mozoghat. Az utolsó szabály azt az esetet fedi le, amikor az ember normál helyzetből lassan ereszkedik le, például föl akar venni valamit a földről. Természetesen jóval több szabályt is lehet definiálni, általában (kis tagszámú premissza esetén), érdemes úgynevezett szabály mátrixot képezni a bementi és kimeneti változók között, így minden esetet lefedve és definiálva. Egy ilyen szabálymátrix a fenti példára, két premissza esetén 9 cellából áll, és a 3.1 táblázatban látható.

Pozíció \ Sebesség	Zéró	Lassú	Gyors
Normál	Nincs	Nincs	Közepes
Alacsony	Nincs	Nincs	Kritikus
Nagyon alacsony	Közepes	Közepes	Kritikus

3.1 táblázat Példa fuzzy szabálymátrixra

A Kinect szenzorból érkező adatokat előfeldolgozás után a következtető rendszer bementére kötjük (megadjuk a változók aktuális értékét). A fuzzy rendszer először fuzzifikálja az adatokat, majd elvégzi a következtetést a szabálybázis alapján, majd az eredményül kapott fuzzy értéket defuzzifikálja, és kiadja a rendszer kimenetére. Így az előbb definiált kimeneti „vészhelyzet” változónkra egy konkrét értéket (a „vészhelyzet” változó értékkészletén belül) kapunk.

A fenti példában azt mutattam meg, hogy AAL rendszerünk esetében hogyan lehet a fuzzy logika eszközkészletét használni. Az AAL rendszer eseményeit kvalitatív mércével (címkék, fuzzy halmazok) felbonthatjuk, emellett, ha a fuzzy halmazba tartozáshoz tartozó függvényeket 1-re normáljuk, és biztosítjuk, hogy minden paraméterérték felett a fuzzy halmazokba tartozások összege 1 legyen, akkor valószínűségi értelmezést köthetünk az eseményekhez (fuzzy változókhoz). A fuzzy rendszereket általában robotvezérlésre, visszacsatolt rendszerek vezérlésére használják, de a fent bemutatott módon az AAL rendszereknél is alkalmazható.

4. Az AAL rendszer megvalósítása

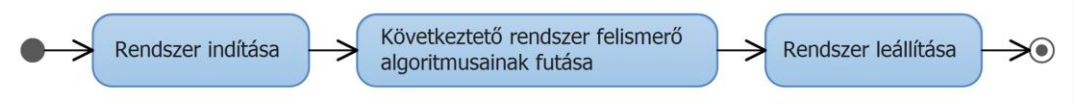
Ebben a fejezetben mutatom be a rendszer előzetes terve alapján az egyes komponensek és részfeladatok fejlesztésének lépéseit, kitérve a fontosabb tervezési megfontolásokra. A rendszer megvalósítása iteratív módon történt, számos tervezési és megvalósítási fázissal, de nem célom az összes fejlesztési lépés részletes leírása, ezért csak a végső megoldást, és az ahhoz szükséges részfeladatok megoldását mutatom be.

4.1 Modellépítés

Az előző fejezetben már ismertettem a modellezés alapkoncepcióját, ebben a szakaszban pedig véglegesítem és pontosan definiálom a teljes modellt. A célkitűzés és tervezés folyamán már többször utaltam rá, hogy a szoftver első verziója nem modellezi a megfigyelt emberek viselkedését. Erre akkor lenne szükség, ha olyan tanuló rendszert építenénk, amivel hosszú távú következtetéseket szeretnénk levonni. Így a feladat kétféle modell definiálására redukálódik: funkcionális modell, mely a rendszer funkcióit és működését írja le, és a fizikai modell, mely a fizikai világot írja le (kontextus).

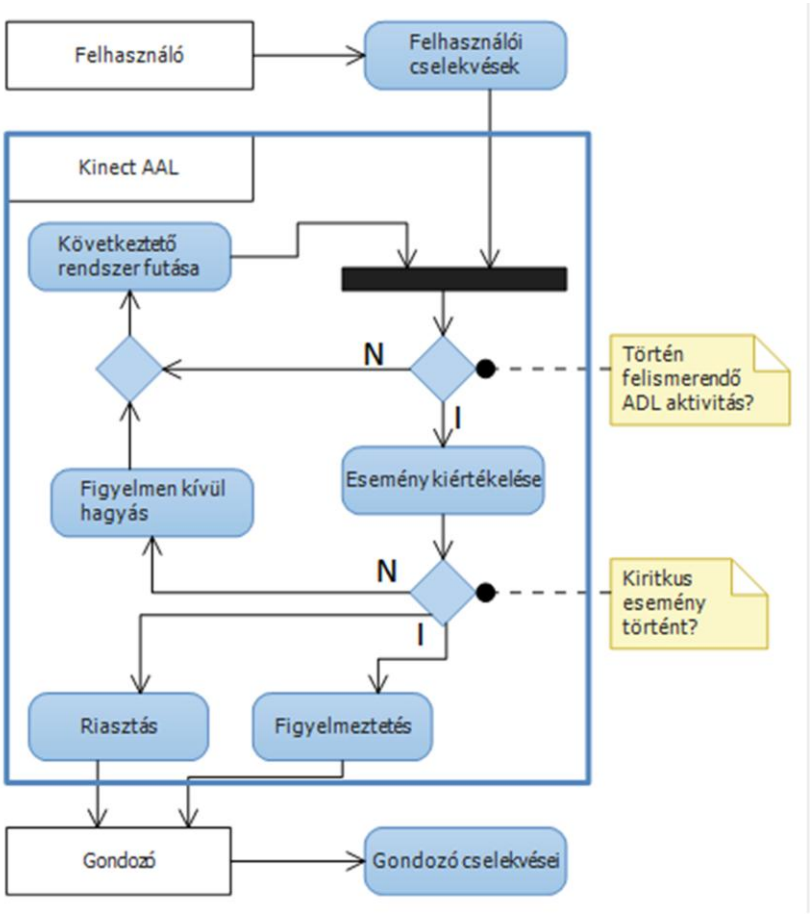
4.1.1 Funkcionális modell

A funkcionális modell írja le a rendszer képességeit, funkcióit, az adatfolyamokat, és a rendszer működését. Alapja a folyamatok (aktivitások, feladatok… stb.) leírása, amely tartalmazza a folyamatban résztvevő külső (való világhoz tartozó) és belső (szoftver rendszerhez tartozó) entitásokat is. Vizualizálására általában adatfolyam diagramokat, aktivitás diagramokat és use case (használati eset) diagramokat használnak.



4.1. ábra Alap funkcionális modell

A 3.1 ábrán már bemutattam a Kinect AAL rendszer használati eset diagramját, mely tartalmazza az alapvető aktivitásokat a rendszer és a külvilág között. A szoftver első verziójában a felhasználó nem képes a rendszer működésébe beavatkozni, hanem mint egy passzív entitás jelenik meg. A 4.1 ábrán az alap funkcionális modell aktivitás diagramja látható. A rendszer indítását követően a Kinect AAL következtető rendszere addig fut, amíg ki nem kapcsolják.



4.2. ábra Kiterjesztett funkcionális modell

Az összetettebb modellt a 4.2 ábra mutatja be. A Kinect AAL szoftver egy jól elhatárolt entitásként szerepel, mely a felhasználóval és gondozóval is kapcsolatban áll. Az alkalmazás következtető rendszere a felhasználó cselekvéseit várja be. Egy aktivitás észlelésekor először ellenőrzi annak érvényességét („Történt felismerendő aktivitás?”), majd ha a kérdésre igenlő a válasz, akkor kiértékeli azt. Az esemény súlyosságától függően vagy nem tesz semmit, vagy figyelmeztetést küld a gondozónak, vagy riasztja a gondozót. Az események figyelmen kívül hagyása nem feltétlenül jelenti azok eldobását, csak ilyenkor az észlelésük nem von maga után olyan akciót, ami a gondozót érinti. A riasztás és figyelmeztetés a Kinect AAL rendszer

skálázhatóságát jelképezi abból a szempontból, hogy mennyire legyen átható a jelenléte a beágyazott környezetben.

4.1.2 Kontextus és fizikai modell reprezentáció

A 3. fejezetben leírt modellterv alapján a végső fizikai modellt az alábbiakban mutatom be. A valós fizikai tér (szoba) minden pontja egy virtuális térkép része. A virtuális térképen a kitüntetett szereppel rendelkező entitások is definiálva vannak, amik lehetnek a tér egyes részei (szektorok), bútorok, vagy egyéb használati tárgyak, továbbá a fizikai modell része a megfigyelt ember aktuális pozíciója is. Összesítve ezek az információk képezik a kontextust. A mozgó ember virtuális térképen való elhelyezéséhez, és az objektumokkal való interakció vizsgálatához szükséges a távolságok kiszámítása az egyes entitásoktól, ezért minden kitüntetett entitás térbeli pozícióját és kiterjedését is be kell vinni a rendszermodellbe. A kis számítási igény miatt a fizikai entitásokat gömbökkel érdemes közelíteni, így két mezővel lehet őket azonosítani:

1. Entitás pontjai (gömbök): X, Y, Z koordináták a 3D térben. Egy entitáshoz több pont is tartozhat.
2. Pontok mérete: a gömbök sugara.

Először furcsának tűnhet ez a megközelítés, de számos praktikus oka van a gömbök használatának. Először is így a távolság számítása kimerül egy euklideszi távolság számításában és egy kivonásban, ugyanis a gömbtől vett távolság egyenlő a gömb középpontjától vett távolsággal csökkentve a gömb sugarával:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} - r \tag{4.1}$$

ahol x_i, y_i, z_i jelölik a két pont három koordinátáját, és r a gömb sugarát.

Ha egzakt módon szeretnénk modellezni az entitásokat, például téglatestekkel, vagy más konvex testekkel, akkor a távolság számítását csak a test minden lapjától vett távolságok megadásával lehetne elvégezni, ami sokkal számításigényesebb. Ugyanakkor a gömbökkel is tökéletesen reprezentálható minden entitás, hiszen a célunk általában az, hogy legközelebbi entitást keressük meg. A gömbök segítségével egyszerűen elvégezhető egy térrészbe való belépés azonosítása, vagy akár egy adott szkeleton csuklópont behatolásának vizsgálata is, hiszen szintén csak két pont távolságának számításával elvégezhető. Megfontolandó, hogy

így nem veszítünk-e a rendszer leíróképességéből? Ez attól függ, hogy mennyire akarunk precízen azonosítani bizonyos eseményeket. A tapasztalat azt mutatta, hogy ezt a közelítő módszert használva is érdemben felhasználható a kontextus információ az ADL aktivitások meghatározásához. A legtöbb esetben arra vagyunk kíváncsiak, hogy mely objektumhoz vagyunk legközelebb, és mekkora ez a távolság. Ez a következő eljárással számolható:

$$d = \min_{\forall i,j} \left(\sqrt{\left(x_{ij} - x_p\right)^2 + \left(y_{ij} - y_p\right)^2 + \left(z_{ij} - z_p\right)^2} - r_j \right) \tag{4.2}$$

ahol x_{ij}, y_{ij}, z_{ij} jelölik az i. objektum j. gömbjének középpontjának koordinátáit, r_i az i. gömb sugarát, és x_p, y_p, z_p a másik kitüntetett pont koordinátáit, ami esetenként eltérő lehet (pl. ember tömegközéppontja). Így minden objektum minden gömbjére kiszámolva a távolságot, majd ezek közül a legkisebbet kiválasztva (min) kapjuk meg a legközelebbi objektumot, és annak távolságát (d).

A fizikai modell felépítését a Kinect AAL szoftveren kívül kell elvégezni, de a szoftver segítségével. A modellt XML formátumban kell megadni a következő elemeket felhasználva:

```
<!DOCTYPE SCENEMODEL [
<!ELEMENT SCENEMODEL (SCENEOBJECTS+, SCENESECTORS +)>
<!ELEMENT SCENEOBJECTS (OBJECT+)>
<!ELEMENT SCENESECTORS (OBJECT+)>
<!ELEMENT OBJECT (POINTS+, ACTIONS+>
<!ELEMENT POINTS (POINT+)>
<!ELEMENT ACTIONS (ACTION+)>
<!ATTLIST OBJECT NAME ID #REQUIRED>
<!ATTLIST POINT X CDATA #REQUIRED >
<!ATTLIST POINT Y CDATA #REQUIRED >
<!ATTLIST POINT Z CDATA #REQUIRED >
<!ATTLIST POINT SIZE CDATA #REQUIRED >
<!ATTLIST ACTION NAME CDATA #REQUIRED >
]>
```

XML DTD séma a modell definiálására

Tehát a SceneModel SceneObjects és SceneSectors elemekből épül fel. Az objektumok és partíciók is Object típusú elemeket tartalmaznak, melyek attribútumként tartalmazzák az objektum nevét (ez azonosítja), és a gömbök halmazát (Points). A Points lista Point

elemekből épül fel, amik attribútumai az x, y, z koordináta és a méret. Az Actions lista tartalmazza azokat az ADL aktivitásokat, melyeket egy adott objektumhoz szeretnénk rendelni. A gömbök középpontjának koordinátáit (x, y, z), és a gömb sugarát is a valós kalibrált fizikai tér koordinátarendszerében kell megadni, milliméterben. Ezzel tehát interpretáltuk a rendszer számára a fizikai modellt, és lehetővé tettük a rendszert számára a kontextus függő működést.

4.2 Kinect kalibrálása

A Kinect bemutatásánál ismertettem, hogy alapvetően két féle adatot szolgáltat a szenzor (és a keretrendszer): színes RGB képet és egy mélységkép (mélység adatok, pixelenként). A szenzor minden mélységadatot saját, Descartes-koordinátarendszerében ad meg, valós mértékegységben (mm). Ez viszont függ az elhelyezéstől (x,y,z), és a három Descartes-koordináta tengely mentén való elforgatástól, azaz a szenzor az aktuális nézési irányához képest relatív koordinátákat ad meg. A Kinect szenzor koordinátarendszere:

- Kinect szenzor az origó (x=0, y=0, z=0)
- Jobbsodrású rendszer, tehát a Kinect szenzor szemszögéből a koordinátatengelyek: x „balra”, y „fel”, z „előre”

Egyes megoldandó feladatokhoz (pl. valós tér modellezése, elesés detektálás) a valós fizikai térhez képest abszolút koordinátarendszert célszerű használni, aminek több, praktikus oka is:

- A Kinect (akár véletlen) átmozgatása után a szenzor koordinátarendszerében rögzített adatok használhatatlanná válnának, hiszen azok a Kinect egy adott pozíciójától függően lettek rögzítve. Gyakorlatilag a virtuális (modell) tér koordinátarendszerének origóját és elforgatását módosítanánk a szenzor elmozgatásával, így viszont a valós fizikai térrel nem lenne összhangban a modell, és a modell alapján rögzített adatok.
- Több szenzor használata esetén a szenzorfúzió egyszerűbben megvalósítható, ha mindegyik szenzor az adatokat egy azonos (abszolút) koordinátarendszerben adja meg.
- A valós tér modelljének felépítése és használata sokkal egyszerűbb egy olyan koordinátarendszerben, ami a valós térrel összhangban van.

Célom tehát a virtuális tér koordinátarendszerének beállítása úgy, hogy az origója a valós tér egy kitüntetet pontja (például a szoba egyik sarka) legyen, tengelyei pedig rendre legyenek a szoba sarkából kiinduló falak által meghatározott síkok metszéseivel párhuzamosak (téglatest alakú szobát feltételezve). Természetesen nem szükségszerű ilyen koordinátarendszert használni, de a lényeg, hogy a koordinátarendszer legyen abszolút, ezáltal független a Kinect szenzor pozíciójától és elforgatásától.

A Kinect kamera dőlésszöge („pitch") a szenzor beépített motorjával +-27^{o} között változtatható. Ezen funkció használata esetében minden bekalibrált entitást és magát a virtuális koordinátarendszert is transzformálni kellene a dőlésszög függvényében. A beépített motor által változtatott dőlésszög igen pontatlan, ráadásul a kamera a dőlésszögét az aktuális helyzetéhez viszonyítva állítja (saját beépített gyorsulásmérő alapján), így ha nem vízszintes felületen áll a szenzor, akkor a dőlésszögek a vízszinteshez viszonyítva állítódnak be, nem az aktuális szenzorpozícióhoz képest. A leírt okok miatt a Kinect AAL rendszer a beépített motort nem használja, így a Kinect szenzor fix pozícióban, fix látószöggel fog rendelkezni.

A megoldandó feladatok szempontjából a következőket kell megfontolni:

- Gesztusok (pl. ivás) felismerését célszerű az egyén lokális koordinátarendszerében vizsgálni, aminek az origója lehet a törzs, vagy csípő pont.
- Elesés detektáláshoz, pozíció meghatározáshoz, térrészbe való behatolás felismeréshez abszolút koordináta rendszert célszerű használni.

A gesztusok felismerése nyilván független kell, hogy legyen az alany aktuális pozíciójától és nézési irányától. Ebből kifolyólag a szkeleton csuklópontok pozícióit (ami a gesztusok alapját képezik) az egyén lokális koordinátarendszerében kell kifejezni. Ez egyszerűen elvégezhető, ha a lokális koordinátarendszer origójának az alany egy kitüntette pontját (pl. a tömegközéppontját) vesszük, és a többi csuklópont pozícióját ezzel az adott ponttal eltoljuk (transzláció). Ettől viszont csak eltolás független lesz a gesztus, de nem lesz elforgatás független. Fontos végiggondolni, hogy mely tengelyek mentén szeretnénk biztosítani az elforgatás függetlenséget. Képzeljünk el egy szituációt, amikor az alany előrefele nyúl a kezével. Ez nyilván különbözik attól a mozdulattól, amikor előre hajolva nyúl a föld felé. Habár a felsőtestéhez viszonyítva mindkét esetben ugyan azt a mozgást végzi a kéz, mégis meg szeretnénk tudni különböztetni a két mozdulatsort. Ugyanakkor, ha az alany csak a valós fizikai tér függőleges (y) tengelye mentén fordul el, akkor az előrefele nyúlást ugyan annak a mozdulatsornak (gesztusnak) kell felismerni. Ebből az következik, hogy a

gesztusokat csak az abszolút (modell) koordinátarendszer *y* tengelye mentén kell elforgatni. A gesztusok eltolás és elforgatás függetlenítését a későbbi a „Gesztusfelismerés” című pontban ismertetem részletesen.

A többi adathoz az abszolút koordinátarendszer használata szükséges, amely feladat átfogalmazható arra a problémakörre, ahol adott két koordinátarendszer, mindkét rendszerben ismert, egymásnak megfeleltethető pontjaink vannak (később ezekre kalibrációs pontokként hivatkozom), és keressük azt a transzformációt, amely átviszi az egyik koordinátarendszert a másikba. Amennyiben csak a szoba síkjának meghatározására lenne szükség, úgy használható lenne a RANSAC sík illesztő algoritmus, vagy Hough transzformáció, ahogyan azt a [26] munkában használták. A Kinect AAL rendszerben viszont nem csak az alapsík egyenletére lesz szükségünk, hanem tetszőleges pont áttranszformálására is, ezért ezek a megoldások nem alkalmazhatók. Számos munkában írnak a pontpárok áttranszformálásának problémájáról, melyre „Absolute Orientation” vagy „Rigid Body Transformation” néven hivatkoznak. Egy Kinect-et használó alkalmazásban, mely a RoboCup (nemzetközi robot foci verseny) elemzéséhez készült, hasonló problémafelvetés olvasható [34]. A kiadványban SVD (Singular value decomposition) módszert használnak a két koordinátarendszer közötti transzformáció kiszámolásához. A [35] cikkben négy különböző analitikus megoldást elemez a szerző robusztusság, stabilitás, és sebesség szempontjából, ideális és zajos adatokon egyaránt. A konklúzió, hogy valós helyzetekben (tehát kis zajjal terhelt adatokon) pontosságát és stabilitását tekintve semelyik algoritmus sem tüntethető ki a többi közül. Az AAL alkalmazás esetében a kalibrációs pontok száma várhatóan kicsi lesz (~10-20), és a kalibráció elvégzése egy adott helyszínhez csak egyszer szükséges, tehát a [35] –ban elemzett algoritmusok futási időbeli különbségei elhanyagolhatóak. A megfelelő algoritmus kiválasztását így arra a praktikus szempontra korlátoztam, hogy mely algoritmus implementálható a legkönnyebben. A [35]-ban elemzett Horn’s Unit Quaternion Method [36] implementációja C++ és Java nyelven megtalálható Ziv Yaniv weblapján [37] (mely egészségügyi számítógépes vizualizációval és AAL alkalmazásokkal foglalkozik). Ez jó alapot adott az algoritmus, és a kalibrációs eljárás teszteléséhez is.

4.2.1 Kalibráló algoritmus

A probléma megfogalmazása általánosan az [35] és [36] alapján: az algoritmus bemenete a két ismert koordinátarendszerben rögzített, egymásnak megfeleltethető pontok (pontpárok) halmaza $\{m_i\}$ és $\{d_i\}$, ahol a megfeleltetés a következő egyenlettel írható le:

$$d_i = \mathrm{R}m_i + \mathrm{T} + \mathrm{V}_i \tag{4.3}$$

ahol **R** egy 3x3-as rotációs mátrix, **T** egy háromdimenziós transzlációs mátrix, és V_i a zajvektor. Keressük tehát az optimális $[\hat{R}, \hat{T}]$ mátrixokat, melyek az $\{m_i\}$ pontokat a $\{d_i\}$ pontokra képzik le, és minimalizálják a négyzetes hibát:

$$e = \sum_{i=1}^{N} \left\| d_i - \hat{\mathbf{R}} m_i - \hat{\mathbf{T}} \right\|^2 \tag{4.4}$$

Horn „Unit Quaternion” megoldása [36] kvaterniókat használ a klasszikus 3x3-as rotációs mátrix helyett. Az eljárás először a rotációt (R), majd a transzlációt (T) határozza meg. Az algoritmus egyik fontos kritériuma, hogy a felhasznált kalibrációs ponthalmaz nem lehet egy egyenesen, és legalább 3 pontpárból kell állnia, a szabadsági fokok száma miatt, hiszen a transzlációnak és elforgatásnak is 3-3 szabadságfoka van, és még egy a nagyítás (scale) miatt. 3-3 ismert pontpár által már 9 megkötésünk van, ami elég a transzformáció meghatározásához.

4.2.2 Kalibráló algoritmus implementációja és tesztelése

A fent leírt algoritmust a [37] forrás Absolute Orientation bekezdésének Java kódja alapján implementáltam, a szerző engedélyével felhasználva bizonyos kódrészeket. Mivel az eljárás mátrixok sajátértékeinek és sajátvektorainak számolását is igényli, ezért egy nyílt forráskódú C# matematikai csomagot (cMatrixLib) használtam ehhez.

Az implementáció helyességét egy ponthalmazzal és egy ismert transzformációval teszteltem: az egyik koordinátarendszerben megadott pontokat átvittem a másik rendszerbe az ismert transzformációval. Az így kapott pontpárokra lefuttatva a kalibráló algoritmust az ismert transzformációt kaptam vissza. Ezzel tehát azt ellenőriztem, hogy ismert összefüggés esetén a pontpárok alapján valóban képes az algoritmus megtalálni a kérdéses transzformációt.

Robusztusságát valós környezetben is tesztelni kell, ami az előbbi teszthez képest annyiban különbözik, hogy a pontpárok kijelölése valós adatok alapján történik, így a mért értékek (pontok koordinátái) zajosak (V_i tag a4.3 egyenletben) lehetnek. Továbbá, mivel a transzformáció nem ismert, nem lehetséges a transzformáció helyességének egyszerű, megfeleltetés alapú visszaellenőrzése, hanem bizonyos ellenőrző pontok alapján kell a kalibrációs hibát meghatározni. Kérdéses, hogy egy hibaérték számolásával mennyire kapunk releváns információt a kalibráció sikerességéről. Az egész kalibrációval a célunk egy olyan transzformáció meghatározása, amely a Kinect-tel mért adatokat a valós koordinátarendszerbe tudja átvinni, és az elesés detektálás szempontjából fontos szerepe van a föld (szoba) síkjának (aminek az abszolút koordinátarendszer Y tengely a normálisa) helyes meghatározásának. Ezek alapján a következő teszteket érdemes elvégezni:

1. Kalibráció hibájának számolása ismert pontpárokra.
2. Szoba síkjának ellenőrzése kijelölt pontok alapján.

Az első fajta teszt egy hibaértéket fog számolni, mely megadja N darab pontra az átlagos hibát, tehát azt a valós mértékegységgel (mm) kifejezhető értéket, mely a kimért pontok, és a transzformációval kapott pontok távolságával számítható. Egy ideális transzformációra ez a hiba 0mm. A második fajta teszt az előzőhöz hasonlóan egy hibaértéket ad eredményül, de speciálisan a szoba síkjában lévő pontok kijelölésével. Ezzel könnyebben kiértékelhető a transzformáció helyessége, hiszen a rögzített szoba síkjában lévő pontok Y koordinátái mind nullák kellenek, hogy legyenek.

A teszteléshez egy külön eszközt használtam, mely a Kinect AAL szoftver része, és lehetőséget ad arra, hogy a Kinect szenzort kalibrálhassuk egy adott helyszínhez, és a kalibrációs transzformációt elmentsük az adattárba későbbi használatra. Az eszköz segítségével a valós környezeten vissza is lehet ellenőrizni a kalibrálást (transzformáció pontosságát) tetszőleges pontra, továbbá, a kalibráláskor megadott pontpárokra a transzformációt lefuttatva automatikus is kiszámolható a kalibráció hibája.

A fentiek alapján, a tesztelés menete a következő:

1. Valós fizikai tér kimérése, és kalibráló pontok meghatározása.
2. Kalibráló pontok Kinect koordinátarendszerbeli pontjainak kijelölése a szoftverrel, és a valós adatok megadása.
3. Kalibráló algoritmus futtatása, és a transzformációs meghatározása.
4. Automatikus visszaellenőrzés az N darab kalibráló pontra.

5. Manuális (kézi) tesztelés: tesztpontok kijelölése a valós fizikai térben, és a Kinect koordinátarendszerben, majd a tesztpontok visszaellenőrzése a transzformációval (a szoba síkjában lévő pontok Y koordinátáinak ellenőrzése).

A kalibráló eszköz segítségével tehát a kalibráló pontokra le kell futtatni az automatikus teszteket (4. lépés), majd további manuális teszteket (5. lépés) is kell végezni. Az automatikus kalibráció hibáját a tényleges és a transzformált pontok átlagos hibája alapján határoztam meg:

$$\hat{e} = \frac{\sum_{i=1}^{N} d_i}{N} \tag{4.5}$$

ahol N a tesztpontok száma, d_i két pont euklideszi távolsága, $\hat{e}$ pedig az átlag hiba. A szoba síkjában lévő pontok esetében a hibát csak az Y koordinátára számoltam, tetszőleges pontokat felvéve a szoba síkjában. Fontos megjegyezni, hogy a Kinect pontossága a távolsággal csökken, és mélységkép azonos pixelhez tartozó értékei is zajjal terheltek, időben nem állandóak, ezért először a zaj kimérése szükséges.

Zaj kimérése

A zajt 5 különböző távolságban lévő pont, 20 értékéből átlagolással határoztam meg. Minden pontot 20-szor mintavételeztem, majd minden pontra meghatároztam az átlagtól való eltérések átlagát és maximumát. A kísérletből az derült ki, hogy a mérési zaj a távolsággal számottevően nem változik, és az átlag zaj elhanyagolható (0.01 mm) nagyságrendű. Hosszútávon tehát a zaj minimális, ugyanakkor egy pont 20 mintavételében a maximális eltérés estenként 30mm-t is meghaladta, amit a kalibrációs pontok kiválasztásánál figyelembe kell venni.

Kalibráció tesztelése és értékelése

A tesztelést a 2. fejezetben bemutatott két teszthelyszínen végeztem el, független kalibrációkat végezve. Minden kalibrációhoz N=8 kalibráló (referencia) pontot használtam, majd további 8, manuálisan megadott pont alapján vizsgáltam a szobasík meghatározásának pontosságát. A mérések eredményei a 4.1 táblázat tartalmazza. Látszódik, hogy a tesztpontokra az átlag hiba 70-110mm között mozgott, míg a szoba síkját körülbelül 40 mm-es pontossággal lehetett meghatározni a kalibráló algoritmus segítségével. A szoba síkjánál egy fontos tényező, hogy a sík mennyire torzul a tényleges föld síkjához képest, azaz mekkora szöget zár be vele. A fenti átlaghibák meghatározását a látótér teljes síkjában

egyenletesen elszórt pontok alapján határoztam meg, és a maximális eltérések 70-80mm között mozogtak. Ez gyakorlatban azt jelenti, hogy a tesztszobák két vége közötti, körülbelül 4m-es távolság megtétele közben az alany tömegközéppontjának föld síkjától vett pozíciója legfeljebb 80mm-t ingadozik.

Teszt helye és sorszáma	Kalibráló pontok átlaghibája	Kalibráló pontok maximális hibája	Szoba síkjának átlaghibája	Szoba síkjának maximális hibája
Teszt szoba 1 - 1	107,158	218.324	38.594	62.354
Teszt szoba 1 - 2	101.402	202.133	41.402	72.652
Teszt szoba 2 - 1	85.072	132.477	37.174	80.577
Teszt szoba 2 - 2	73.242	121.623	39.467	66.479

4.1 táblázat Kalibrációs algoritmus hibája (mm)

Mivel észlelhető volt a szoba síkjának monoton eltérése a valóstól (tehát a hibaértékek monoton nőttek egy adott irányban), egy becslést végeztem a két sík által bezárt szög meghatározására. A fenti maximális 80.577mm-es eltérés egy körülbelül 2m-es sugarú területen volt észlelhető. Ezt alapul véve a két sík által bezárt szög kiszámítható a szinusz tétel segítségével, és eredménye képen meghatározható, hogy a legrosszabb esetben is 0.04^{o} fokos szöget zár be két sík. A kalibrációk után még teszteltem a rendszer működését, és azt tapasztaltam, hogy már három, pontosan meghatározott, egymástól nagyobb távolságra, és minden dimenzióban (x,y,z) eltérő pontból képes volt kis hibával maghatározni a transzformációs mátrixot. További pontok felvételével a zaj miatt sokszor növekedett a hiba, ezért iteratív módon vettem fel a pontokat, minden új pont felvételekor elvégezve a kalibrációt, és figyelve az egyes pontokra számított hibát.

A Kinect AAL által elvégzendő célfeladatokat szem előtt tartva, kijelenthető, hogy a kalibráció hibája az elfogadható tartományon belül van. A kalibrációs algoritmus már kevés pont alapján is képes a transzformációt kellően kis hibával meghatározni, és az elforgatások (a transzformált és a valós koordinátarendszerek között) nem számottevők. Az elesés detektálásnál, és a tömegközéppont meghatározásánál a kalibráció fontos szerephez jut, de a szükséges mérés precizitás 50-100mm-es nagyságrendű, hiszen egy sétáló ember tömegközéppontjának ingadozása is ebben a tartományban van.

4.3 Kinect AAL következtető rendszere

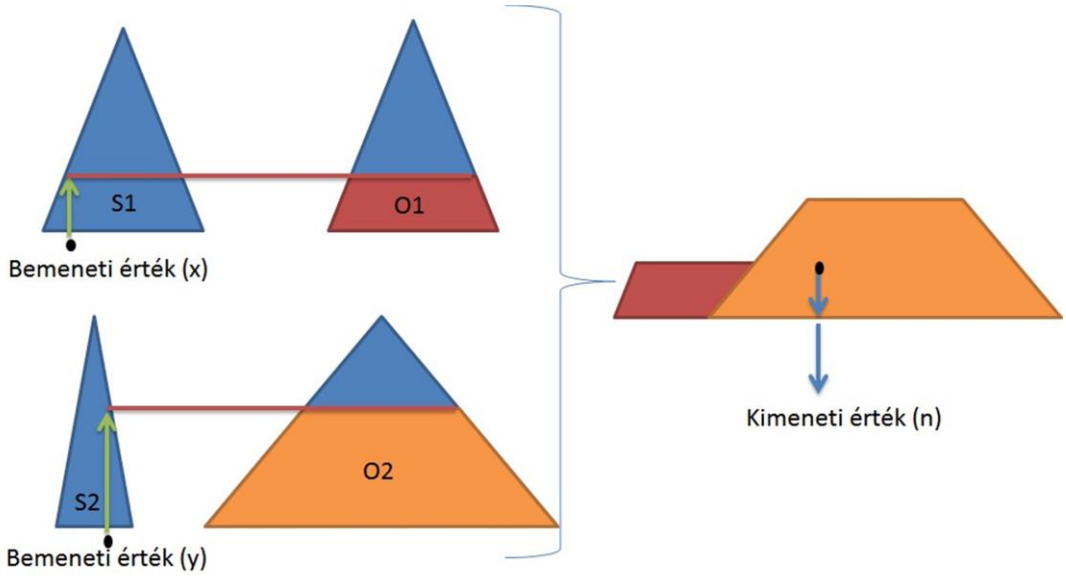
A 3. fejezetben bemutattam a fuzzy következtető rendszerek alapjait és ismertettem a fuzzy következtető használatának előnyeit AAL alkalmazásokban. A [24] munka egy olyan elesés detektáló rendszert mutat be, mely ez RGB kamera adatait feldolgozva és fuzzy következtetőt használva képes a vészhelyzetet jelezni. Ennek elméleti alapjait felhasználva építettem fel a Kinect AAL következtető rendszerét. A 3. fejezetben ismertetett megoldandó problémákra és felhasználandó paraméterekre alapozva alakítottam ki a fuzzy rendszer bemeneteit, és kimeneti változóit. Habár a Kinect AAL alkalmazás egy egységes rendszert képez, mégis két jól elkülönülő problémát old meg:

1. elesés detektálás
2. kontextus függő pozíció és gesztusfelismerés

A következtető rendszer szempontjából a két feladat nem különül el, de a fuzzy szabályok definiálásnál már igen, hiszen az adott problémához tartozó szabályokban csak a problémához tartozó paramétereket kell használni. Fontos definiálni a fuzzy rendszer defuzzifikálásának módját. A defuzzifikálás határozza meg, hogy fuzzy halmazok alapján hogyan számoljuk ki a valós kimeneti értéket. Ez a módszer a következtető rendszerre általánosan jellemző, nagyban meghatározza a kimeneti jel tulajdonságait, tehát kiválasztása átgondolandó. Összesen több mint 20 defuzzifikálási módszer létezik, melyek közül vannak, amik a maximális tagsági értékkel rendelkező értéket adják vissza, vagy az átlagot, de a legelterjedtebb a súlypont alapján működő, úgynevezett Centroid defuzzifikálás, mely a 4.3 ábrán látható. Ennek lényege, hogy a bementi változók értékei alapján (az ábrán *S*) meghatározza a kimeneti változó minden halmazára (az ábrán *O1*, *O2*) a tagsági függvény értékét, majd kettévágja a függvényeket az adott konstans értéknél. Ez például az alábbi két függvény alapján történhet:

- IF x IS S1 THEN n IS O1
- IF y IS S2 THEN n IS O2

Így annyi fél-trapéz területet kapunk, ahány fuzzy halmaza van az adott kimeneti változónak. Végül ezeket a területeket egymás mellé vetítve (az eredeti tagsági függvények alapján) meghatározza az összterület súlypontját. A centroid módszer ezzel a súlyozással biztosítja, hogy a kimenet folytonos, és az összes halmaz felett átlagolt.



4.3. ábra Defuzzifikálás centroid (COG) módszerrel

Esetünkben ez a módszer éppen megfelelő, mert a zajosabb bemeneti adatokat valamilyen szinten átlagolja is, és célunk egy folytonos, durvább ugrásoktól mentes kimenet képzése. Léteznek nagyon hasonló módszerek (Center of Area), és nagyon eltérőek is (Last of Maxima, Mean of Maxima), melyek általában a legnagyobb halmaz érték (az ábrán *O2*) alapján számolják a kimenetet, de ezek élesebb határokkal operáló következtető rendszert valósítanának meg.

4.3.1 Elesés detektálás

Ezt a problémát a [24] munkában leírtakhoz hasonlóan oldottam meg, de más paramétereket használva. Az említett forrásban az ember háromdimenziós rekonstrukcióját (voxel alak létrehozása) követően próbálják meg az elesést felismerni, alapvetően három paramétert használva: az ember (azaz a rekonstruált voxel alak) magasságát, súlypontjának helyét, és a földsík normálvektorának és a voxel alak orientációjának hasonlóságát. A munkában rámutatnak egy fontos tényre, hogy az elesés detektálásnál nem elég a földön fekvés helyzetét felismerni, hanem időbeli (temporális), lefolyásbeli paramétereket is figyelembe kell venni.

A Kinect AAL rendszer több komplexebb paraméter alapján határozza meg az elesést, melyek számításba veszik az idő lefolyását is. A szenzor paraméterek mellett két időtényezőt

is beépítettem a rendszerbe: az elesés óta eltelt idő, és a földön fekvés ideje. Ezeket az időzítőket fix értékhatárok alapján indítom el, vagy állítom le:

- Tömegközéppont magassága alacsony →földön fekvés időzítő indítása
- Tömegközéppont magassága normális → földön fekvés időzítő leállítása
- Elesés lett detektálva a fuzzy rendszer alapján → elesés időzítő indítása
- Tömegközéppont magassága normális → elesés időzítő leállítása

Látszódik, hogy a tömegközéppont aktuális magassága itt is számottevő paraméter. Az időmérőket úgy kell tekinteni a rendszerben, mint származtatott paramétereket, amit a fuzzy következtető megkap. A fekvés időzítője pusztán tömegközéppont aktuális magasságától függ, míg az elesés időzítő az elesés veszélyének detektálásától (fuzzy következtető kimenete). A felhasználandó fuzzy változókat a Kinect szenzorikus adatira alapozva határoztam meg.

Összes fuzzy változó

- Tömegközéppont magassága (koordinátája az Y tengely mentén) [cm]
- Tömegközéppont sebessége [cm/s]
- Földön fekvés időtartama [s]
- Elesés óta eltelt idő [s]
- Elesés kockázata [fix skála]
- Elesés [fix skála]
- Vészhelyzet [fix skála]

Észrevehető, hogy szétválik a konkrét problémát jelző „elesés” változó, és az általánosabb „vészhelyzet” változó. Ezzel célom a következtető rendszer felkészítése más változók bevezetésére, amik a „vészhelyzet” változót befolyásolhatják.

Paraméterek számítása

A pozíció és sebesség paraméterek adatait a Kinect szenzor segítségével mérem. A szoftver keretrendszer képes a követett ember tömegközéppontjának meghatározására a háromdimenziós térben. A már ismertetet kalibráció segítségével, kiszámolhatjuk a tömegközéppont helyét a valós fizikai térben. Mivel a szoftver a pozíciót milliméterben és lebegőpontos számként adja meg, és a mérés pontossága is elég zajos, ezért először egy átlagoló szűrővel simítom pontokat. A simítás mértéke a szűrőt megvalósító puffer hosszával állítható (simító ablak szélessége). A tesztrendszernél 10-es ablakot használtam, tehát körülbelül 1/3-ad másodperc adatait átlagolja futóablak szerűen.

A mozgási sebesség irányának és nagyságának számolása csak kicsivel komplexebb feladat, hiszen a simításon kívül az időt is mérni kell. A sebesség vektor számításához a simított pozíciót használom fel, és minden új frame (adat frame) beérkezésekor az előző és az új simított pozíció alapján először kiszámolom az elmozdulás vektorát, majd elosztom az eltelt idővel, ahogy a 4.6 képlet is mutatja:

$$\vec{v} = \left(\overrightarrow{P_1} - \overrightarrow{P_2}\right)/dt \tag{4.6}$$

ahol $\overrightarrow{P_1}$ és $\overrightarrow{P_2}$ a két simított pozíció vektor, dt pedig a két mérés között eltelt idő.

A végső fuzzy változókhoz a pozíció és a sebesség vektornak az Y koordinátáit használom fel.

Az elesés felismerésének algoritmusa

1. lépés
   a) Tömegközéppont magassága és sebesség alapján meghatározzuk az elesés kockázat aktuális értékét.
   b) Az elesés kockázatának mértékétől függően elindítjuk az elesés óta eltelt időt mérő időzítőt.
   c) A pozíció függvényében elindítjuk a földön fekvést mérő időzítőt.
2. lépés
   a) A földön fekvést és az elesést mérő időzítők alapján meghatározzuk az elesés kimeneti változó értékét.
   b) Az elesés változó mértékétől függően változik a vészhelyzetet jelző kimeneti változó.

Felmerülhet a kérdés, hogy érdemes-e külön mérni a földön fekvés idejét és az elesés detektálása óta eltelt időt. A válasz: igen, ugyanis, habár a legtöbb esetben, amikor elesés veszélye áll fent, akkor mindkét időmérő aktiválódik (hiszen földközelbe kerülünk), mégis meg tudjuk különböztetni (és ez által rögzíteni) azt az eseményt is, amikor az alany csak lefekszik a földre. Ezzel tovább lehet finomítani a vészhelyzet jelzését, ugyanis adott környezetben ez is jelenthet abnormális viselkedést.

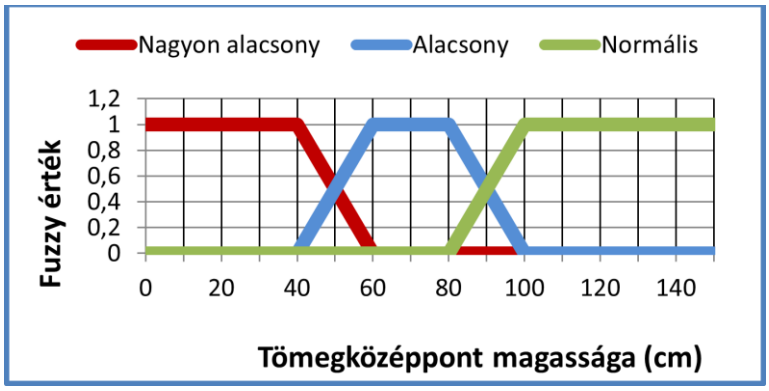
Elesés detektálás fuzzy változói és szabályai

A fuzzy változók halmazait, értékkészletét, és a tagsági függvények alakját és elhelyezését tapasztalati úton határoztam meg. Az értékek finomhangolását is kísérletek

alapján végeztem el. Minden fuzzy változó értékkészlete folytonos, a fuzzy halmazok számát pedig mindegyiknél háromra választottam, ami a tesztek alapján is elégnek bizonyult. Alább két fuzzy változót definiálok, a többi változó leírása a 3. függelékben található.

Tömegközéppont magassága (Y tengely mentén)

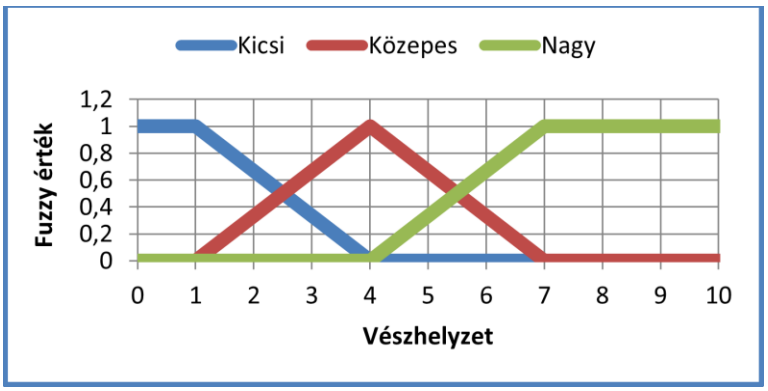
Értékkészlet: [0, 150] cm



4.4. ábra Tömegközéppont magassága fuzzy változó tagsági függvényei

Vészhelyzet

Értékkészlet: [0, 10]



4.5. ábra Vészhelyzet fuzzy változó tagsági függvényei

A fuzzy szabályokat intuitív módon határoztam meg, és a 4.1 és 4.2 táblázatok (szabálymátrixok) alapján rögzítettem. Az elesés kockázatának meghatározásánál egyértelmű, hogy ha közel nulla a mozgási sebesség, akkor minimális az elesés kockázata (első oszlop). Viszont akármilyen magasságban vagyunk, a hirtelen lefele irányuló mozgás mindig nagy kockázatott jelent.

Az elesés kikövetkeztetésénél a két időmérőt használjuk bemenetként. Kiemelném azt az esetet, amikor a földön fekvés ideje „Hosszú”, míg a másik „Zéró”. Ez fedi le azt az esetet, amikor lefeküdt az alany a földre, de már túl régóta, így a veszély kockázata megnő.

Pozíció (Y) \ Sebesség (Y)	Zéró	Lassú	Gyors
Normál	Alacsony	Alacsony	Közepes
Alacsony	Alacsony	Alacsony	Magas
Nagyon alacsony	Alacsony	Közepes	Magas

4.2 táblázat Elesés kockázat szabálymátrixa

Elesés óta eltelt idő / Földön fekvés ideje	Zéró	Rövid	Hosszú
Zéró	Alacsony	Alacsony	Alacsony
Rövid	Alacsony	Közepes	Közepes
Hosszú	Közepes	Közepes	Magas

4.3 táblázat Elesés szabálymátrixa

4.4 Gesztus és ADL aktivitás felismerés

A Kinect szenzor és a szoftver keretrendszer képességeinek köszönhetően könnyebbé vált olyan alkalmazások készítése, melyek a teljes testtel, vagy annak egy részével végzett tetszőleges mozgások felismerésére képesek. A gesztusfelismerés témakörét a megvalósítást tekintve, két részre lehet bontani: a teljesen általános gesztusfelismerés, és a „szabály alapú” felismerés. Az első esetben betanítás segítségével adjuk meg a felismerendő gesztusokat és így elméletben bármilyen gesztusokat készíthetünk. A második esetben a gesztusok sokkal korlátozottabban vannak létrehozva, és általában a szoftver forráskódjában rögzített szabályok alapján történik a felismerés. Ezeknél nincs lehetőség dinamikusan változtatni, vagy bővíteni a gesztusok tárházát, és leggyakrabban menük és virtuális valóság alkalmazásokban használják a navigációhoz. A továbbiakban a teljesen általános gesztusfelismerővel foglalkozom.

Ahhoz, hogy fel tudjunk ismerni egy gesztust, valamilyen módon le kell képezni (kódolni) a mozgást, amihez először meg kell határozni a jellegzetes elemeket (feature).

Általánosabb esetben, a képfeldolgozás témakörében, olyan jellegzetességeket használnak az objektumok azonosítására, mint például a képintenzitás egy adott területen, vagy különböző származtatott paramétereket [38], egy képfeldolgozási probléma kapcsán én magam is foglalkoztam ilyen mértékek meghatározásával [39]. Ez egy alacsonyabb feldolgozási szint, amit a Kinect megold helyettünk, így nekünk az eggyel magasabb szintű felismeréssel elég foglalkozni: a már megtalált és követett emberalak alapján a mozgásban megkeresni a jellegzetességeket, és ez alapján a gesztusokat felismerni. Tehát a feladat első fele egy olyan feature vektor definiálása, amely képes emberi (általában kézzel végzett) mozgások jellegzetességeit megragadni.

Az általános gesztusfelismerésben több elterjedt módszer létezik a gesztusok azonosítására. Ilyenek például a rejtett markov modelleket használó (HMM [28]) algoritmusok, a nagyszámú tanítómintákra alapozott módszerek, és a mintaillesztéses módszerek (DTW). A témakör igen komplex és szerteágazó, ezért nem volt célom az összes megoldás elemzése és összehasonlítása, inkább egy olyan megoldást próbáltam keresni, ami a legnagyobb valószínűséggel használható a Kinect szenzorral. Kutatómunkám során ismerkedtem meg a DTW (Dynamic Time Warping) algoritmussal, amit egy komolyabb, Kinect-et használó alkalmazásban sikeresen használtak gesztusfelismeréshez [40]. Ez nem jelenti azt, hogy ez az optimális módszer, de a rendszer első verziójában érdemes letesztelni AAL környezetben.

Megfontolandó, hogy milyen jellegű mozgásokat szeretnénk felismerni, az egyes gesztusokat milyen szinten szeretnénk megkülönböztetni? A felismerni kívánt mozdulatsorozatok várhatóan 2-3 másodperc hosszúak lesznek, hiszen képzeljük el egy ivás, vagy egy integetés mozdulatsorát. Ugyanakkor az emberek két mozdulatsort, ha nincsenek tudatában annak, hogy hogyan végzik, igen kis valószínűséggel fogják ugyan abban az ütemben elvégezni, tehát a probléma egy nemlineáris illesztést igényel. Lévén itt egy valósidejű felismerést szeretnénk megvalósítani, ezért már egy 5 másodperces intervallumon kívül nincs is értelme mintákat keresni, továbbá a gesztusok viszonylag rövid időtartama miatt már kis számú adat alapján is jól meg kell tudni különböztetni a gesztusokat.

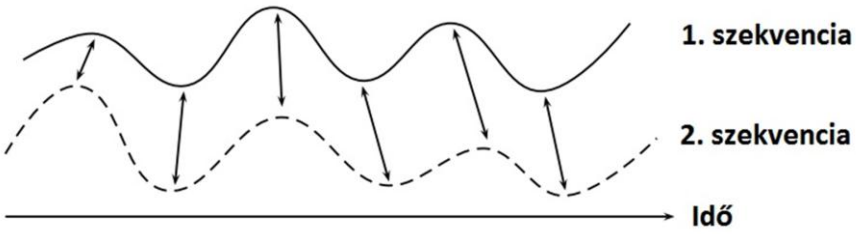
A gesztusfelismerés témakörében a legtöbb megoldás gyorsulásmérőket (Wii kontrollerhez hasonló megoldások), közeli kameraképeket és a képfeldolgozás apparátusát használja [41], de számos példa van a szórakozató iparban a Kinect használatára is. A Kinect-et használó szórakozató programok kivétel nélkül célzott gesztusfelismerést végeznek, tehát a program jellegéből adódóan pontosan tudható, hogy egy adott időszegmensben milyen gesztusokat (ráadásul általában csak egyet) kell megkeresni, így a probléma valójában egy

1:1 megfeleltetésnek tekinthető, míg AAL környezetben ez a probléma N:N-es, hiszen N darab gesztust szeretnénk folyamatosan felismerni, és nem tudható, hogy a gesztusok mikor kezdődnek, vagy fejeződnek be.

A következő szakaszban a DTW mintaillesztő algoritmust fogom bemutatni, amit a szoftver első verziója fog gesztusfelismerésre használni. A 6. fejezetben elvégzett tesztek alapján a dolgozat végén értékelem a Kinect szenzor adatokon alapuló DTW algoritmus használhatóságát AAL alkalmazásokhoz.

4.4.1 DTW algoritmus

Az algoritmust 1978-ban publikálták [42], eredetileg beszédfelismerésre tervezve, de általánosan minden olyan problémához használhatóvá vált, ahol optimális megfeleltetést kell megadni egy sablon (ismert minta) és egy teszt jel között, megengedve a tesztjel nemlineáris torzulását (elnyújtás, *warping*)., ahogyan azt a 4.6 ábra is szemlélteti. Felismerték, hogy egy ilyen probléma megoldása exponenciális komplexitású, ezért a DTW algoritmus a Dinamikus Programozás (DP) erejét kihasználva *O(nm)* időben oldja meg a feladatot, ahol *n* és *m* a két jel hossza.



4.6. ábra Két időfüggetlen jel szekvencia megfeleltetése (nyilak) egymásnak; forrás: [44]

Az alap DTW algoritmus egy optimális költségű utat keres a jelek által kifeszített két dimenziós térben, minimalizálva a teljes út költségét, ami lényegében a két jel távolságának minimalizálása, ami ugyanakkor maximalizálja a két jel hasonlóságát. Azóta egy másik speciális algoritmust is publikáltak, a Sztochasztikus DTW-t, mely kihasználja sok jel sztochasztikus tulajdonságait, tehát azt a jelenséget, hogy a jelsorozaton belüli váltások egy valószínűségi modellen alapszanak. Ilyen például a beszéd, ahol a szavak és hangok egymást követő sorozata mögött nyelvenként eltérő, de felismerhető modell van. Gesztusok esetében ilyen modell nem feltételezhető, hiszen a mozgások lehetnek teljesen függetlenek egymástól. Legfeljebb olyan átmenet valószínűségeket lehetne belevenni, ahol egyes végtagok bizonyos

helyzetekből nem kerülhetnek akárhova a térben (pl. könyök hajlásának szöge), vagy sok tanítóminta felhasználásával lehetne ilyen modellt építeni. Ez a megközelítés viszont túlmutat a diplomamunka keretein, így nem vizsgálom tovább.

A probléma megfogalmazása és az algoritmus definíciója [42,43,44] alapján a következő. Adott két idősor adat $X = x_1, x_2, \dots x_i, \dots x_{|X|}$ és $Y = y_1, y_2, \dots y_i, \dots y_{|Y|}$, ahol $|X|, |Y|$ rendre a két idősor hossza, és készítsünk egy $W = w_1, w_2, \dots w_K$ útvonalat, ahol $\max(|X|, |Y|) \leq K < |X| + |Y|$ és Kjelöli az útvonal hosszát, és a k-adik eleme az útvonalnak: $w_k = (i, j)$, ahol i, j az X, Yidősorok egy-egy indexe. Az útvonal a kialakított mátrix bal (1,1) eleméről indul és $(|X|, |Y|)$ eleménél ér véget. Ez biztosítja, hogy mindkét idősor minden indexe szerepelni fog az útvonalban. Egy másik kritérium, hogy a i és j monoton növekvő kell hogy legyen, tehát „visszalépés" nem fordulhat elő. Ezek után az optimális útvonal a minimális távolságú Wútvonal, ahol a távolságot

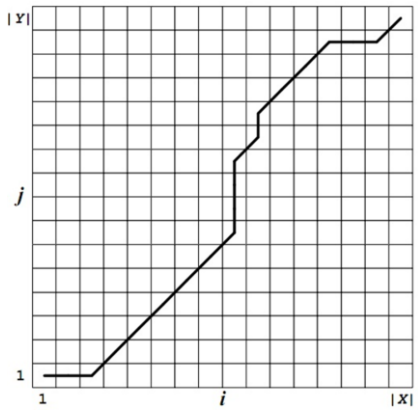
$$Dist(W) = \sum_{k=1}^{k=K} Dist(w_{ki}, w_{kj}) \tag{4.7}$$

alapján számoljuk, ahol a $Dist$ függvény tetszőleges távolságot definiálhat (de általában Euklideszi távolságot használnak), és $Dist(w_{ki}, w_{kj})$ két adatpont közötti távolság.

A dinamikus Programozás elvét az algoritmus futásánál használjuk fel. Az algoritmus által felépített D kétdimenziós mátrix mindkét dimenziója az időt reprezentálja (hosszuk $|X|, |Y|$), a mátrix $D(i, j)$ eleme az az X, Y sorozatok $i.,\ j.$ indexig tartó elemeiből konstruálható minimális útvonal hossza. Így a $D(|X|, |Y|)$ elem a két sorozat közötti optimális útvonal hosszát fogja tartalmazni. Minden egyes iterációban, amikor megpróbáljuk a két idősor $i.$ és $j.$ elemét megfeleltetni, ahelyett hogy kiszámolnánk a teljes problémára minden lehetséges variációt, az előző iteráció legjobb eleméből indulunk ki (ez a DP alapgondolata). Tehát feltételezve, hogy a $t.$ iterációban az optimális útvonal mentén haladtunk, a $t + 1$ iterációban ezt az információt felhasználva lépünk tovább, szintén minimális költségű utat képezve. Formálisan:

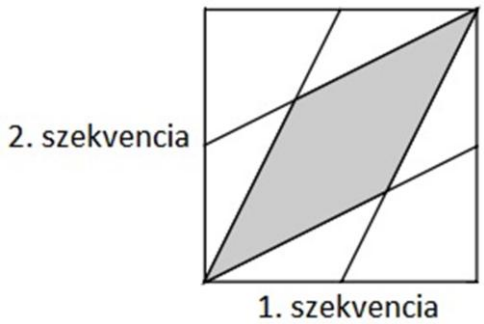
$$D(i, j) = Dist(i, j) + \min[D(i-1, j), D(i, j-1), D(i-1, j-1)] \tag{4.8}$$

Ezzel a módszerrel kitöltjük a teljes mátrixot, amely így minden lehetséges torzított útvonalra tartalmazni fogja a minimális útvonal hosszát. Ahhoz hogy az optimális útvonalat megtaláljuk a két idősor között, a D mátrix minden celláját (oszloponként, alulról felfelé, balról jobbra) ki kell tölteni, majd a $D(|X|,|Y|)$ cellából visszafelé indulva végigkövetni az útvonalat a minimális értékek mentén, ahogy ez a 4.7 ábrán is látszódik.



4.7. ábra DTW algoritmus által adott optimális útvonal; forrás: [44]

Az útvonal visszakövetése általában egy mohó keresési algoritmus, mégis többféleképpen végezhető el attól függően, hogy mekkora „területen” járjuk be a mátrixot. A DTW algoritmust számos „kényszerrel” tovább lehet gyorsítani. Ilyen például a meredekség kritériuma, mely megadja, a visszakövetésnél az útvonal minimális meredekségét, ezzel csökkentve a keresési teret. A 4.8 ábrán látható, hogy milyen mértékben korlátozza a keresést már egy 2-es meredekség korlát is.



4.8. ábra DTW meredekség korlát által behatárolt terület; forrás: [44]

További sebességnövelést mutat be a [43] FastDTW algoritmusa, mely iteratív módon először kisebb felbontású jelsorozaton végez előzetes becsléseket az útvonal irányára, amiket utána a teljes felbontású jelen szűkítő kritériumként használ fel.

A Kinect AAL gesztusfelismerésnél az alap DTW-t használom, kisebb módosításokkal, szükség esetén később tovább optimalizálható az algoritmus. A DTW algoritmus implementációját a [40] forrásra támaszkodva készítettem el, de a DTW-hez használt adatok kinyerését, és kódolását magam implementáltam az alább leírt koncepció alapján.

4.4.2 DTW algoritmus gesztusfelismeréshez

Gesztusfelismerésnél fontos leszögezni azokat a működési kritériumokat és elvárásokat, amik a gesztus felismerés szempontjából feladat specifikusak. Először bemutatom az alap DTW algoritmust, majd a gesztusfelismeréshez használható, specializált változatát.

function DTW(szekvencia1, szekvencia2) **returns** optimális útvonal, optimális költség
inputs:
szekvencia1, az első idősor szekvenciájának adatai (X)
szekvencia2, a második idősor szekvenciájának adatai (Y)
költségmátrix ←üres mátrix; m[1,1] = DIST(x_1, y_1); dimenziói: |X| és |Y|;
foreach x_i **in** szekvencia1
foreach y_j **in** szekvencia2
if(m[i, j-1]<m[i-1,j-1] **and** m[i,j-1] <m[i-1,j])
then m[i,j] = DIST(x_i, y_j) + m[i,j-1]
else if(m[i-1, j]<m[i-1,j-1] **and** m[i-1,j] <m[i,j-1])
then m[i,j] = DIST(x_i, y_j) + m[i-1,j]
else m[i,j] = DIST(x_i, y_j) + m[i-1,j-1]
end
end
return optimális költség ← m[|X|, |Y|]; optimális útvonal ← TRACE(m)

4.9. ábra Alap DTW algoritmus pszeudokódja

Az alap DTW algoritmus pszeudokódja a 4.9 ábrán látható, és lépései a következők:

1. Költségmátrix inicializálása: üres mátrixot inicializálunk. A futás első lépésénél az (i-1) és (j-1) érvénytelen indexekre mutatnának, ezért az m[1,1] az első két elem távolságára inicializáljuk.

2. Dinamikus programozással a költség mátrix kitöltése: minden indexre meghatározzuk azt a cellát, amiből tovább lépve a legkisebb marad a költség (a DIST függvény adja meg két adat távolságát).
3. Útvonal visszakövetése (TRACE) egy keresési algoritmussal, a megfeleltetés megadásához.
4. Kimenet: optimális útvonal, és annak költsége.

Rögtön látható, hogy esetünkben a harmadik lépés teljesen felesleges, hiszen nekünk csak a legjobb útvonal költségére van szükségünk, ami jellemzi az adott gesztus jelsorozatának és a teszt jelsorozat hasonlóságát. Ahhoz hogy a további módosításokat megértsük, előbb definiálni kell a problémát:

- Adott N darab, különböző hosszúságú jelsorozat, amik a gesztusokat kódolják.
- Egy m hosszú tesztsorozatban keressük a gesztusokat.

Ebből kifolyólag a tesztsorozat (puffer) bármely pozícióban végződhet gesztus, és a tesztsorozat hossza a vizsgálni kívánt időtartammal arányos. Az alap algoritmus továbbá annyiban módosul, hogy gesztusfelismerésnél a kimenet nem feltétlenül a $D(|X|, |Y|)$ értékű optimális útvonal költsége, ugyanis egy gesztus jelsorozatát nem feltétlenül akarjuk a teljes tesztsorozathoz hasonlítani, mivel az csak egy puffer, ami maximalizálja a keresési időintervallumot. Ezért a tesztsorozat minden lehetséges végződésével, ami még a gesztus hosszával összeegyeztethető, össze kell hasonlítani az adott gesztus jelét. A gyakorlatban ezt meg lehet oldani úgy is, hogy minden lehetséges hosszra lefuttatjuk a DTW algoritmust, de ez felesleges. Hiszen gondoljuk végig, hogy a tesztsorozat teljes hosszára lefuttatva a DTW-t már megkapjuk azokat a mátrixokat is, amiket a tesztjel rövidebb részsorozataira kapnánk, és így megkapjuk a minimális utak költségét, amik nem mások, mint az utolsó oszlop elemei. A módosított gesztusfelismerő algoritmusát (egy adott gesztusra) a 4.10 ábra pszeudokódja írja le, és a következő lépésekből áll:

1. Bemenet: gesztus jele, a teljes tesztjel (puffer), és a maximális meredekség.
2. Költségmátrix inicializálása, majd dinamikus programozással a költség mátrix kitöltése, figyelembe véve a maximális meredekség kritériumát.
3. Kimenet: a mátrix utolsó oszlopának legkisebb eleme.

```
function GESTURE-DTW(szekvencia1, szekvencia2, maxSlope) returns optimális költség
    inputs:
        szekvencia1: a gesztus szekvenciájának adatai (X);
        szekvencia2: a tesztjel (puffer) szekvenciájának adatai (Y);
        maxSlope: a maximális meredekség;
    költségmátrix ←m[0,0] = 0; m[a,0] = +inf; m[0,b] = +inf; (1<a<|X|, 1<b<|Y|)
    slopeI, slopeJ ←0
    foreach x_i in szekvencia1
        foreach y_j in szekvencia2
            if(m[i, j-1]<m[i-1,j-1] and m[i,j-1] <m[i-1,j] and slopeI< maxSlope)
                then m[i,j] = DIST(x_i, y_j) + m[i,j-1];
                slopeI = slopeI +1; slopeJ = 0;
            else if(m[i-1, j]<m[i-1,j-1] and m[i-1,j] <m[i,j-1] slopeJ< maxSlope)
                then m[i,j] = DIST(x_i, y_j) + m[i-1,j];
                slopeJ= slopeJ +1; slopeI= 0;
            else m[i,j] = DIST(x_i, y_j) + m[i-1,j-1];
            slopeI = 0; slopeJ = 0;
        end
    end
    return optimális költség ← MIN(m[a, |Y|]), ahol 1<a<|X|
```

4.10. ábra Gesztusfelismerő DTW algoritmus pszeudokódja

Az algoritmus paraméterként megkapja a maximális meredekséget is, ami megadja, hogy a DTW mátrix kitöltésénél egy adott mezőbe maximálisan mekkora meredekségű szakaszról léphetünk, tehát azt a maximális lépésszámot, amit a DTW mátrix kitöltésénél csak *i* vagy csak *j* mentén tehetünk. Ezzel korlátozható, hogy egy adott időpont adata maximálisan mennyi frame-mel torzítható (azaz feleltethető meg másik időbeli adatpontnak). Ha értéke 1, akkor gyakorlatilag csak az átló mentén vehetjük a minimális távolságot. Értéke nagyban befolyásolja a felismerést, hiszen nagyobb meredekség esetén a DTW sokkal jobban egymásra tudja illeszteni a jelsorozatokat. Fontos leszögezni, hogy az általános DTW algoritmusban a maximális meredekséget a legjobb útvonal visszakövetésre használják. Ezzel ellentétben, a Kinect DTW kódjában a maximális meredekség alapján kerül kitöltésre a DTW

mátrix, aminek első oszlopa és első sora plusz végtelenre van inicializálva, ezáltal a 4.8 ábrán látható fekete vonalakon kívül plusz végtelen értékek lesznek. Ezzel gyakorlatilag azokat a mezőket a végső minimális távolság meghatározásánál az algoritmus nem veszi figyelembe.

Az összes gesztusra lefuttatva az algoritmust megkapjuk, hogy egy adott időpillanatban minden egyes gesztus mennyire volt felfedezhető az adott pufferben. Minden gesztusnál a minimális útvonal értéke a puffer elejétől vett jelsorozatok és tetszőleges végződéseik, és a gesztus jelsorozat közötti távolságok közül a minimális. Erre az értékre úgy is tekinthetünk, mint egy hiba értékre, hiszen ez adja meg a távolságát az adott sorozattól, és a célunk egy gesztus elvégzésénél, hogy minél jobban hasonlítson az eltárolt mintához, azaz minimalizáljuk a hibát.

4.4.3 Gesztus reprezentálása

Definiálni kell, hogy egy gesztus milyen adatokkal reprezentálunk, tehát meg kell határozni a feature vektort. A DTW algoritmussal tetszőleges dimenziójú adatokat össze lehet hasonlítani, a lényeg csak az, hogy a két adatpontra meg kell tudni határozni a távolság értékét. Ez lehet Euklideszi távolság, de gyakorlatilag bármilyen függvény alkalmazható, ami számszerűsíteni tudja két adatpont „különbségét”.

A [40] által készített DTW algoritmusnál a szkeleton ízületi pontjainak 3D pozícióját használja fel a gesztusok leírására. Tekintve, hogy egy gesztus lényegében a végtagok mozgatásából áll, ez egy elég kézenfekvő megközelítés, és a Kinect AAL szoftver első verziójában ugyan ezt az irányelvet követem. Először felmerülhet a kérdés, hogy a DTW algoritmust használva, amelynek legnagyobb jelentősége, hogy időtől függetlenül tud két jelet összehasonlítani a torzítás (warping) miatt, nem tűnik e el olyan jelentős információ, ami egy gesztus esetében fontos lehet? A DTW egy gesztus dinamikáját is elmoshatja, hiszen pont az a célja, hogy a kis különbségeket ne vegye figyelembe. Ez a tesztelésnél fog kiderülni, de várhatóan ezt a problémát pont a felismerés kimenetének kiválasztása ellensúlyozza. Ugyanis mindig a minimális költségű elemet keressük meg az utolsó oszlopban, tehát azt a gesztus megfeleltetést, ami így dinamikájában globálisan is a legjobban hasonlít az eredetire, és maga a DTW csak a kis csúszásokat segít elmosni. Az ismertetett maximális meredekség kényszer (*slope constraint*) pedig segít kordában tartani az „elmosás” mértékét. Az előbb leírt okok miatt nem tervezem egyéb paraméterek (pl. végtag mozgások sebességének változása) felvételét a feature vektorba.

Összegezve tehát a gesztus reprezentációja:

- Ízületi pontok elhelyezkedése a térben (X, Y, Z) koordináta.
- A teljes gesztus *n* elemből áll, mindegyik elem egy képkockán rögzített ízületi pontok pozícióit kódolja.
- Egy elem hossza (dimenziója): K = ízületi pontok száma * 3 (a három koordináta miatt). Tehát a jelsorozatok elemei mind K dimenziós vektorok, melyek minden ízületi pont koordinátáit tömörítik egybe.

Két elem (adatpont) távolságának kiszámolásához egyszerű euklideszi távolságot használok:

$$d = \sqrt{\sum_{i=1}^{K} (pc_{1i} - pc_{2i})} \tag{4.9}$$

ahol *d* a távolság a két adatpont között, pc_{1i} és pc_{2i} pedig a két adatpont *i.* koordinátája.

4.4.4 Gesztusok eltolás és elforgatás függetlenítése

A gesztusfelismeréshez használt algoritmus lépéseit eddig a szakirodalomra és létező megoldásokra alapozva mutattam be. A továbbiakban ismertetem azokat a módosításokat és megvalósításokat, amik speciálisan csak a Kinect AAL rendszer részei.

A gesztusok előző pontban bemutatott reprezentációjának az a hátránya, hogy mivel a végtagok 3D térbeli koordinátáit használja, ezért alapesetben a pozíciótól és elforgatástól is függenek. A kérdés, hogy szükségünk van-e egy általános felismerőre? A 3. és 4. fejezetben bemutatott modell alapján a célunk olyan aktivitások felismerése, melyek a kontextustól, a Kinect AAL esetében a fizikai pozíciótól függenek. Tehát az ivás mozdulatsorát (ami például egy gyógyszer bevételének mozdulatsora is lehet) egy adott szekrény mellett akarjuk csak felismerni. Ugyanakkor a gesztusok kódolása miatt az alany már nagyon kis elmozdulása, vagy elfordulása (az alany arrébb lép egy 20 centimétert) felismerhetetlenné tenné a gesztust, ha abszolút koordinátákat használnánk, tehát szükséges a pozíció és elforgatás függetlenítés.

A gesztusok kódolása így a következőképpen algoritmizálható:

1. Ízületi pozíciók lekérése a keretrendszertől (Kinect koordinátarendszere).
2. A pontok transzformálása a valós világ kalibrált koordinátarendszerébe.
3. Pontok elmozdulás függetlenítése egy kitüntetet pont (test „origója”) segítségével: a felsőtesthez tartozó ízületi pontok (váll, könyök, kézfej) eltolása

a nyak ízületi pontjával. Az alsótesthez tartozó pontok (csípő, térd, lábfej) eltolása a törzs ízületi pontjával.

4. Pontok elforgatás függetlenítése: a felsőtest nézési irányának Z komponense és a kalibrált koordinátarendszer Z tengelye által bezárt szöggel elforgatni az összes pontot.

Az első két lépés a Kinect kalibrációnál ismertetet transzformációs eljárással végezhető el. A harmadik lépés egy egyszerű transzlációs transzformáció, tehát a test „origó" koordinátáit rendre ki kell vonni a pontok koordinátáiból (vektorok eltolása a kalibrált koordinátarendszer origójába). A negyedik lépés a legösszetettebb, hiszen ahhoz, hogy az Y tengely menti elforgatást elvégezhessük először meg kell határoznunk az ember nézési irányának vektorát, ami a következő vektoriális szorzattal határozható meg:

$$\vec{f} = (\overrightarrow{v_2} - \overrightarrow{v_1}) \times (\overrightarrow{v_0} - \overrightarrow{v_1}) \tag{4.10}$$

ahol $\vec{f}$ a nézési irány vektora (forward vector), $\overrightarrow{v_0}, \overrightarrow{v_1}, \overrightarrow{v_2}$ pedig rendre a jobb váll, törzs és bal váll vektora. $\vec{f}$ –et normalizálva egy origóból kiinduló vektort kapunk, ami az aktuális felsőtest „nézési" irányát határozza meg. A pontokat ennek a vektor Z komponensének és a kalibrált koordinátarendszer Z tengelyének szögével kell az origóban az Y tengely körül elforgatni. A fenti lépések eredménye képen minden ízületi pontot eltolás és elforgatás függetlenül kaptunk meg.

4.4.5 Gesztusfelismerő működése

A végső algoritmusban nem a DTW algoritmus által számolt útköltségeket használom a gesztusok hasonlóságának meghatározására, hanem először normálom a DTW eredményét:

$$confidence_i = 1 - \frac{DTW(g_i, T)\ /(J * L)}{maxError} \tag{4.11}$$

ahol $DTW(g_i, T)$ adja az *i.* gesztus és a T puffer legkisebb útköltségű megfeleltetését (távolság érték, a gesztus hibája), J az ízületi pontok száma, L a gesztus hossza, $maxError$ pedig egy konstans, amit tapasztalati úton kell meghatározni, és egy felső korlátot képez a gesztusok felismeréséhez.

```
function GESTURE-RECOGNITION (szekvenciák, puffer,maxSlope) returns confidences
    inputs:
        szekvenciák: a gesztus szekvenciák;
        puffer: a tesztjel (puffer) szekvenciája;
        maxSlope: a maximális meredekség;
    confidences(g_i) ←0 minden i-re
    foreach g in szekvenciák
        mindist = GESTURE_DTW(g,puffer,maxSlope);
        confidences(g_i) = NORM(mindist);
    end
    return confidences
```

4.11. ábra Gesztusfelismerés pszeudokódja

Az ízületi pontok számával és a gesztus hosszával azért kell leosztani, hogy kiátlagoljuk a hibát a pontok között, erre pedig azért van szükség, mert különböző gesztusokhoz különböző számú ízületi pontot használhatunk, és a gesztusok hossza is különbözhet. A $confidence$ értéke 0-nál kisebb értéket is felvehet (ha az átlaghiba nagyobb a $maxError$ -nál), amiket 0-val kell egyenlővé tenni. A $confidence$ adja meg tehát egy gesztus felismerésének bizonyosságát, és értékkészlete [0, 1] között folytonos. A teljes felismerő algoritmus egy iterációja, N darab gesztust feltételezve a 4.11 ábrán látható. A puffer utolsó elemét minden GESTURE-RECOGNITION lefutása előtt el kell dobni, és első elemként az új adatot (transzformált ízületi pontokat kódoló vektor) felvenni.

4.4.6 Gesztusfelismerő bekötése a következtető rendszerbe

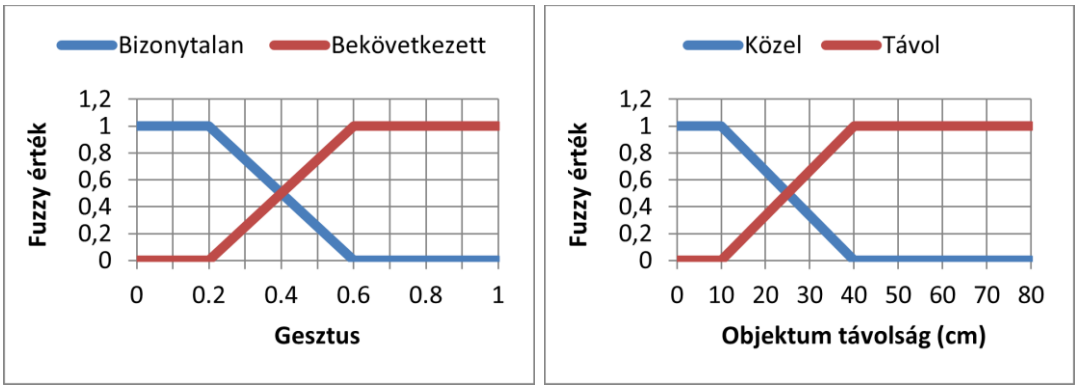
Eddig a gesztusok felismerését önmagukban vizsgáltuk, és a bemutatott algoritmus eredményeként minden egyes Kinect-ből érkező adat frame után megkaptuk az előre

betanított gesztusok *confidence* értékét. A következtető rendszer ezeket az értékeket fogja felhasználni és tovább szűrni, hogy megállapíthassuk milyen ADL cselekvéseket hajtott végre a megfigyelt alany. A 3. fejezet 3.3 szakaszában bemutatott több szintű ADL aktivitás modellezése alapján dolgoztam ki a gesztus alapú ADL aktivitások felismerését. A kontextus modell leírásánál ismertetettem, hogy a Kinect AAL rendszer minden időpillanatban meghatározza az alanya távolságát minden azonosított objektumtól. A gesztusokat valamilyen kapcsolatba kell hozni ezekkel az objektumokkal, tehát a kontextus alapján valamilyen módon súlyozni érdemes a gesztusokat. Ahelyett, hogy ezt a súlyozási formulát explicit módon megadnám, itt is a fuzzy következtető rendszerre hagyatkozom, és jól definiált fuzzy változókat és tagsági függvényeket adok meg helyette. Tehát lényegében a kontextus jelenti a felismerés 1. szintjét, a gesztus a 2. szintet, és az ezek alapján fuzionált eredmény a tényleges ADL aktivitás (3. szint).

A gesztusfelismerő által minden gesztus bekövetkezésének mértéke a [0, 1] intervallumra normált valós számban lett meghatározva (*confidence*). Az azonosított objektumoktól (pl. asztal, szekrény) való távolságot a kalibrált térben, abszolút léptékben (mm) képes a Kinect AAL szoftver megadni. Ezt a két bemenetet felhasználva állapítható meg egy ADL aktivitás bekövetkezésének mértéke az alábbi fuzzy változó definíciók alapján.

Gesztus és Objektum távolsága

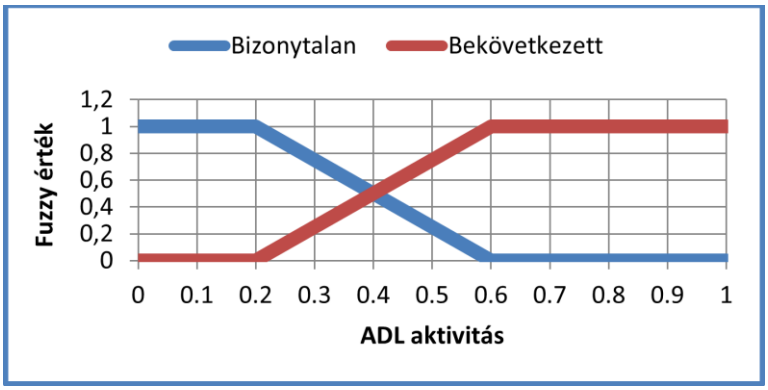
Értékkészlet: [0, 1] és [0, 80] cm



4.12. ábra Gesztus és Objektum távolság fuzzy változók tagsági függvényei

ADL aktivitás

Értékkészlet: [0, 1]



4.13. ábra ADL aktivitás fuzzy változók tagsági függvényei

Az elesés detektálásnál használt fuzzy változókkal ellentétben itt csak két halmazt definiáltam, mert ennyi is elégnek bizonyult a probléma leírásához. Mivel tetszőleges számú gesztus és objektum adható meg a kontextus modellben, ezért a változók létrehozását és a szabályok generálását dinamikusan kell végezni. Minden objektum és gesztus változóhoz az objektum és gesztus név alapján egy egyedi azonosítót rendelek, és a 4.12 ábrán látható fix értékekkel és halmazokkal fedem le. Minden objektum és gesztus párhoz két szabályt definiálok:

- HA Gesztus = Bekövetkezett ÉS Objektum = Közel AKKOR ADL aktivitás= Bekövetkezett.
- HA Gesztus = Bizonytalan VAGY Objektum = Távol AKKOR ADL aktivitás = Bizonytalan

Ezzel a két szabállyal azt az elvárt működést lehet leírni, hogy egy ADL aktivitást lehetőleg csak akkor észleljünk, ha egy adott objektum közelében vagyunk, és a gesztust nagy bizonyossággal elvégezték. Ha valamelyik nem teljesül, akkor bizonytalanok vagyunk az ADL aktivitás bekövetkezésében is. Gyakorlatilag ezzel valósul meg a többszintű ADL aktivitás modellezés, tehát amikor az alacsonyabb szintű információkat (pozíció a térben, fizikai kontextus) felhasználva a magasabb szintű aktivitásokat (pl. ivás) lehet szűrni, vagy súlyozni.

4.4.7 További ADL aktivitások

Az előző szakaszban bemutattam, hogy a legalsó ADL aktivitás modellezési szintre épülő további adatokkal hogyan lehetséges magasabb szintű felismerést végezni. Ugyanakkor a Kinect AAL rendszernek gesztusoktól független ADL aktivitásokat is tudnia kell kezelni, mint például a szobába való be- és kilépés, a szobában való mozgás, és esetleg bizonyos kitüntette objektumok használatát (pl. ágy, vagy szekrény). Az eddig megtervezett rendszer esetében ez azt jelenti, hogy csak az ADL aktivitás első modellezési szintjét felhasználva akarunk bizonyos információkat kikövetkeztetni az alany aktuális aktivitásáról.

Szobába be- és kilépés, és a szobában való mozgás.

Az elgondolás alapját a 4.1.2 szakaszban már ismertettem, és lényege, hogy nem szükséges további elemeket bevezetni a rendszerbe, hiszen a rendszer eleve minden időpillanatban megadja az azonosított objektumoktól és térrészektől való távolságot. Ezt felhasználva, a következtető rendszer bővítése nélkül megállapítható, hogy az alany mely térrészben van. Tehát jelezze külön térrész (szektor) a szoba mindegyik kijáratát (ajtók), és osszuk fel a szobát annyi szektorra, ahány térrészt meg szeretnénk különböztetni a követés szempontjából. Az ADL aktivitások explicit megadása nélkül a rendszer automatikusan meghatározza a legközelebbi térrészt, az ADL aktivitás pedig a térrészek közötti váltás eseményeként értelmezhető. Összesítve, tehát a legalacsonyabb szintű ADL aktivásokat meghatározó algoritmus a következő lépésekből áll:

1. Alany pozíciójának meghatározása.
2. Távolságok kiszámítása mindegyik térrészre (és objektumra).
3. A legközelebbi térrész kiválasztása az aktuális tartózkodási szektornak.
4. Szektor váltások esetében esemény jelzése a rendszer visszajelző felületén.

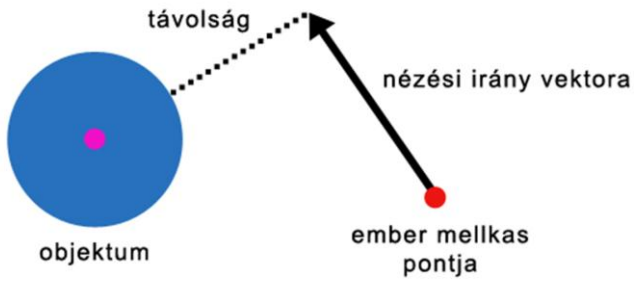
Így egy „szobába belépés” ADL aktivitást akkor tekinthetünk megtörténtnek, ha eddig nem észleltük az alanyt, de egy tetszőleges szektorban megjelenik. A „kilépés a szobából” esemény pedig akkor aktivizálódik, ha a követett alany a megadott „szoba” térrészből átlép az „ajtó” térrészbe, majd a kamera elveszti az alanyt. Észrevehető, hogy a 3. lépésnél pusztán egy minimum keresést végzünk, ami arra a feltételezésre épül, hogy a teljes tér helyesen van felosztva. Ugyanis amennyiben csak két térrész van megadva („ajtó” és „szoba”), és a „szoba” nem fedi le a teljes teret, akkor elképzelhető, hogy az „ajtó” akkor is aktivizálódik, amikor még a szobában van az alany, de túl messzire kerül a szobát szimbolizáló térrésztől.

Objektumok használata

Az előző elgondolásra alapozva terveztem meg az objektum használat ADL aktivitásit is, annyi különbséggel, hogy nem elég pusztán a legközelebbi objektumot megkeresni, hanem további paramétereket is célszerű felvenni a rendszerbe. Egy objektumhasználat bekövetkezésének vizsgálatánál olyan paramétereket érdemes figyelembe venni, hogy az alany milyen távolságra van az objektumtól, megállt-e mellette, vagy felé néz-e. A gesztusfelismerésnél már ismertettem a nézési irány vektorának meghatározását, amit ehhez a feladathoz is fel lehet használni. Toljuk el a normalizált nézési irány vektorát a követett alany szkeleton mellkas pontjába, majd szorozzuk fel az egységnyi hosszt azzal az értékkel, amekkora távolságból az objektumokat az alany közelében akarjuk tekinteni:

$$\vec{v} = \vec{f} * d_m + \vec{t} \tag{4.12}$$

ahol $\vec{f}$ a normalizált nézési irány, $\vec{t}$ a mellkas vektora a kalibrált térben, d_m a kívánt minimális távolság (ez lesz a vektor hossza), és $\vec{v}$ a végső nézési irány vektor, amit felhasználhatunk a végső távolság számításhoz. Annak megállapítására, hogy az alany az objektum felé néz, és közel van hozzá, a $\vec{v}$ vektort használjuk fel, és a távolság számítását a 4.2 egyenlet alapján végezzük az x_p, y_p, z_p értékeknek a $\vec{v}$ vektor koordinátáit behelyettesítve. Vizuálisan, két dimenzióban a 4.14. ábra szemlélteti az elképzelés lényegét, ami egy felülnézeti képnek is tekinthető. A nézési irány vektorát úgy kell elképzelni, mint

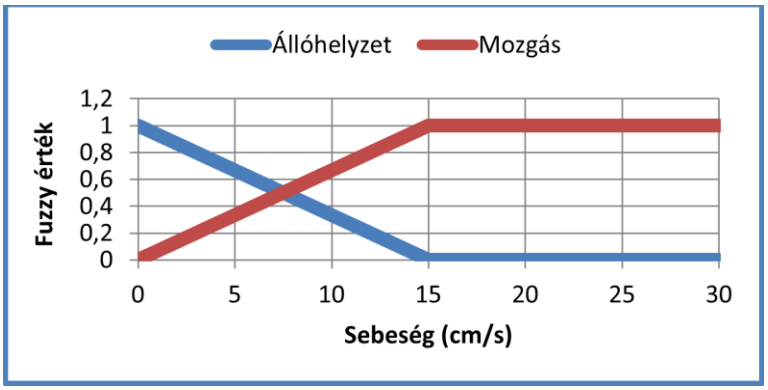


4.14. ábra Alany nézési irány függő távolságának meghatározása az objektumoktól

egy, az alany mellkasából kiálló, adott hosszúságú vektort. Az ábrán is látszódik, hogy ha az alany közvetlenül az objektum (kék kör) mellett állna, de a nézési irány vektora az objektumtól elfele mutatna, akkor az alany-objektum távolság pont a nézési irány vektorának

hossza. Az alany tömegközéppontjának és az objektumok távolsága helyett az ábrán látható szaggatott vonallal jelzett távolságot használtam fel az „objektum használat”ADL aktivitásokhoz, ami egy érték számítására egyszerűsíti a nézési irány függő távolság számítást.

A második felhasznált paraméter az alany háromdimenziós sebességvektora, melyet az elesés detektálásnál ismertettem. A két paramétert és a kérdéses ADL aktivitást is a fuzzy rendszer segítségével kötöttem össze. A 4.12 ábra jobb oldali fuzzy változóját használtam az egyik bemeneti változónak, az aktuális sebességet (4.15 ábra) másodiknak, kimenetnek pedig a 4.13 ábra fuzzy változójával megegyező „Objektum használat” változót.



4.15. ábra Sebesség fuzzy változók tagsági függvényei

A változókat összekötő szabályokat az ADL aktivitáshoz használtakhoz hasonlóan adtam meg:

- HA Objektum = Közel ÉS Sebesség = Állóhelyzet AKKOR ADL aktivitás= Bekövetkezett.
- HA Objektum = Távol VAGY Sebesség = Mozgás AKKOR ADL aktivitás = Bizonytalan

Az „objektum távolság” bemeneti fuzzy változó habár ugyan az, mint a gesztusfelismerésnél használt, de a bemeneti értékének a nézési irány függő távolságot adom meg.

4.5 Összefoglalás

Ebben a fejezetben bemutattam a 3. fejezetben felvázolt rendszer főbb komponenseinek megvalósításának lépéseit, megindokolva a fontosabb tervezői döntéseket. A fejezet elején

definiáltam a rendszer funkcionális és fizikai modelljét, mely a kontextus érzékeny alkalmazás alapját képzi. A célfeladatok és a Kinect adatreprezentálása indokolttá tette egy kalibráló algoritmus beépítését a rendszerbe. Ennek segítségével minden Kinect-ből érkező adatot rögtön a valós fizikai teret modellező virtuális tér koordinátarendszerében kapunk meg, megkönnyítve ezzel a fizikai modell megadását, és az elesés detektálást is. A Kinect AAL rendszer egyik alapvető célja (feladata) a kontextus függő gesztusok felismerése, amit a Kinect AAL rendszerhez specializált, DTW alapú gesztusfelismerő algoritmus valósít meg. A megtervezett algoritmus képes elforgatás és elmozdulás függetlenül azonosítani az előre betanított gesztusokat.

A Kinect AAL rendszerben az ADL aktivitások egy része kontextus függő gesztusokként valósul meg. A gesztusok bármilyen, végtagokkal végzett mozdulatsorokat takarhatnak, és ezek alapján az ADL aktivitásokat - a fizikai modell objektumait szűrőként felhasználva – automatikusan ismeri fel a rendszer. A fizikai modell objektumaihoz a modell leíró fájlban akár futási időben is hozzárendelhetők a betanított gesztusok, amik alapján a Kinect AAL rendszer dinamikusan építi fel a fuzzy következtető rendszerhez szükséges elemeket. A fejezet végén bemutattam a fizikai kontextustól függő „mozgás a térben” és „objektum használat” ADL aktivitásokat felismerő algoritmusokat, melyeknél az „objektum használat” olyan komplex paraméterek alapján lett megállapítva, mint az aktuális sebesség és a nézési irány.

5.Szoftver a Kinect AAL rendszerhez

A Kinect újszerű technológiája, a szoftver keretrendszerek gyakori frissítése miatt a Kinect AAL rendszer szoftverjét komponens alapúra tervezem, így a tervezési és implementálási szakaszokban is modulárisan lehet cserélni az egyes komponenseket, és bővíteni újakkal, továbbá az általam fejlesztett komponenseket is egyszerűbben lehet kezelni. A szoftver keretrendszerek elemzésénél bemutattam a két keretrendszer erősségeit és gyengéit, és végül a nyílt forráskódú OpenNI használata mellett döntöttem. A szkeleton felismerés tekintetében alig volt különbség a két rendszer között, de az OpenNI támogatottsága jelenleg (2012.május) jóval nagyobb, és a köré épülő technológiák (NITE) sokkal dinamikusabba fejlődnek, így nagyobb bennük a potenciál. Továbbá az OpenNI a Kinect szenzoron kívül két másik típusú szenzort, illetve több szenzor együttes használatát is támogatja (lásd. 2.3 ábra), ami a rendszer későbbi verziójában lehetővé tenné a szenzorok összehangolásával egy sokkal pontosabb és kiterjedtebb rendszer kiépítését.

Programozási nyelvnek .NET 4.0 C# -ot használok, mivel ez támogatja mind a két keretrendszert, így egy későbbi átállás, vagy egyes komponensek felhasználása a másik keretrendszerben egyszerűbbé válik. A szoftverben a komponenseket úgy alakítom ki, hogy mindegyik jól definiált interfészekkel rendelkezzen, és ezekre az interfészekre illeszkedve használják fel az adatokat. A komponensek így függetlenek maradnak a többi komponens megvalósításától. Ezzel biztosíthatom, hogy egy komponens cseréje nem befolyásolja majd a többi működését, és minimális kódolással beilleszthető a rendszerbe.

A szoftver tervezésével és implementálásával egy olyan platform elkészítése volt az elsődleges cél, ami a Kinect szenzorra épülő AAL rendszert valósít meg. A teljes diplomamunka fontos részét képezte ennek a szoftvernek a kialakítása, amely architektúrájának és komponens alapú megközelítésének köszönhetően (későbbi bővítésekkel) más szenzorok adatait is képes integrálni a teljes rendszerbe. Az egész struktúrára úgy lett kialakítva, hogy egy új szenzor integrálása egy új adatlekérő modul implementálását és bekötését, majd a szenzoradatokat felhasználó függvények kis módosítását igényli csak. A teljes Kinect AAL szoftver ezzel egy olyan rendszert valósít meg, mely magába foglal:

- egy Kinect-hez specializált, általános gesztusfelismerő megoldást,

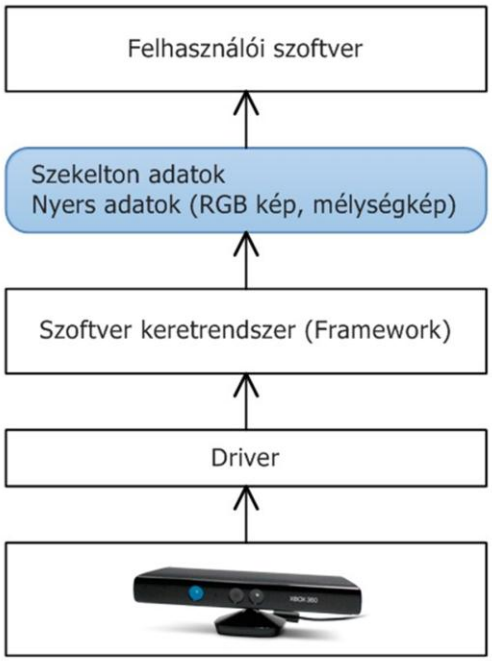
- a Kinect kalibrálásához és a valós tér leképzéséhez szükséges algoritmusokat és modellező eszközöket (kontextus modellezés),
- egy általánosabb, összefogó következtető rendszert, mely a szenzoradatokat integrálja,
- és egy egységes felületet az AAL rendszer működésének követéséhez.

5.1 Architektúra

A rendszer három rétegből áll, amelyek lényegében maguktól adódnak, a Kinect kialakításának köszönhetően. A rétegek (legalsótól kezdve):

1. hardver (driver)
2. keretrendszer (framework)
3. felhasználói szoftver

A hardver réteg a Kinect-et, és az azt elérő drivert foglalja magába. Ettől a rétegtől a keretrendszer kapja a nyers adatokat. A keretrendszer rétege a Kinect SDK-t, vagy az OpenNI-t foglalhatja magába, de a szoftver első verziója a nyílt OpenNI-t használja, ami a nyers adatokat dolgozza fel, és bizonyos eljárásokat és adatfolyamokat biztosít a felhasználói



5.1. ábra Kinect AAL program architektúrája

szoftvernek. E réteg használata nélkül is készíthető felhasználói szoftver, de ehhez saját eljárásokat kellene készíteni, hogy az infravörös szenzor által kinyert mélységtérkép és az RGB kamera jeleit feldolgozzuk. A keretrendszer ezt megteszi számunkra, de ugyanakkor, szükség esetén, minden nyers adatot is elérhetünk rajta keresztül. A legfelső réteg maga a felhasználói program, mely magába foglalja a szenzoradatok feldolgozását (szenzorfúzió, intelligens algoritmusok), az üzleti logikát, és a felhasználói grafikus interfészeket (GUI).

A hardver rétegtől a keretrendszerig nem befolyásolható az adatfolyam, de erre nincs is szükség, hiszen mindkét vizsgált keretrendszerben (Kinect SDK, OpenNI) elérhető minden nyers adat is. A keretrendszer és a felhasználói szoftver belső komponensei között kell jól definiált interfészeket kialakítani. A program komponensei így a keretrendszertől függetlenül tudnak működni, biztosítva ezáltal, hogy egy későbbi keretrendszer csere csak nagyon kismértékben befolyásolja a programot, és az algoritmusokat megvalósító kódokat egyáltalán, vagy minimális módosítással újra fel lehessen használni.

5.2 Felhasználói szoftver komponensei

A szoftver teljes architektúrája az 5.2 ábrán látható. A felhasználói beavatkozást biztosító komponens maga a felhasználói interfész, mellyel minden komponensig visszaterjeszthető bizonyos vezérlési információ vagy adat. Ezek a beavatkozási lehetőségek explicit nem szerepelnek az ábrán, de a végső rendszer részei. A komponensek és funkcióik:

Kamerát és a szoftver keretrendszert reprezentáló komponensek

A keretrendszert közvetlenül éri el a kamerát, ahonnan a nyers adatok érkeznek, és biztosítja azokat az eljárásokat, amiken keresztül az adatfeldolgozó a szkeleton és képadatokat kapja. A kamerát reprezentáló komponens függ a hardvertől (Kinect kamera), a keretrendszer pedig a kamerától. Az adatfeldolgozó a szoftver keretrendszer interfészein keresztül éri a szükséges adatokat, és elfedi a keretrendszert és a kamerát a szoftver többi komponensétől.

Adatfeldolgozó komponens (Data Processor)

A keretrendszerből érkező adatokat transzformálja olyan adatfolyamokká, melyeket a többi komponens felhasználhat. Ez a felhasználói program keretrendszer függő része, ami interfészt biztosít a többi komponens számára az adatok szabványos eléréséhez. Ez az egyetlen komponens, ami a kamerát reprezentáló komponensen keresztül, illetve közvetlenül

is hozzáfér a keretrendszer adatfolyamaihoz. A keretrendszerből érkező adatok elsőszintű fuzionálását (összefésülését) és transzformálását végzi. A transzformált adatok felhasználás előtti pufferéléséért (átmeneti tárolás) is az adatfeldolgozó komponens felel.

Adattár (Data Storage)

Ez egy absztrakt komponens, mely a perzisztens adattárolást reprezentálja, de a konkrét megvalósítása mindig más komponensen belül történik. Az adattár komponensek adatbázisként, fájlként (pl. idősor adatok, kalibrációs adatok), vagy alkalmazás beállításként valósulhatnak meg. A program az adattárban tárolja a kalibrációs adatokat, a gesztusok és a rendszer-modell adatait. Az adattáron keresztül éri el az adatfeldolgozó a feldolgozáshoz szükséges információkat (kalibráció, kontextus modell).

Kalibráló komponens

Ez a komponens végzi a kalibrálást, és menti el a kalibrációs adatokat az adattárba és a háttértárra.

Következtető rendszer (intelligens következtető rendszer, Inference System)

Ez a komponens felelős az adattárból és az adatfeldolgozó puffereiből kiolvasott adatok, és a gesztusfelismerő rendszer alapján a következtetések létrehozásért, és az események generálásáért.

Gesztusfelismerő és betanító komponens (Gesture Analyzer)

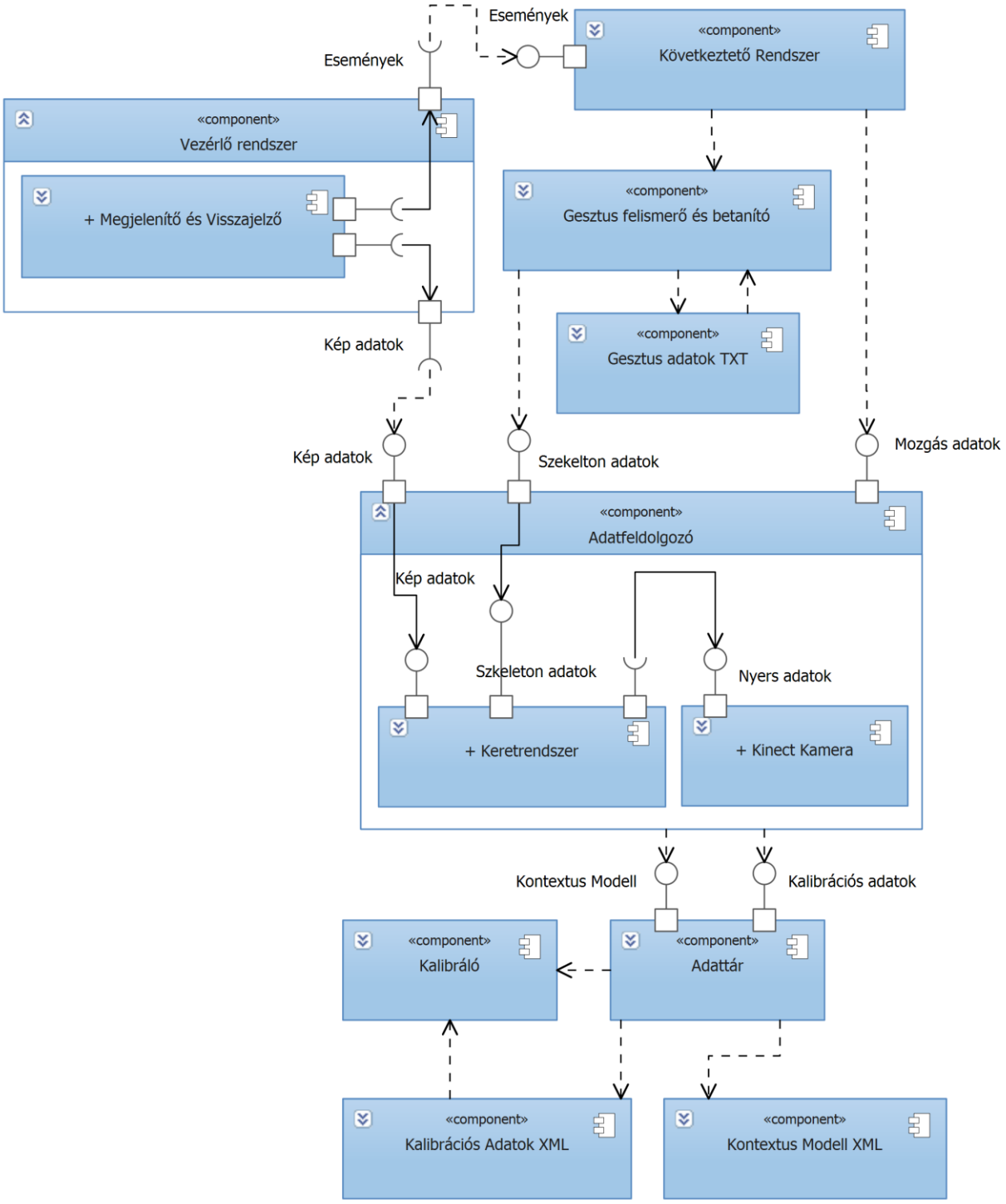
Előre betanított gesztusok azonnali felismerését végző komponens. Az adatfeldolgozó komponensből érkező adatokat valós időben értelmezi, és ezek alapján generál gesztus specifikus adatokat, amiket a következtető rendszer dolgoz fel. A gesztus adatbázist a háttértárról tölti be.

Vezérlő rendszer (Controller System)

Ez a program magja, mely a többi komponens létrehozásáért, vezérléséért, és a felhasználói interfészen történő beavatkozások rendszerbe történő visszacsatolásáért felelős. A következtető rendszer által generált események kezelését is ez a komponens végzi.

Megjelenítő és visszajelző komponens (Viewer and Notification System)

Ez a megjelenítésért, és az események követéséért felelős komponens. Része a vezérlő rendszernek és megvalósítja a felhasználói interfészt, amin keresztül jelez a rendszer, és amin keresztül a rendszerbe be lehet avatkozni.



5.2. ábra Kinect AAL program szoftverkomponensei

5.3 Komponensek működése és kapcsolata

5.3.1 Valós idejű működés

A valós idejű működés egy AAL rendszer esetében két szinten jelenik meg. Az első szint az adatfeldolgozás, tehát amikor a szenzorokból érkező adatokat rögtön (vagy pufferelve) dolgozzuk fel, és a szenzor adatgenerálásának ütemében kell működnie. A második szint a visszajelzés és a kezelőfelület sebességére utal, tehát a rendszer által feldolgozott adatokat milyen időközönként frissítjük, illetve az AAL rendszer eseményeit milyen időközönként generáljuk. A kettő között nagyságrendbeli különbség van, hiszen az adatfeldolgozás sebessége rendszerint 30-40ms, míg a visszajelzések másodperces, vagy akár perces nagyságrendben lehetnek, de fontos kijelenteni, hogy mindkettő valós idejű működés.

A Kinect AAL rendszert megvalósító szoftver valós időben dolgozza fel a Kinect szenzorból érkező adatokat. A valós idejű adatfeldolgozás és visszajelzés megköveteli, hogy az adatfolyamok és események szinkronizálva legyenek, illetve minden művelet szál-biztosan legyen megvalósítva. A szoftverek sok esetben a számításigényes feladatokat a program keretében futó, de különálló feldolgozási egységben (*thread*, szál), aszinkron módon végzik. Ennek legnagyobb előnye, hogy a többi szál, mely különböző algoritmusok futtatásáért felelős így egymástól elkülönülve, egymást nem megszakítva, vagy várakoztatva tud futni.

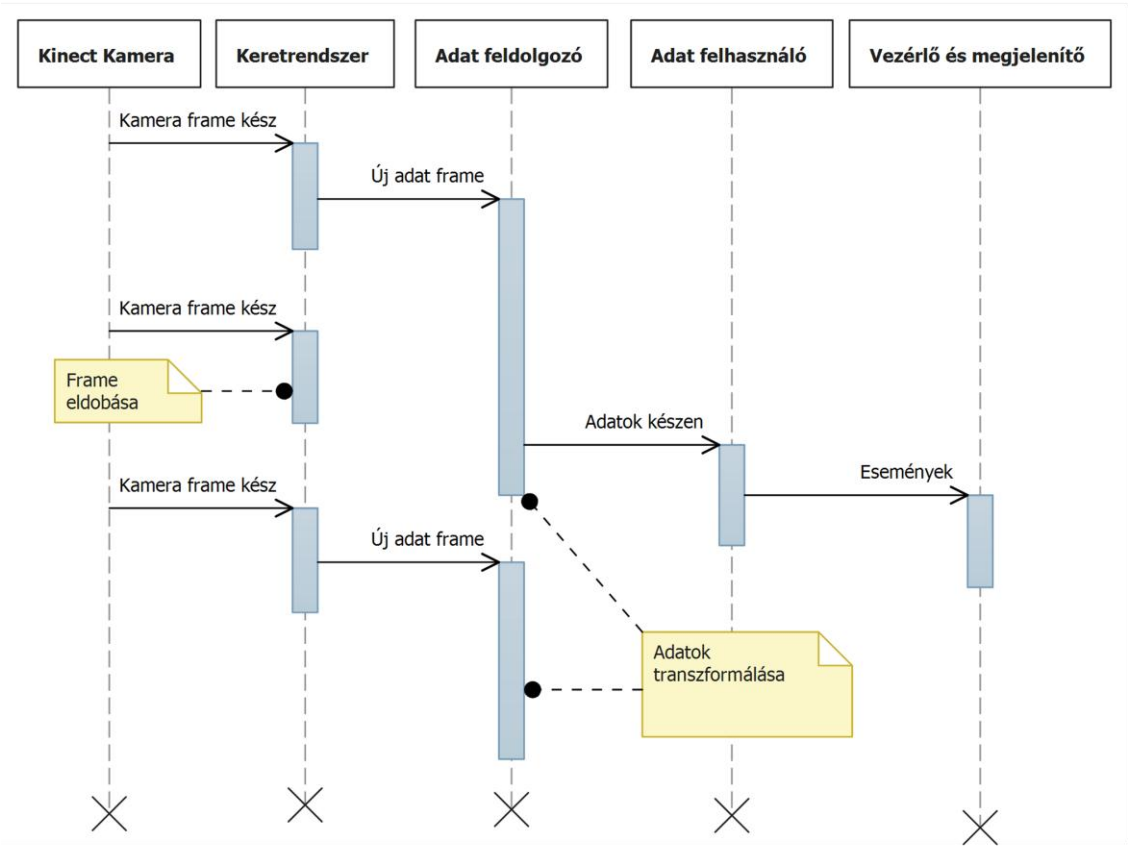
A keretrendszert magába foglaló osztály a kamera elindítását követően egy külön feldolgozási szálban kéri le az adatokat a szenzortól. Az adatok lekérése szinkron módon történik, tehát az új adat megérkezéséig az adott szálat blokkolja, ezért szükséges külön szálban futtatni. Amint rendelkezésre áll az új adat (data frame), a szálon belül esemény generálódik, mely jelezi az adatfeldolgozó komponensnek, hogy frissültek az adatok és lekérhetők a szoftver keretrendszer API-ján keresztül. A kamera 30 FPS (30 képkocka, tehát 30 adat frame másodpercenként) sebességgel működik, így az adatfeldolgozó komponensnek 33ms belül kell minden szükséges transzformációt elvégeznie. Tovább a gesztusfelismerő és következtető rendszer algoritmusainak is ezen a műveleti sebességen kell futnia, különben adatvesztés veszélye áll fenn. Ugyanakkor a rendszert nem szabad ettől függővé tenni. Ha az adatfeldolgozás előző ciklusa még nem fejeződött be, de a keretrendszer jelzi az új adat frame készenlétét, akkor nem kezdhetjük el párhuzamosan feldolgozni azt, hiszen az adatok időben egymást kell, hogy kövessék (idősor adatok), illetve ezzel csak tovább lassulna a feldolgozás, ami végül a rendszer teljes túlterhelődéséhez és adattorlódáshoz vezetne. Ezért fel kell

készíteni a rendszer feldolgozási folyamatát az adatvesztésre, tehát a feldolgozás sebességének ingadozása vagy lassúsága miatt bizonyos framek eldobására.

Mivel az adatfeldolgozás algoritmusai számításigényesek, ezért azokat külön háttér szálban futtatja a szoftver. Új adat frame érkezésekor, amennyiben nem fejeződött be az előző feldolgozási ciklus (tehát a háttérszál még fut), a frame eldobásra kerül. Ezzel biztosítható, hogy az új adatok véletlenül sem írhatják felül az adatfeldolgozó puffereit még a feldolgozás közben, ezzel inkonzisztensé vagy hamissá téve az adatokat. Természetesen arra kell törekedni, hogy a feldolgozási szál futási ideje 33ms belül legyen, így szinkronban tud maradni a kamerával.

5.3.2 Adatfolyamok és adatfeldolgozás

A Kinect szenzorból érkező adatok egészen az adatfeldolgozóig módosítás nélkül jutnak el. Az adatfeldolgozó alakítja át az adatokat olyan adatreprezentációvá, amit a többi komponens az adatfeldolgozó interfészein keresztül el tud érni.



5.3. ábra Adatfeldolgozás lépései

A feldolgozás lépései

1. Kinect szenzor adatot generál (kamera frame kész)
2. Keretrendszer előfeldolgozza az adatokat, majd eseményt generál (új adat frame)
3. Az adatfeldolgozó az esemény hatására elkezdi a háttérben transzformálni az adatokat, majd eseményt generál (adatok készen)
4. Az adatfelhasználó komponensek (megjelenítő, gesztusfelismerő, következtető rendszer) az esemény hatására elkezdik a beépített algoritmusaik futtatását, majd szükség szerint eseményeket generálnak, amiket a vezérlő és megjelenítő lekezel.

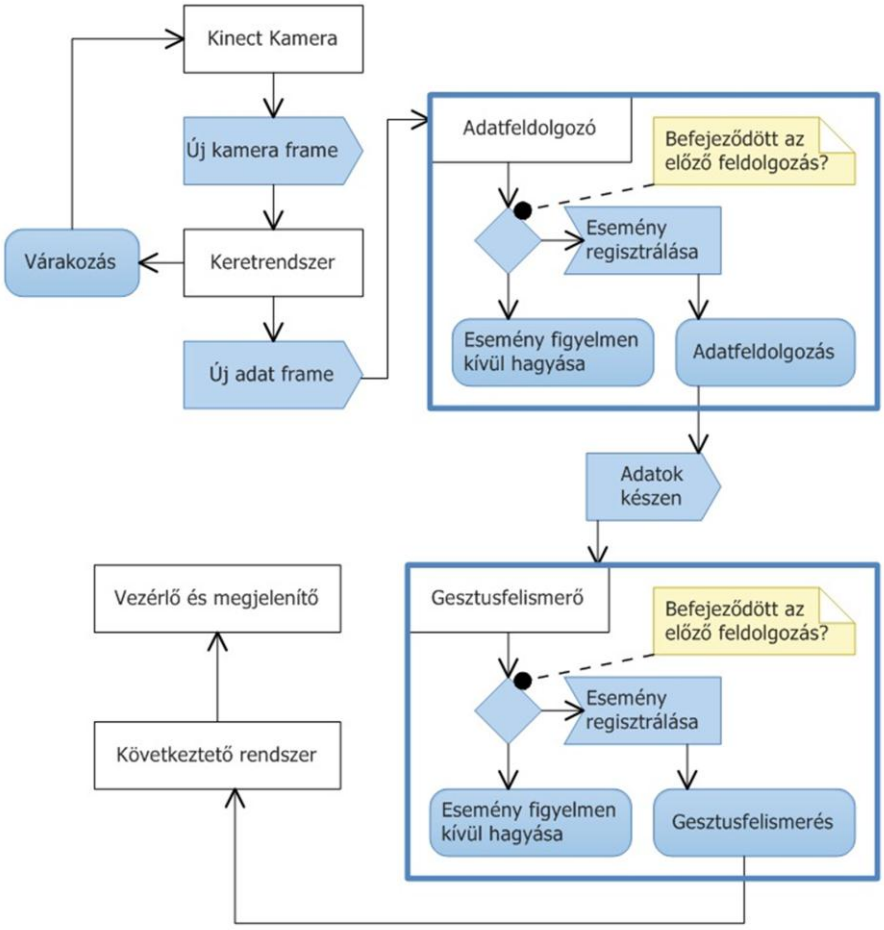
A feldolgozási szekvencia az 5.3 ábrán látható. A komponensek közti átmenetek az események és adatok terjedésének irányát jelzik. Az ábrán az az eset is látható, amikor az adatfeldolgozó komponens túl sokáig végzi a transzformációt, ami miatt az egyik kamera frame-et eldobja a rendszer.

A gesztusfelismerő a valós idejű adatokat elemezve, és az adattárból kiolvasott felismerendő gesztusok alapján egy mintaillesztést végez. A felismert gesztusokat jelzi a következtető rendszernek, és a vezérlőnek. Habár az adatfeldolgozó a számításigényes feladatok nagy részét elvégzi, így nehezen fordulhat elő adattorlódás (ezért dob el indokolt esetben kamera frame-et), de a gesztusfelismerő algoritmus futása az új adat érkezéséig be kell, hogy fejeződjön. Ha ez mégsem történne meg, akkor a gesztusfelismerő komponens képezné a rendszer feldolgozási szekvenciájában a szűk keresztmetszetet, és a már áttranszformált adat frame-eket kellene eldobni. Ez a gesztusfelismerés szempontjából nem előnyös, hiszen (ahogy az 4. fejezetben ismertetem) minél több frame marad ki, annál nehezebben ismeri fel a betanított gesztusokat. Ezért arra kell törekedni, hogy a gesztusfelismerő és következtető rendszer algoritmusai a lehető legoptimálisabban fussanak, ugyanakkor itt is érvényesül az a szabály, hogy a feldolgozás lassulása esetén (az adattorlódást elkerülve) az adat frame-eket el kell dobni.

5.3.3 Szálkezelés és szinkronizáció

A szoftvernek egyszerre kell valósidejű működést (adatfeldolgozást) biztosítania és számításigényes feladatokat végeznie, ezért minden esetben pontosan kell ismerni a feldolgozás állapotát, és ez alapján szinkronizálni a szálakat. A programban a legegyszerűbb

szál szinkronizációs megoldást alkalmaztam, aminek lényege, hogy a feldolgozási szekvenciában minden komponens a saját algoritmusának futásáért felel. Mivel minden számításigényes feladatot az adott komponens külön háttérszálon futtat, ezért a szinkronizáció kimerül abban, hogy az adott komponens a saját *scope*-jában (saját algoritmusait figyelve csak) ellenőrzi az előző ciklus futásának állapotát, és annak befejezéséig nem indít el új ciklust.



5.4. ábra Szálak szinkronizációja

A rendszer alapvetően a „pull” szerű adatfolyamot valósítja meg, tehát az új adatokat egy komponens mindig magában tárolja, majd eseményekkel jelzi azok frissülését. Azon komponensek, akiknek szükségük lehet az adott adatra, az adatszolgáltató komponens interfészén keresztül kérik le a nekik szükséges adatokat. A szinkronizáció tehát úgy valósul meg, hogy ha a frissülést jelző esemény kiváltásakor egy komponens még nem áll készen a feldolgozásra, akkor figyelmen kívül hagyja (ezzel gyakorlatilag eldobva az adott adat frame-et). A szál szinkronizáció vázlatos működése az 5.4 ábrán látható. A keretrendszer aszinkron módon várakozik az új kamera frame-re. A kamera frame készenlétét jelző esemény

beérkezésekor a keretrendszer jelzi az új adat frame elkészültét az adatfeldolgozónak (és minden olyan komponensnek, amelyik közvetlenül értesülni akar erről az eseményről). Az adatfeldolgozó megvizsgálja saját állapotát. Amennyiben nem fejeződött be a feldolgozás, úgy eldobja az eseményt, máskülönben regisztrálja és megkezdi a feldolgozási ciklust. Ugyan ezt az analógiát követve dolgozza fel az adatokat a többi komponens is.

5.4 Komponenseket megvalósító osztályok és a felhasználói leírás

A teljes szoftver, bőségesen kommentezve, megtalálható a diplomamunka DVD mellékletén. A szoftverkomponenseket megvalósító osztályok, és a forrásfájlok leírását az 5. függelék tartalmazza. A szoftver első (1.0-ás) verziója az elképzelt végfelhasználói szoftver funkcióin felül azokat az eszközöket és beállítási lehetőségeket is tartalmazza, amik a fejlesztéshez és teszteléshez szükségesek. A fő megjelenítő komponens (igény esetén) a kameraképet is megjeleníti, de egy végfelhasználói szoftverben, a magánszféra tiszteletben tartása érdekében, ez a funkció teljesen elrejthető. A Kinect AAL szoftver tervezésekor célom volt az ilyen kényelmi és testre szabási lehetőséget demonstráló funkciók beépítése a végső szoftverbe, így azok a főfelületen keresztül elérhetők. A szoftver felhasználói leírása külön fájlként megtalálható a DVD-n (Kinect_AAL_Szoftver_Leiras.pdf), és a diplomamunka nyomtatott mellékleteként.

Harmadik féltől származó forráskódok

A Kinect AAL szoftver több, harmadik féltől származó forráskód részletet, vagy szerelvényt (assembly) használ. Minden felhasznált kód az ingyenes GNU GPL licenc alatt szabadon felhasználható, vagy a szerző engedélyével vettem át. A forrásfájlokban minden helyen jelöltem az adott kódok eredetét. A felhasznált forráskódok és szerelvények:

- AForge.NET (fuzzy következtető rendszer) [45]
- DynamicDataDisplay (idősor adatok megjelenítése) [46]
- Patzold Media3D (3D megjelenítés segédeszközök) [47]
- Absolute Orientation (kalibrációs algoritmus) [37]
- cMatrixLib (mátrixműveletek implementálása) [48]

6.Kinect AAL rendszer tesztelése

A tesztesetek összeállítása előtt speciálisan elesés detektálással, gesztus és ADL aktivitás felismeréssel foglalkozó publikációkat [49,26,50,51,28,27] elemezve az derült ki, hogy az elesés detektálásra és gesztusfelismerésre nem létezik általánosan elfogadott teszt szabvány. Elesés detektálás és gesztus felismerés esetében a tesztek általában néhány tipikus eset definiálásából és sokszori elvégzésből állnak. ADL aktivitás felismerés tesztelésénél általában hosszú ideig (napok, hetek) figyelik meg az alanyokat, és az általuk visszajelzett információk alapján ellenőrzik a rendszer képességeit. Mivel a Kinect AAL rendszer még nincsen éles helyzetben tesztelhető stádiumban, ezért az elesés detektáláshoz, és gesztus és ADL aktivitás felismeréshez is előre definiált esetekkel fogom elvégezni a teszteket.

	Alany 1	Alany 2
Kor	24	21
Magasság	188cm	168cm
Tömeg	85kg	57kg
Testalkat	normál	vékony

6.1 táblázat Kinect AAL rendszerteszteket végző alanyok adatai

Egy fuzzy rendszer esetében, amennyiben a fuzzy változók és a változókat összekapcsoló szabályok jól modellezik a valóságot, a rendszer működésének tulajdonságait legnagyobb mértékben a szabályokban használt halmazok értéktartományai határozzák meg. Ezért két lépésben végeztem el a tesztelést: először a rendszer finomhangolását végeztem el (paraméterek beállítása), hogy az egyértelmű szituációkra biztosan jól reagáljon, és ideális helyzetekben az elvárásoknak megfelelően, kellően precízen működjön a rendszer. Ehhez elsősorban a részletes felületet használtam, és futásidejű elemzéseket végeztem. A rendszer paramétereit ezután már nem állítottam, és azok a diplomamunkában rögzített értékeken maradtak. A második lépésben végeztem el az éles teszteket, melyeket a fejezet további részében ismertetek.

A felhasználói szoftvernek két külön felülete van, melyen az elesést (és általánosabban a vészhelyzeteket), illetve az ADL aktivitásokat nyomon lehet követni. Az egyiken a mérő rendszer számszerű adatait és következtető rendszer kimenetét figyelhetjük meg, míg a

másikon (ez a fő felület és visszajelző), csak a legfontosabb eseményeket. A részletes felület elsősorban a fejlesztés és teszteléshez szükséges, míg a főfelület képezi azt a nézetet, amit mindennapos használat esetén a felhasználó és gondozó látna.

A teszteket két alany végezte el, akik (tesztek szempontjából fontos) adatait a 6.1 táblázat tartalmazza. A két alany megválasztásában elsősorban a magasság játszott szerepet, mivel a Kinect keretrendszer ez alapján határozza meg a tömegközéppont helyét, ami az elesés detektálásnál fontos tényező. A teszteket a 2. fejezetben is bemutatott BME MIT AAL laborban, és két, valós lakókörnyezet szobáiban végeztem el. Az eredményeket a Kinect AAL szoftver főfelülete alapján rögzítettem (felismerés/riasztás), elemzésüket (miért nem ismerte fel, vagy értékelte rosszul az eseményt) pedig a részletesebb idősorok alapján végeztem. Ezek alapján határoztam meg (ahol lehet) a TRR (*True Recognition Rate*, szenzitivitás) értéket és a specificitást. A fejezet elején említett forrásokban a specificitás helyett általában az FPR (*False Positive Rate*) értéket szokták használni, ami a hamis pozitív felismerések arányát adja meg, de ez a tesztek értékelésénél zavaró lehet, ugyanis ekkor a kisebb érték a jobb. A TRR értéket azokra a tesztekre számítottam, amelyeknél fel kell ismernie a rendszernek az adott eseményt (esés, gesztus… stb.), míg a specificitást (később: Spec.) azokra a tesztekre, amelyeknél nem kéne a rendszernek eseményt jeleznie.

- TRR (szenzitivitás) számítása:

$$\frac{TP}{TP + FN}$$

  azaz Felismert / Felismert + Nem felismert.

- Specificitást számítása:

$$\frac{TN}{TN + FP}$$

  azaz Helyesen nem felismert / Helyesen nem felismert + Helytelenül felismert

Mindkét érték esetében a nagyobb számít jobb teljesítménynek.

6.1 Elesés detektálás tesztelése

Egy elesés detektáló rendszer esetében fontos, hogy a rendszer képes legyen megkülönböztetni a valós esési szituációkat az egyéb esetektől. Az elvárt működés, hogy a valós esési szituációk esetében mindig, egyéb esetekben egyáltalán ne riasszon a rendszer. A [50] forrásban leírt tesztelések mintájára a következő eseteket definiáltam:

- valós esési szituációk:
    - esés tetszőleges irányba (előre, hátra, oldalra zuhanás)
    - ájulás (lassú összeesés)
    - elesés kapaszkodással (esés közben az alany ágyba, székbe kapaszkodhat, lassítva a zuhanást)
    - esés bútor mögé (80-90%-ban kitakart test)
- egyéb (nem esési) szituációk
    - sétálás, mozgás a szobában
    - leguggolás cipőt kötni, vagy lehajolás egy tárgyért
    - lehuppanás az ágyra, vagy székbe (gyors leülés)
    - leülés, lefekvés a földre (lassú, kontrolált)

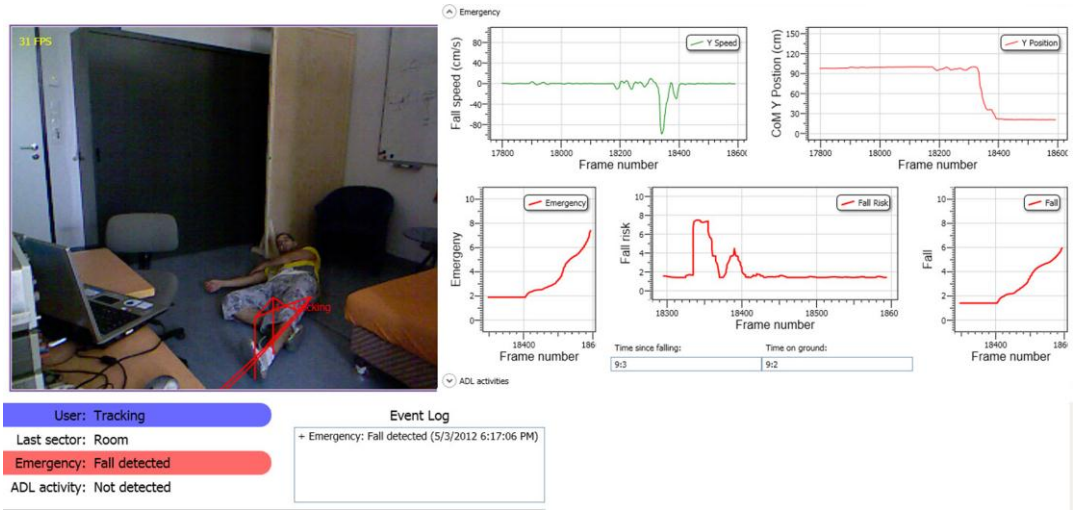
Minden esetet 25-ször hajtott végre mindkét alany, álló helyzetből indulva, egymástól függetlenül, ez által minden szituáció összesen 50-szer lett elvégezve. Az elesés teszteket a BME MIT AAL laborban (2. tesztszoba) végeztem el, a tesztszobában nem használtam semmilyen tompító szivacsot, az alanyok szükség esetén kezükkel tompították az eséseket (az esések jellege ettől függetlenül a valóságot tükrözte).

	Alany 1	Alany 2
Esés tetszőleges irányba	25/0 (100%)	25/0 (100%)
Ájulás (lassú összeesés)	24/1 (96%)	23/2 (92%)
Esés kapaszkodással, félig a székre/ágyra	23/2 (92%)	25/0 (100%)
Esés bútor mögé	22/3 (88%)	20/5 (80%)
Összesítve	TRR: 94%	TRR: 93%
	TRR: 93.5%	

6.2 táblázat Elesés detektálás teszteredményei valós esési szituációkra

A tesztelési eredményeket a 6.2 táblázatban rögzítettem. A cellákban lévő három szám közül az első jelöli a szituációk helyes felismerésének számát (riasztás történt, True Positive), a második a nem felismert esetek számát (False Negative), és a harmadik a szenzitivitást. Látható, hogy a tiszta esési helyzeteket 100%-osan felismerte a rendszer mindkét alany esetében. Egy ilyen esési helyzet látható a 6.1 ábrán, és további ábrák a melléklet DVD-n, a 4. függelékben leírt mappákban találhatók. Az ábrán jól kivehető a zuhanás pillanata (zöld grafikon), a tömegközéppont határozott süllyedése (jobb felső grafikon), és az elesés

kockázatának ugrásszerű növekedése, ami a vészhelyzetet mérő időzítőket indítja el (középen). Az idő lefolyásával pedig az esés és a vészhelyzet értékei monoton növekednek, míg el nem érik a kritikus értéket.



6.1. ábra Egyértelmű esési szituáció képe és adatai

Az ájulások észlelésénél a rendszer az első alany esetében egyszer azért tévedett, mert a felsőteste túl magasan marad (ülő pozícióban), és a tömegközéppont a vészhelyzetet jelölő határ fölött maradt. A második alany két esetben lassan ereszkedett térdre, amit a rendszer leülésnek érzékelt. Bútor mögé esésnél, ha az alanya legalább 10%-ban még látható maradt, akkor a rendszer képes volt tovább követni, és az elesést detektálni.

	Alany1	Alany2
Séta / Mozgás	25/0 (100%)	25/0 (100%)
Leguggolás cipőt kötni, vagy felvenni valamit	25/0 (100%)	24/1 (96%)
Lehuppanás az ágyra, fotelbe	25/0 (100%)	25/0 (100%)
Leülés a földre	21/4 (84%)	8/17 (40%)
Összesítve	Spec.: 96%	Spec.: 82%
	Spec.: 89%	

6.3 táblázat Elesés detektálás teszteredményei egyéb szituációkra

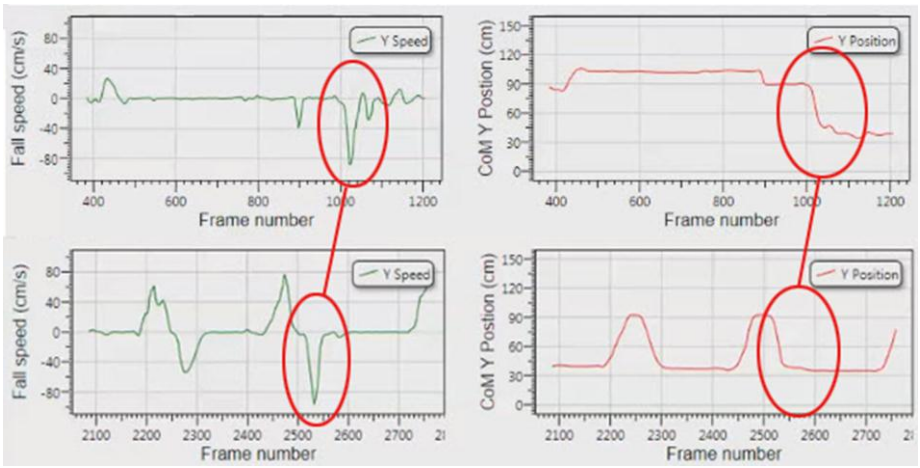
A nem esési szituációk eredményeit a 6.3 táblázat tartalmazza, ahol a cellákban lévő első szám jelzi, hogy a rendszer hány esetben nem jelzett elesést, a második a téves elesés riasztások számát. Látszódik, hogy a földre ülés a második alany esetében 17 esetben esésnek

lett felismerve. A részletes visszajelző képernyő alapján ennek elsődleges oka, hogy az alany alacsony termete és mozgékonysága miatt gyorsan ült le a földre, közel ugyan olyan karakterisztikájú mozgással, mint egy ájulásnál, ahogy ez a 6.2 ábrán is látható.

Az első alany tömegközéppontja eleve magasabban maradt (testalkatából adódóan), illetve a leülés mozdulatsora minden esetben sokkal lassabb volt. Az ágyra vagy fotelbe huppanáskor habár a mozgás mindkét alany esetében általában gyors volt, a tömegközéppont mindig a kritikus érték fölött maradt.

Tesztek értékelése

Az eredmények alapján kijelenthető, hogy az elesés detektálási eljárás algoritmusa helyesnek bizonyult, de további átgondolást igényel a nem felismert esetek miatt.
A [49] megoldás szenzitivitása 91%, a [26] munkában bemutatott megoldás habár nem valós időben dolgozik, de teljesítménye 98.7%, míg a [50] megoldása közel 100%-os teljesítményű, de csak ideális eséseket vizsgáltak, amik a Kinect AAL rendszer esetében is 100%-osak. A [52] munkában közölt megoldásban a TRR 78%-os, a specificitás meglepően alacsony: 69%. Az előbb felsorolt megoldásokkal összehasonlítva a Kinect elesés detektáló rendszer kiemelkedően jól teljesít, még a nehezített körülmények között is. A tényleges esési



6.2. ábra Ájulási esés (felül) és gyors leülés a földre (alul) mozgás karakterisztikája

szituációkban a legtöbb nem felismert esetet a kitakart helyzetben való esés eredményezte. A Kinect AAL szoftver nagy részben a keretrendszerből érkező követési algoritmus eredményére támaszkodva határozza meg az alany pozícióját és sebességét. Amikor a keretrendszer elveszti a követett alanyt, ezt a szoftver úgy értelmezi, hogy az alany elhagyta a szobát, így leállítja az adatok gyűjtését, így a következtető rendszer is leáll. Ezt a későbbiekben a következőképpen kellene módosítani:

- A fizikai kontextus alapján meghatározni, hogy az alany elhagyhatta-e egyáltalán a szobát. (Kijárat közelében volt?)
- Az alany legutóbbi pozíciójából és sebességéből kikövetkeztetett elesés kockázatát adott ideig meg kell jegyezni, és összevetni a fizikai modellel.
- Szoftver szinten a következtető rendszer működését függetleníteni kell az adatfeldolgozó működésétől, lehetővé téve annak futását olyan esetekben is, amikor a Kinect szenzor nem szolgáltat valós időben adatot.

A téves riasztások aránya nagyon magas, aminek elsődleges oka a leülési mozdulatsor hibás kiértékelése. A rendszer jelenleg a tömegközéppont és sebesség, illetve temporális (időbeli) információ alapján végzi az elesés detektálását. Ezek alapján viszont az ájulás karakterisztikája sok esetben megegyezik egy gyors földre üléssel. Ugyanakkor valós helyzetekben, főleg idősebb embereknél, nem valószínű, hogy az alanyok sűrűn, és gyors tempóban fognak a földre leülni. A diplomamunkában rögzített, és a tesztelésnél is használt paraméterek úgy lettek tehát meghatározva, hogy a rendszer a tényleges esési szituációkra mindig helyesen reagáljon (a szélsőségesen kitakart helyzeteket nem számítva 97.5% a felismerés teljesítménye), és a természetes körülmények között előforduló, de kétes helyzeteket (pl. normál sebességű földre ülés) is jól kezelje. A szoftver könnyen paraméterezhető, ezáltal könnyen kialakítható egy olyan rendszer, ami sokkal kritikusabban értékeli az esési szituációkat, és csak a nagy sebességű eséseket ismeri fel. Ez olyan esetekben lehet használható, ahol fiatalabb ember élnek, és elképzelhető, hogy a gyerekek is sokat mozognak a föld közelében.

A tesztelést demonstráló videofelvételek elérési helye a 4. függelékben található. Ezek valós tesztelési szituációkat demonstrálnak, de nem a fenti tesztek elvégzésekor készültek.

6.2 Gesztusfelismerés tesztelése

6.2.1 Gesztusfelismerő DTW algoritmus paraméterezése

Először a gesztusfelismerő tesztelését önmagában, a következtető rendszer használata nélkül, a gesztusbetanító segédeszközzel végeztem. Ennek célja megtalálni azokat a paramétereket, melyekkel a DTW algoritmus kellően pontosan képes diszkriminálni a

különböző gesztusokat. Tehát ezzel még nem a végső rendszert teszteljük, hanem teszteseteken keresztül a rendszerparaméterek optimális értékeit állítjuk be.

Gesztusfelismerés paraméterei

- Adatok mintavételezésének gyakorisága
- Puffer méret
- Minimális gesztus hossz
- Gesztus felismerés határszintje (threshold)
- Maximális hiba (maxError)
- Maximális DTW útvonal meredekség (maxSlope)

Az adatok mintavételezésének gyakoriságát a felismerendő gesztusok jellegéhez, és a feldolgozandó adatmennyiséghez kell beállítani. A mintavételezett adatok kerülnek bele a pufferbe, így a puffer méretét is ennek megfelelően kell állítani. Minél hosszabb gesztusokat akarunk felismerni, annál nagyobb puffer kell. Ugyanakkor, ha a mintavételezést ritkábban végezzük, például csak minden 2. adat frame-et rögzítünk, akkor fele akkora puffer is elég. A DTW algoritmus futási ideje nagyban függ a gesztusok számától és a puffer hosszától, hiszen minden gesztust megpróbál a puffer minden lehetséges hosszára illeszteni. Ezért célszerű a mintavételezést ritkábban elvégezni, és rövidebb pufferrel dolgozni. Egy-egy gesztus maximális hossza 3-4 másodperc, és a rendszer ~30FPS-sel generálja az új adatokat, ezért legalább 5 másodpercnyi (150 hosszúságú) puffere lenne szükség. Ugyan ennek az időtartamnak a megfigyelésére, minden második adatot eldobva, csak 75 méretű puffer kell. A teszteléshez ezért az alábbi kiindulási értéket használom:

- Puffer méret: 75 frame
- Mintavételezés: minden 2. frame
- Maximális megfigyelt időtartam: ~ 5mp

A minimális gesztushossz meghatározása azért fontos, mert túl rövid gesztusokat a DTW algoritmus (a szétnyújtás révén) könnyen ráillesztene már egy kicsit is hasonló adatsorra. Ez persze a maximális meredekség kritériumát állítva korlátozható. Ezért a minimális gesztushossz kezdetben ~1mp legyen, ami (ugyan azt a mintavételezési frekvenciát használva, mint a puffer feltöltésénél) 15 adat frame.

A maximális DTW hibát (ami valójában a gesztusok „távolsága” az aktuális adatsortól, és mértékegysége mm) egyszerűen tapasztalati módon határoztam meg. Mivel a DTW algoritmussal átlaghibát számoltatok (teljes hiba elosztva a csuklópontok számával és a

gesztus hosszával), ezért ennek maximális értéke könnyen becsülhető pár megfigyelés alapján is. Három tetszőleges gesztussal betanítottam a rendszert, majd teljesen véletlenszerű mozgás közben figyeltem az átlaghiba változását. A mérések azt mutatták, hogy 200-250 között változott a hiba. Amikor valamelyik gesztushoz hasonló mozdulatsort végeztem, akkor 100-200 közé csökkent. Ezek alapján a maximális távolságot (hibát) 200-ra állítottam, és ezt a tesztelés folyamán sem módosítottam.

A gesztusfelismerés határszintje azt a minimális *confidence* értéket jelenti, ami fölött egy gesztust „bekövetkezettnek" tekint a felismerő. Ez nagyban függ a maximális hibától, hiszen az alapján normáljuk az értékeket. A maximális hiba meghatározásánál használt módszerhez hasonlóan itt is három gesztus alapján határoztam meg a kezdeti 0.7 küszöbértéket.

A maximális meredekséget célszerű kicsire venni (1-5), mivel a gesztusok hosszával és a meredekség kritériummal arányosan növekszik a hiba is. Minél nagyobb a maximális meredekség, annál jobban elcsúsztathatja a gesztust a DTW algoritmus. Kezdeti értéke 5, de a tesztelés során ez változtatható.

6.2.2 Gesztusok felismerése

A fenti paramétereket felhasználva végeztem el a gesztus felismerési teszteket, egyelőre még mindig a kontextus (fizikai modell) felhasználása nélkül. A hasonló gesztusfelismerő rendszerek mintájára [28,27] a tesztelés itt is a gesztusok betanításából, majd sokszori elvégzéséből áll. Tehát az alany először rögzíti saját gesztusait, majd többször ismételve, egymás után, tetszőleges kontextusban elvégzi őket újra. A tesztekhez szándékosan két olyan gesztust választottam, melyek az AAL alkalmazások szempontjából fontosak lehetnek (lásd 3.4-es szakasz):

- ivás: pohár felemelésétől az ivás kezdetéig (4mp, csak jobb kéz követése)
- öltözködés: póló felvétele vagy levétele fejen keresztül áthúzva (4mp mindkét kéz követése)

Az elesés detektáláshoz hasonlóan itt is fontos a hamis pozitív esetek kiszűrése, ezért az alany nem csak a kijelölt két gesztust végzi el, hanem ahhoz hasonló, más gesztusokat is, és ezek alapján kerülnek kiszámításra a TRR és specificitás értékek. Az egyéb (nem felismerendő) mozdulatsorok:

- hajsimítás
- áll vakarás

- nyújtózkodás
- karok összefüggéstelen mozgatása

Első tesztsorozat

A fenti lista minden mozdulatát a 20-szor hajtotta végre az első alany a 2. tesztszobában, az eredményeket pedig a Kinect AAL szoftver gesztus menedzser kimenete alapján rögzítettem, melyeket a 6.4 táblázat tartalmaz. A cellákban lévő első szám jelzi a helyesen felismert gesztusokat (vagy nem felismert egyéb mozdulatot), míg a második szám a rendszer hibás reakcióinak számát (gesztus nem felismerése, vagy egyéb mozdulat gesztusként való értelmezése). Az eredmények azt mutatják, hogy az ivás gesztust 90%-ban fel tudta ismerni, de nagyon sok hamis pozitív felismerés is történt (az egyéb mozdulatok 67.5%-át ismerte fel ivás gesztusnak). Habár az összefüggéstelen mozgások közben csak 10%-ban tévedett a rendszer, az öltözködés mozdulatsorát nem volt képes felismerni.

Mozdulatsor	Helyes/ Hibás felismerés
Ivás	18/2 (TRR 90%)
Öltözködés	3/17 (TRR 15%)
Hajsimítás	8/12 (Spec. 40%)
Áll vakarás	5/15 (Spec. 25%)
Nyújtózkodás	19/1 (Spec. 95%)
Összefüggéstelen gesztusok	18/2 (Spec. 90%)
Összesítve	TRR: 52.5% Spec.: 62.5%

6.4 táblázat Gesztusfelismerő 1. teszteredményei

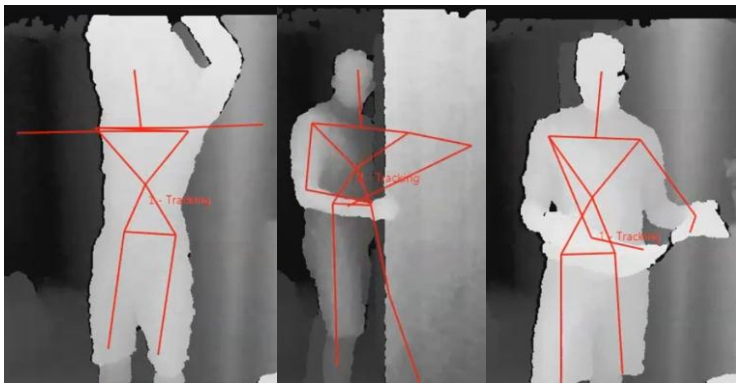
Az első tesztsorozat konklúziója, hogy a fenti paraméterekkel a rendszer gyakorlatilag képtelen volt egyértelműen diszkriminálni az ivás gesztust egyéb, hasonló mozdulatoktól. Ez valószínűleg azért van, mert a DTW könnyen azonosítja a kéz felemelésének akármilyen mozdulatát az ivás gesztussal, amiből az következik, hogy szigorítani kellene a kritériumokat, mind a meredekség kritérium, mind a felismerési küszöb szempontjából.

Továbbá egyértelműen kiderült, hogy az öltözködés jellegű gesztust a rendszer nem fogja tudni csak a szkeleton alapján felismerni, ugyanis a ruhadarabok megzavarják a felismerő rendszert, és a karokat a fejen áthúzás pillanatában nem tudja követni, és véletlenszerű pozícióba helyezi. Habár a 2. fejezetben bemutatott előzetes teszteknél már

kiderült, hogy a tárgyak használata zavarja a szkeleton illesztést (tárgy jellegétől függően kisebb nagyobb mértékben), de az öltözködés gesztus áthatóbb tesztelésénél vált nyilvánvalóvá, hogy a szkeleton alapján ilyen gesztusok felismerésére a rendszer nem képes. Hogy megbizonyosodjak a fent állítottakról további olyan teszteket végeztem, melyek mind olyan tárgyhasználatot igényeltek, ami befolyásolhatja a szkeletont:

- újság lapozgatása
- ruhák vagy ágynemű kirázása
- szekrény kinyitása

Mindegyiknél tapasztalható volt olyan ugrás, vagy vibráció a szkeletonban, ami lehetetlenné teszi a DTW (és bármilyen közvetlenül szkeleton adatokra alapozott) gesztusfelismerő algoritmus működését. A tesztelést bemutató videó a 4. függelékben megadott mappában található, és a tipikus esetek a 6.3 ábrán láthatók.



6.3. ábra Szkeleton pontatlansága gesztusfelismerés közben.
Az aktivitások balról jobbra: öltözés, szekrény kinyitás, újság olvasás

A fenti okok miatt az öltözés és hasonló tárgyhasználatot igénylő gesztusok felismerését nem teszteltem tovább, mivel a szkeleton alapján ezzel a rendszerrel önmagában ez a feladat nem megoldható.

Második tesztsorozat (paraméter módosítás)

A második tesztsorozatban - módosított paraméterek mellett - az előző tesztsorozat gesztusait végeztem el, az öltözködés gesztust kihagyva. Csak a meredekség kritériumot módosítottam 5-ről 3-ra, majd 3-ról 1-re. Az ivás gesztust nem tanítottam újra, így elvégezhető az eredmények összehasonlítása az előző teszttel. A tesztsorozat kimenetelét a 6.5 táblázatban rögzítettem.

Mozdulatsor	Meredekség kritérium: 3	Meredekség kritérium: 1
Ivás	15/5 (TRR 75%)	17/3 (TRR 85%)
Hajsimítás	10/10 (Spec. 50%)	14/6 (Spec. 70%)
Áll vakarás	13/7 (Spec. 65%)	15/5 (Spec. 75%)
Összefüggéstelen gesztusok	19/1 (Spec. 95%)	20/0 (Spec. 100%)
Összesítve	TRR: 75% Spec.: 70%	TRR: 85% Spec.: 81,6%

6.5 táblázat Gesztusfelismerő 2. teszteredményei

3-as meredekség kritérium használatakor az első tesztsorozathoz képest az ivás gesztus felismerésének TRR értéke 90%-ról 75%-ra romlott, ugyanakkor az iváshoz kapcsolható specificitás érték az első teszt 51.6%-hoz képest 70%-ra javult. Ez várható volt, hiszen így a két idősort sokkal nehezebben tudja megfeleltetni egymásnak, és a tényleges ivás gesztusok elvégzése közötti kisebb időbeli különbségeket is észreveszi a rendszer. Az egyes meredekség kritérium esetében meglepő módon az ivás gesztus felismerése valamivel javult (85%-ra), viszont a hamis pozitív esetek szám majdnem a felére csökkent, ami jól tükrözi a meredekség kritérium csökkentésének hatását.

Az általános felismerési küszöb állításától sokkal jobb eredmény nem várható, hiszen a két tesztsorozat alapján az derült ki, hogy a DTW gesztusfelismerő képtelen kis mértékben eltérő mozdulatsorok között különbséget tenni, és a küszöb emelésével a TRR növekedne, de a specificitás azonban csökkenne. A mintavételezés gyakoriságának növelése és puffer méret változtatása sem eredményezett észrevehető különbséget, és a mintavételezés ritkítása csak rontaná a felismerés precizitását.

Harmadik tesztsorozat (vezérlő gesztusok vizsgálata)

A harmadik tesztsorozatot csak kiegészítőként végeztem el, hogy felmérjük, hogy a gesztusfelismerő használható-e AAL alkalmazásokban intelligens ház vezérlésére, vagy célzott gesztusok felismerésére. A teszt annyiban különbözik az előbbiektől, hogy most az alany tudatában van annak, hogy mely gesztust akarja, vagy kell elvégeznie. Három, teljesen eltérő gesztust tanítottam be, és vizsgáltam azok felismerésének teljesítményét. A DTW algoritmus maximális meredekség paramétert 1-ra állítottam, a többi paramétert az első tesztnél használt értékeken hagytam. A felismerendő vezérlő gesztusok:

Mozdulat	Helyes/ Hibás felismerés
Integetés (Wave)	19/1 (TRR 95%)
Vízszintes húzás (Slide)	20/0 (TRR 100%)
Előre nyúlás (Push)	19/1 (95%)
Összefüggéstelen gesztusok	20/0 (100%)
Összesítve	TRR: 96.6% Spec.:100%

6.6 táblázat Gesztusfelismerő teszteredményei vezérlési gesztusokra

1. integetés: jelzés a rendszernek, gondozónak kapcsolatfelvétel céljából (4mp, csak jobb kéz követése)
2. kar vízszintes elhúzása: virtuális csúszka állítása (3mp, csak jobb vagy bal kéz követése)
3. előre nyúlás: gomb megnyomásának imitálása (3mp, csak jobb vagy bal kéz követése)

Az eredmények a 6.6 táblázatban láthatók, és a 4. függelékben jelölt videóban valós időben demonstrálom a gesztusok betanításának és tesztelésének módját. A teszteredmények alapján az a következtetés vonható le, hogy a DTW gesztusfelismerő a célzott gesztusoknál nagyon jól működik, a felismerő szinte minden esetben képes volt az adott vezérlési gesztust felismerni. Az összefüggéstelenül végzett gesztusokat egyetlen esetben sem ismerte fel tévesen a rendszer.

6.2.3 Tesztek összefoglaló értékelése

Összesítve a tesztek alapján az derült ki, hogy a rendszer képes felismerni, és diszkriminálni a gesztusokat, de hasonló mozdulatsorok esetében könnyen megtéveszthető. A felismerő nagymértékben a szkeletonra hagyatkozik, így bármilyen kis zavar, vagy pontatlanság a szkeleton érzékelésében lehetetlenné teszi a gesztusok felismerését (mint például azokban az esetekben, amikor a tesztalany úgy fordult, hogy a gesztust végző keze(i) nem látszódtak). A tesztelés folyamán két, előre nem felmérhető jelenség is tanúbizonyságot nyert. Az egyik, hogy a gesztusok felismerése a betanított gesztusok hosszával arányosan javult. A TRR érték nem változott számottevően (kis mértékben csökkent), de a specificitás

viszont nagy mértékben növekedett. Ebből következik, hogy hosszabb (3-4mp) gesztusok használata javasolt. A második jelenség, hogy a gesztusok számával a rendszer sebessége, és ezáltal a felismerés sebessége is nagy mértékben romlott. Habbár ez a DTW algoritmus alapján várható volt, a rendszer teljesítménye egy további gesztus betanításával hirtelen romlik le. Hat darab, 3 másodperces gesztus esetében már érezhető lassulás történt, ami a felismerés teljesítményére is hatással volt, ugyanis a gesztusfelismerő által „eldobott" adatok miatt a tényleges idősor adatok torzultak. Ilyenkor a mozdulatokat lassabban kellet elvégezni, így a rendszer képes volt felismerni őket.

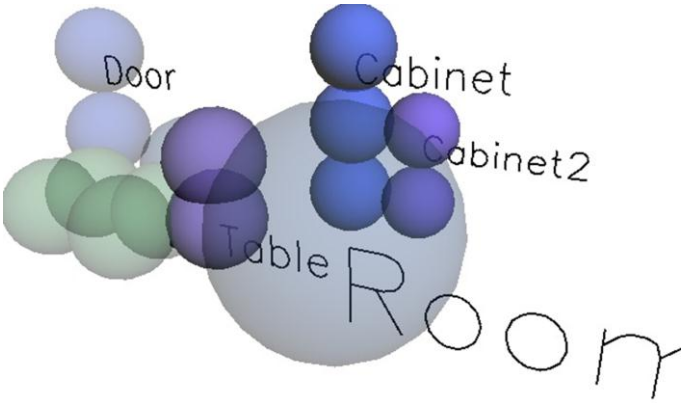
Konklúzióként levonható az a következtetés, hogy a szkeletonra alapozott gesztusfelismerés DTW algoritmussal csak ideális esetekben használható, tehát ha biztosítható, hogy a gesztus felismeréséhez szükséges végtagok biztosan tisztán látszódnak, ilyen esetekben viszont tökéletesen működik. Az algoritmus teljesítménye nagyban függ a gesztusok és a megfigyelt időszegmens hosszától (puffer méret), de ezek csökkentése viszont a gesztus felismerhetőségén ront. A harmadik tesztsorozatban a gesztusfelismerő olyan alternatív alkalmazási lehetőségét mutattam be, ami eredetileg nem része a tervezett Kinect AAL rendszernek, de célszerű lenne a Kinect-et ilyen feladatok megoldására is használni. A kitekintésben ezeket kihasználó rendszerbővítési javaslatokat is adok.

6.3 ADL aktivitás felismerés tesztelése

Az előző pontban letesztelt gesztusfelismerés képezi a gesztus alapú ADL aktivitások felismerésének alapját. A már elvégzett tesztek alapján a következő megszorításokkal végeztem el a további tesztsorozatokat:

- Csak olyan gesztusokat és ADL aktivitásokat végeztem el, amiknél a szkeleton felismerhető maradt.
- Csak azokat a gesztusokat használtam fel az ADL aktivitásokhoz, melyek kellő bizonyossággal felismerhetők alapesetben, tehát az öltözés, vagy más tárgyhasználat gesztusait kizártam.

Az ADL aktivitások közé tartozik a térrészek közötti mozgás felismerése, és az eszközhasználat (szekrény, asztal) felismerése is. A tesztelés elve megegyezik az előbbiekkel, tehát itt is valós helyzeteket szimulálva teszteltem a Kinect AAL rendszert az 1. tesztszobában, melynek fizikai modellje (kontextus) a 6.4 ábrán látható.



6.4. ábra 1. tesztszoba fizikai modellje a Kinect szoftver 3D megjelenítéssel

A szoba elemei:

- objektumok:
  - asztal (Table)
  - szekrény (Cabinet)
  - kis szekrény (Cabinet2)
- térrészek (szektorok):
  - ajtó (Door, kékes színnel)
  - szoba (Room, szürke színnel)
  - ágy (Bed, zöld színnel)

Szobában való mozgás (szektorok közötti váltás)

Az első tesztsorozat célja ellenőrizni, hogy a rendszer helyesen pozícionálja-e be a követett alanyt a fizikai modellbe és ismeri-e fel az aktuális szektort. Mivel itt nehezen lehet esemény alapú teszteket elvégezni, ezért az alany rögzített ideig (300 mp) tetszőlegesen mozgott a szobában (akár a szobából kilépve), miközben én a rendszer főfelületén visszajelzett szektorváltásokat figyeltem, és viszonyítottam a valós világbeli pozíciójához (ez a *Ground Truth* a szubjektív megítélésem alapján). A rendszer fejlesztése és első tesztelése közben azt tapasztaltam, hogy a szektorváltások közötti hibás jelzések elsődleges forrása a kamera véges látótávolsága. Amikor az alany belépett az ajtón, és a kamera megpróbálta lekövetni, az volt tapasztalható, hogy gyors (1-2mp belüli) váltás történt a szektorok között,

majd miután az alany megállt, vagy közelebb mozgott a kamerához, akkor a szektor meghatározása stabilan megtörtént. Ezért egy időbeli szűrést vezettem be a rendszerbe, mely a szektorváltások közötti eltelt időt minimalizálja. A végső tesztek közben a minimális váltási időt 2 másodpercre állítottam, mivel ez egy racionális időtartam a szektorok közötti mozgásra.

A végső teszteredményeket a 6.7 táblázatban rögzítettem, ahol minden fontos szektorváltást feltüntettem, a cellákban lévő szám pedig a nem felismert vagy hibás váltásokat jelöli. Az alany összesen 66 felismerendő szektor váltást végzett el az 5 perces időtartam alatt, melyekből 8 esetben tévedett a rendszer. A visszajelzőn követhető

Váltás típusa	Hibás váltások száma
Ajtón való kilépés a szobából	0
Kamera látóteréből való kilépés	0
Belépés a szobába	3
Ágyba fekvés	2
Ágyból felkelés	3

6.7 táblázat Szektorok közötti mozgás teszteredményei

információk alapján a rendszer hibás szektor meghatározásának elsődleges oka, hogy ha az alany a kamera látóterébe való belépéskor nagyon gyorsan mozgott, akkor a rendszernek nem volt ideje felismerni, és a szükséges adatokat időben kiszámolni. Ilyen esetekben a rendszer hibásan határozta meg a szektort, és a szektorok közötti váltást pedig a minimalizált váltási idő miatt nem tudta követni. Ugyanakkor ezek szélsőséges eseteknek tekinthetők.

A rendszer TRR értékét kétféle képen számíthatjuk. A váltásokat tekintve 87% a felismerési teljesítmény, de ez nem feltétlenül tükrözi a tényleges működést. Sokkal relevánsabb értéket kapunk, ha a teljes megfigyelési idő, és a hibásan megállapított szektorokban eltöltött idő hányadosát vesszük. A videofelvételek alapján kiszámítható, hogy a teljes 5 perces időtartamból körülbelül 5 másodpercig jelzett a rendszer hibás szektort, ami körülbelül 98%-os teljesítményt jelent. Előfordult olyan kritikus eset, mikor az alany nem követett állapotból hirtelen állt fel és ment ki a helyiségből és a rendszer hibásan azt jelezte, hogy az alany az ágyon maradt, pedig a valójában elhagyta a szobát. Az ilyen eseteket a Kinect kamera véges látótávolsága miatt egykamerás rendszerrel nehéz kezelni, de több kamera használatával megoldható lenne.

Objektumokkal való interakció vizsgálata

A tesztsorozat célja meghatározni, hogy a rendszer helyesen következteti-e ki az objektumhasználat szándékát az alany mozgása és nézési iránya alapján. Az alany minden azonosított objektumot 30-szor próbált meg „használni” (asztalhoz lehajolni, poharat felvenni; szekrény ajtaját kinyitni; kis szekrényen tárgyat mozgatni). A teszteredményeket a 6.8 táblázatban rögzítettem, a cellákban lévő számok pedig a felismert / nem felismert objektumhasználatok számát jelölik.

Objektum	Felismert / Nem felismert
Asztal	27/3 (90%)
Szekrény	25/5 (83,3%)
Kis szekrény	28/2 (93,3%)
Összesítve	TRR: 92.2%

6.8 táblázat Objektumhasználat teszteredményei

Az eredmények azt tükrözik, hogy a következtető rendszer jól működött ehhez a feladathoz, és az esetek több mint 90%-ban sikeresen felismerte az adott objektumhasználatot. A teljes tesztelés közben hamis pozitív eseteket gyakorlatilag nem produkált a rendszer, mivel nagyon szűkre lettek állítva a paraméterek, azaz csak nagyon kis sebességnél, és majdnem teljesen ráfordult felsőtesttel ismeri fel a rendszer az aktivitást. Emiatt viszont az asztalnál három esetben nem is ismerte fel a rendszer az objektum használati szándékot, mivel az alany nem hajolt le eléggé. Néhány esetben az alany szándékosan háttal közel állt az objektumokhoz, ezzel tesztelve a nézési irány vektorát használó távolságszámítás működését. A rendszer szinte minden esetben képes volt lekezelni ezeket az eseteket. Ugyanakkor a tesztelés közben észrevehető volt a szkeleton felismerésének gyenge pontja: a keretrendszer nehezen tudja felismerni azt az esetet, mikor egy ember sokáig, háttal a szenzornak mozog, ekkor ugyanis a keretrendszer úgy értelmezi, hogy az alany szemben áll a szenzorral. Sajnos ez a szkeleton felismerési algoritmus hibája, így nehezen korrigálható. A szekrénynél 5 esetben nem volt képes a rendszer felismerni az aktivitást, elsősorban a gesztusfelismerésnél is felmerülő probléma, az alany más mozgó objektumokkal való összeolvadása miatt (szekrényajtó megmozdítása).

Gesztus alapú ADL aktivitás vizsgálata

A gesztusalapú ADL aktivitás felismerés célja, hogy egy olyan speciális ADL aktivitást definiáljunk a rendszer számára, mely azt figyeli, hogy egy adott objektum (asztal) közelében történik-e valamilyen beállított gesztus (ivás). A teszteléshez az ivás gesztust az asztal objektumhoz rendeltem a fizikai modell leíró fájlban. Az ADL aktivitást 50-szer hajtotta végre az első alany az 1. tesztszobában, több irányból, és guggolásból is, ezzel is ellenőrizve a rendszer megbízhatóságát. A tesztesetek közül csak 41-et ismert fel a rendszer, ami 82%-os TRR teljesítményt jelent. Egy ilyen esemény felismerése nagyon nehéz feladat, hiszen a gesztus végrehajtása közben sem szabad az alanynak számottevően eltávolodnia az adott objektumtól, hiszen a fuzzy következtető ezek alapján próbálja azonosítani az aktivitást. Az 1. tesztszobában az asztal távolabb van az ideális (2m-re szembefordulva a kamerával) gesztusfelismerési pozíciótól, de az ivás mozdulatsort végző kéz végig látható maradt. A Kinect AAL szoftver részletes adatmegjelenítője alapján azt a következtetést vontam el, hogy a nem felismert aktivitások esetében a szkeleton illesztésének zaja, vagy a gesztusfelismerő hibája miatt maga a gesztus felismerése nem történt meg (tehát a gesztusfelismerés lett az ADL aktivitás felismerés szűk keresztmetszete).

6.3.1 Tesztek összefoglaló értékelése

Az alacsonyszintű ADL aktivitásokat nagyon pontosan volt képes felismerni a rendszer. A szektorok közötti mozgás és az objektumhasználat információi jó alapot képezhetnek akár más szenzorokat használó ADL alkalmazásoknak, mivel megbízhatóan képes ezeket megállapítani. Az objektum használathoz tervezett fuzzy következtető képes volt megkülönböztetni azokat a komplex eseteket is, mikor egy ember az objektum közelében forgott, és csak azokban az esetekben jelzett, mikor az alany éppen az objektum felé nézet. A gesztus alapú ADL aktivitások felismerése viszonylag pontos volt, habár a gesztusfelismerés tesztelésénél megállapított hiányosságok és nehézségek itt is akadályozták a következtető rendszer működését.

Az ADL aktivitások tesztelését demonstráló video anyagok a mellékelt DVD-n találhatók, és a 4. függelékben olvasható a magyarázatuk.

6.4 Szoftver teljesítmény értékelése

A szoftver alap esetben, csak a Kinect kamera adatokat lekérdezve is igen nagy számítási igényű, emellett viszont további számításigényes feladatokat (pl. DTW algoritmus) is végez. Egy valós idejű, beágyazott rendszernél fontos számításba venni a szükséges erőforrásokat. Megállapítható, hogy a szűk keresztmetszetet a Kinect AAL-nél a processzor teljesítménye jelenti, ugyanis a szoftver maximálisan csak körülbelül 300MB rendszermemóriát foglal, és a 3D grafikus megjelenítés is csak a tesztelés közbeni vizualizációra volt használva, így a végső szoftverben kikapcsolható. Két konfiguráción végeztem teljesítmény teszteket: AMD X64 2 magos 2.2 GHz CPU, és Intel i7 4 magos 2,6GHz Intel CPU. Az első konfiguráción a keretrendszerek alapprogramjai, melyek pusztán a Kinect képét jelenítik meg, átlagban 70%-on terhelték a processzort, míg a másodikon ez csak 15%. A Kinect AAL szoftver, minden számítást elvégezve (3D megjelenítést kikapcsolva), az első konfiguráción majdnem 100%-on, a másodikon 20%-on terheli a rendszert. Ebből az a következtetés vonható le, hogy az algoritmusokat sikerült kellően optimálisan megvalósítani, de a Kinect szenzort használó szoftverek általánosan nagy számításigényűek, így komolyabb hardver szükséges a futtatásokhoz.

7. Összefoglalás és kitekintés

A diplomamunkában a Kinect szenzor képességeire alapozva megterveztem és implementáltam egy általános AAL rendszert, amely képes az elesés detektálás problémáját megoldani, ideálisabb körülmények között általános gesztusfelismerésre, és előre rögzített ADL aktivitások felismerésére. Kijelenthető, hogy a Kinect alkalmas és jól használható szenzorikus eszköz AAL alkalmazásokhoz.

Munkám elején rámutattam az AAL témakörben használt technológiák jelenlegi állására, amely indokolttá tette egy potenciálisan hasznos szenzorikus eszköz, a Kinect használatának kivizsgálását AAL alkalmazásokhoz. A tervezési fázisban számos meglévő elesés detektáló, gesztusfelismerő és általánosabb AAL rendszerben használt ötletre alapozva, de azokat finomítva és tovább gondolva alakítottam ki a Kinect szenzor képességeit kihasználó Kinect AAL rendszert.

Az elesés detektálás már számos AAL alkalmazás által megoldott probléma, de az általam kidolgozott, sebességet, pozíciót és időbeli paramétereket is használó, valós idejű rendszer a konkurens rendszerekkel összehasonlítva pontosabban, szélsőséges helyzeteket is jól kezelve oldja meg a problémát, emellett további fejlesztésekre (teljesen kitakart helyzetbe való esés) is lehetőséget ad. Az általános gesztusfelismerést egy meglévő, DTW algoritmust használó gesztusfelismerőt használva alakítottam ki, de számos módosítással: elforgatás és elmozdulás függetlenné tettem a felismerést, ezáltal alkalmassá téve AAL alkalmazásokban való használathoz, továbbá az algoritmushoz új paramétereket definiáltam, és a kimenetét is úgy alakítottam ki, hogy a fuzzy következtető rendszerbe bekötve, ADL aktivitások felismerésére is használható legyen. A tesztelés eredményei alapján a teljesen általános felismerés ezzel a módszerrel csak ideális helyzetekben oldható meg, tehát amikor a gesztust végző végtagok teljesen a Kinect szenzor látóterében vannak. AAL környezetben vezérelési gesztusok felismerésére (vezérelhető intelligens ház) a rendszer jól használható lenne.

Az ADL aktivitások felismerésére a Kinect képességeit kihasználó többszintű modellt mutattam be. A szenzor háromdimenziós térrekonstrukciójának alapötletét felhasználva egy fizikai kontextus modellt terveztem meg, amely a teljes megfigyelés alapú AAL alkalmazások alapját képezheti. A beépített kalibráló algoritmus segítségével a rendszer a valós fizikai teret egy virtuális térképként képes kezelni, és olyan alapszintű információkat biztosítani az AAL alkalmazás számára, mint például az alany aktuális pozíciója, mozgási

sebessége és nézési iránya a valós, háromdimenziós térben. A Kinect AAL alkalmazás ezeket felhasználva képes kikövetkeztetni a tárgyhasználatot igénylő és gesztus alapú ADL aktivitásokat.

A rendszer és az azt megvalósító szoftver úgy lett megtervezve, hogy további szenzorokkal könnyen bővíthető legyen, ezáltal egy Kinect alapú platformot biztosítva AAL alkalmazások fejlesztéséhez. A moduláris felépítés lehetővé teszi további szenzorok és funkciók egyszerű integrálását a meglévő rendszer független komponenseinek minimálist módosításával.

Az elkészült Kinect AAL rendszerrel a kitűzött célokat megvalósítottam, de számos bővítési és fejlesztési lehetőség adott. A Kinect mikrofonját a rendszer első verziójában nem használtam fel, de a szenzor és a keretrendszerek speciális képességeinek (ultra érzékeny hangfelvétel, hangforrás irányának becslése, beszédfelismerés) felhasználhatóságát AAL alkalmazásokban, és a Kinect AAL rendszerben külön tanulmányként érdemes lenne elvégezni. A mikrofonból kinyert adatokkal lehetőség nyílna a hangvezérlésre és a vészhelyzet észlelés segélyhívásokkal való kiegészítésére. A kialakított következtető rendszer a lehető legegyszerűbben, csak a célfeladatok megoldására lett kialakítva, teszteléssel meghatározott paraméterekkel és szabályokkal. Érdemes lenne más következető rendszereket is letesztelni a Kinect szenzor adataival, vagy a fuzzy következtető paramétereit és szabályait példák alapján tanítani. Ezzel egy sokkal általánosabb következtetőt lehetne létrehozni, mely könnyebben adaptálható egy adott környezethez.

A szoftvert fejlesztése kihívást jelentett, mivel egy valós idejű, de komplex algoritmusokkal operáló rendszert kellett kialakítani. Az egyes komponensek tesztelését és finomhangolását olyan funkciók beépítésével lehetne könnyíteni, mint például a DTW algoritmus paramétereinek valós idejű állíthatósága (puffer méretek, küszöbértékek), és a fuzzy változók értékhatárainak és szabályok dinamikus módosítása. A fizika modell megadása jelenleg manuálisan végezhető csak el, ami automatikus 3D felület kereső algoritmusokkal, és a felhasználói interfész fejlesztésével sokkal könnyebben lenne elvégezhető. Habár a kalibráló algoritmus jól működött, és a teljes kalibrációs eljárás egyik tesztszobában sem tartott tovább 15 percnél, a kalibráló pontok kimérése, megadása és a kalibrálás elvégezése nagyon körülményes feladat. Elméletben megoldható lenne egy olyan eljárás kifejlesztése, mely a Kinect mélységkép alapján és felhasználói beavatkozás segítségével képes lenne a megfigyelt szoba nagyobb kiterjedésű felületeit megkeresni (falak, mennyezet, padló), és ez alapján egy abszolút koordinátarendszert meghatározni.

A Kinect AAL által használt gesztusfelismerő algoritmus az ideális esetek jól kezelte, de érdemes lenne egy teljesen más megközelítést használó gesztusfelismerést is kipróbálni a rendszerrel. A gesztusfelismerést a keretrendszer szkeleton adataira alapoztam, de érdemes lenne a szenzor mélységadatait önmagukban is felhasználni, hiszen ezekből rengeteg további információ is leszűrhető lenne (pl. kézfej pozíciója és csavarodása az alkarhoz képest). A Kinect szenzor által szolgáltatott adatok pontossága és felbontása a távolsággal romlik, de a Microsoft tervezi a Kinect 2 szenzor kifejlesztését, mely az előzetes információk alapján jelentősen jobb felbontással fog rendelkezni, és képes lesz beépített módon felismerni az ujjak állását is. Továbbá sorra jelennek meg az alternatív hardverek (Panasonic D-Imager, Xtion Pro Live) melyek eltérő képességekkel rendelkeznek, de a Kinect alternatívájaként érdemes lenne megvizsgálni őket. Kiemelném, hogy a Kinect AAL szoftver tervezéskor a 3D szenzort használó keretrendszer kiválasztásában ezért is esett a végső választás a nyílt OpenNI-re, mivel így ez könnyebben megtehető. A fejlesztés folyamán a keretrendszerek is változtak, és mostanra a Microsoft elérhetővé tette a Kinect SDK hivatalos, 1.0-és verzióját, mely képességeit tekintve sokban nem különbözik a béta verziótól, viszont képes már 40cm távolságú mélységadatokat szolgáltatni, és több kamerát képes akár szinkronban kezelni. Ettől függetlenül ez nem jelent előnyt a Kinect AAL alkalmazás szempontjából, így tesztelése fölösleges.

A szoftverben implementált algoritmusok nem terhelik feleslegesen a rendszert, és további számottevő számítási igény nélkül is szinkronban tudnak működni a kamerával. Ez alól kivétel a DTW algoritmus, mely már 6-8 gesztus esetében is számottevően lassult, és az adatvesztés miatt a gesztusokat sem volt képes felismerni. Erre megoldást jelenthet, ha az ADL aktivitásokhoz felhasznált többszintű modell segítségével a felismerendő gesztusokat előszűrnénk, és a DTW algoritmust csak a kontextus alapján lehetséges gesztusokra futtatnánk le. Így két kategóriára kéne bontani a gesztusokat: tetszőleges helyen elvégezhető gesztusok, melyeket mindig vizsgálni kell, és kontextus függő gesztusok, amiket csak adott esetekben kell vizsgálni. Ezzel nagyban csökkenthető a rendszer számításigénye, és a gesztusfelismerés is pontosabban működhet.

1. Függelék - Kinect technikai specifikáció [17]

Fő tulajdonságok

- RGB kamera (640x480; 30FPS)
- Mélységérzékelő (IR; 640x480; 30FPS)
- Hangérzékelő (mikrofon)
- Mozgató motor (függőleges dőlésszög változtatáshoz)

Látószög

- Vízszintes FOV (Field of View): 57°
- Függőleges FOV (Field of View): 43°
- Motor függőleges szögváltoztatása: 27°
- Mélységérzékelő érzékelési tartománya (ajánlott): 1.2m–3.5m
- Mélységérzékelő érzékelési tartománya: 0.8m–6 m

Adatfolyam

- Mélység-térkép: 640x480 felbontás,11-bit (2,048 jelszint), 30 FPS
- Színes kamera kép: 640x480 felbontás, 32-bit, 30 FPS
- Audió: 16-bit, 16 kHz mintavételezés

Szkeleton (csontváz; ember) érzékelés

- mozgó alakok érzékelése
- aktív alakok érzékelése (szkeletonnal)
- aktív alakonként 20 „joint point” (ízületi- vagy végpont) érzékelése

Audio érzékelő

- 4 mikrofon
- 16 bit, 16 kHz mintavételezés
- Szoftverfüggő funkciók: hangfelvétel, visszhang kiszűrés, beszédfelismerés, hangforrás irányának meghatározására

2. Függelék – Szoftver keretrendszerek tesztelése

1. Kalibrálás, távolságok tesztelése
   a) OpenNI: 0,5m –en belül is képes volt csak a felsőtest alapján azonosítani az emberi alakot. 5m-en kívül a felismert emberi alak követését elveszti, de a közeledő alakot rögtön azonosítja.
   b) Kinect SDK: Csak felsőtest alapján nehezen tudta azonosítani az emberi alakot, de különben u.a. mint az OpenNI.
2. Ember felismerés
   a) OpenNI: Érzékenység: 5-10cm-es, lassú mozgástól már felismeri az emberi alakot 2,5m távolságban.
   b) Kinect SDK: u.a. mint az OpenNI.
3. Ember követése karosszékbe ülés közben
   a) OpenNI: Leülés után a két kontúr összeolvadt (lásd 6-os teszt és 2.7 ábra 3. képe). Felállás után a fotelt sokszor megtartotta külön mozgó entitásként (nem háttér), és megpróbált rá szkeletont illeszteni.
   b) Kinect SDK: Hasonlóan az OpenNI-hez, itt is hibásan észlelete az emberi alakot, ha a székben ülve, azzal együtt mozgott, de felállás után szeparálta, és eldobta a székhez tartozó részeket.
4. Ember követése lefekvés közben
   a) OpenNI: Fekvő pozícióban az emberi alak követése pontatlanabb, de a felismerő algoritmus nem veszítette el az emberi alakot.
      Nem felismert helyzetből kiindulva a fekvő embert egyáltalán nem ismerte fel, sem mozgás (forgolódás) közben, sem felüléskor.Felálláskor észlelete csak az emberi alakot.
   b) Kinect SDK: u.a. mint az OpenNI.
5. Ember követése félig kitakart helyzetben
   a) OpenNI: A felismert emberi alakot kitakart helyzetben is tudta követni. Nem látható pozícióból félig kitakart ember alakot egyáltalán nem ismerte fel.
   b) Kinect SDK: u.a. mint az OpenNI.
6. Ember követése más mozgó tárgy mellett (karosszék)

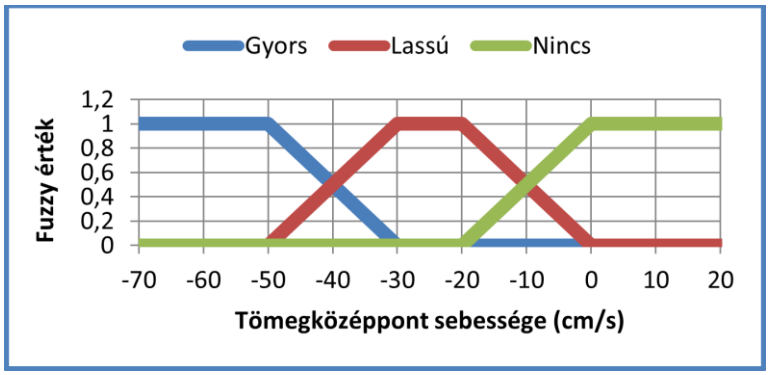
a) OpenNI: A tárgy mozgatása közben a két kontúr egybeolvadt, és ha közel maradt, akkor a felismerő nem is választotta szét. 0,5m távolságban már szeparálni tudta a két entitást, de a széket is megtartotta mozgó entitásként.
b) Kinect SDK: Hasonlóan működött, mint az OpenNI, de a szeparált széket már nem követte tovább.

7. Szkeleton kalibrálás
   a) OpenNI: Képes automatikusan megtalálni a szkeletont. Kalibráló póz végrehajtása nélkül kb. 1mp alatt illeszti a szkeletont az emberi alakra. Az első pár másodpercben az illesztés pontossága nagyban javul.
   b) Kinect SDK: Automatikusan illeszti a szkeletont, gyakorlatilag az emberi alak felismerésével egy időben (1mp-en belül), akárcsak az OpenNI.
8. Szkeleton félig (vertikálisan) kitakart helyzetben
   a) OpenNI: Félig kitakart helyzetben a szkeleton illesztése nem teljes, de a látható pontok csak kis mértékben ugrálnak. Kitakart helyzetből való teljesen látható pozícióba mozgás közben vissza tudja illeszteni a pontokat.
   Teljesen kitakart helyzetből (mikor elveszti az ember alakot) (hasonlóan a 7-es teszthez) rögtön felismeri és illeszti a szkeletont.
   b) Kinect SDK: A szkeleton illesztése pontatlan, az ízületi pontok ugrálnak.
   Teljesen kitakart helyzetből is képes újra szkeletont illeszteni, de csak a teljes ember alak megjelenésekor.
9. Szkeleton félig (horizontálisan) kitakart helyzetben
   a) OpenNI: Viszonylag pontosan tudja követni a felsőtestet. A lábak látótérbe kerülésekor kalibrálás nélkül tudja követni a teljes testet.
   b) Kinect SDK: Felsőtest követése nagyon pontatlan, az ízületi pontok ugrálnak.
   A lábak látótérbe kerülésekor kalibrálás nélkül tudja követni a teljes testet.
10. Szkeleton követés
    a) OpenNI: Megbízhatóan tudja követni az embert és illeszteni a szkeletont. Kb. 5m távolságon kívül nem tudja követni.
    b) Kinect SDK: A szkeleton követése pontatlanabbnak tűnik, több az ízületi pontok ugrálása, de az 5m-es hatótávolság itt is érvényes.
11. Szkeleton illesztése és követése fekvő helyzetben
    a) OpenNI: Pontatlanul, de pozícióját tekintve tudja követni az ember alakot, és valamilyen módon a szkeletont is illeszteni próbálja (legtöbbször rosszul).
    Felüléskor a felsőtestet már megtalálja, majd felálláskor a teljes szkeletont.

12. Kinect SDK: Egyáltalán nem tudja fekvő, sem fekvésből felülő helyzetben illeszteni a szkeletont. Csak teljes álló pozícióban képes illeszteni és követni.
13. Szkeleton illesztése leülés közben
    a) OpenNI: Leülés közben sokat számít, hogy a lábak látszódnak-e. Ha nem látszódnak, akkor felsőtest csuklópontjainak illesztése pontatlan. Az általános emberfelismerésnél leírt összeolvadás itt is megjelenik, és emiatt a szkeleton is pontatlan. Ha a lábak is látszódnak, akkor pontosabb a felismerés, ritkán illeszti a székhez a szkeleton egyes pontjait (székben forgás, mozgás közben sem).
    b) Kinect SDK: Hasonló tulajdonságokat mutatott, mint az OpenNI.

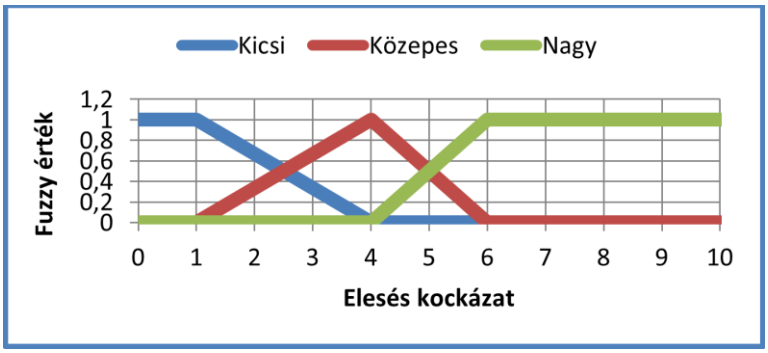
3. Függelék – Elesés detektálás fuzzy változói

Tömegközéppont sebessége (Y tengely mentén), értékkészlet: [-100, 100] cm/s



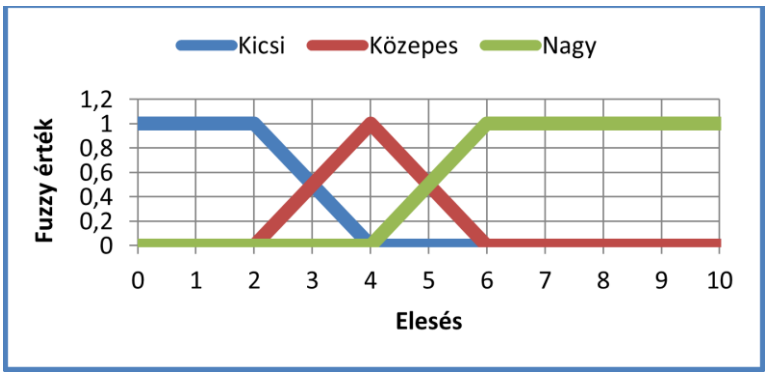
2. ábra Tömegközéppont sebessége fuzzy változó tagsági függvényei

Elesés kockázata, értékkészlet: [0, 10]



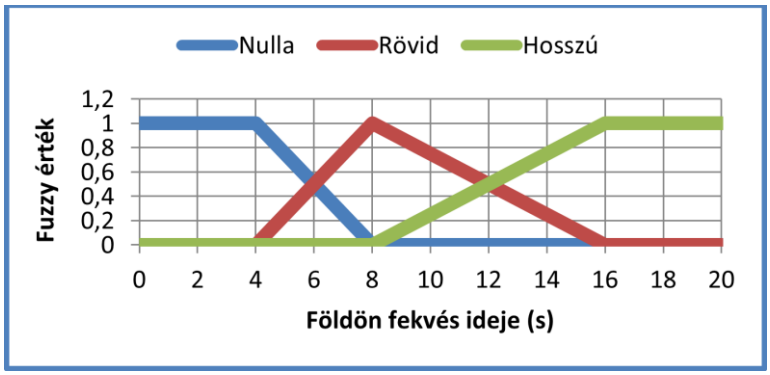
3. ábra Elesés kockázat fuzzy változó tagsági függvényei

Elesés, értékkészlet: [0, 10]



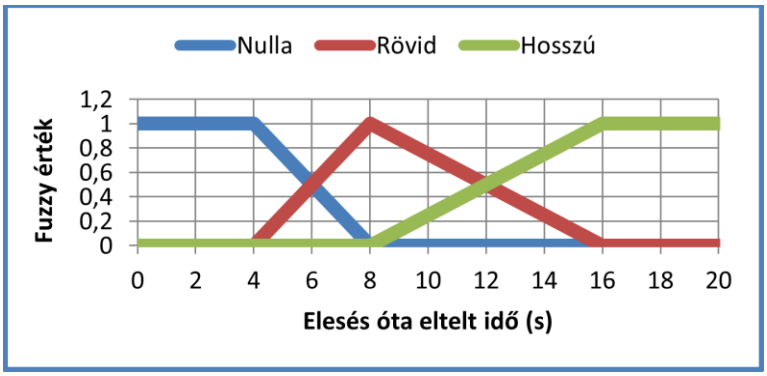
4. ábra Elesés fuzzy változó tagsági függvényei

Földön fekvés ideje, értékkészlet: [0, 20] másodperc



5. ábra Földön fekvés ideje fuzzy változó tagsági függvényei

Elesés óta eltelt idő, értékkészlet: [0, 20] másodperc



6. ábra Elesés óta letelt idő fuzzy változó tagsági függvényei

4. Függelék – Mellékelt DVD tartalma

A mellékelt DVD-n található a szoftver teljes forráskódja, a Visual Studio 2012 project fájlok, a futtatható program, és a Kinect AAL rendszer fejlesztése és tesztelése közben készült demonstrációs videók és képek. A videókon szereplő események nem tükrözik a tesztelések menetét, csak demonstráló jelleggel lettek rögzítve, az alanyok által sokszor szándékosan olyan érdekes szituációkat előidézve, melyek érdekesek lehetnek a későbbi elemzések szempontjából.

Kinect AAL szoftver

- Kinect AAL\
  - release\
    A végső futtatható szoftver fájljait tartalmazza. A szükséges szoftver előkövetelmények (OpenNI, NITE, driver) telepítése után a programot a KinectAAL.exe futtatásával lehet elindítani.
  - source\ KinectAAL_WPF\
    A teljes Visual Studio 2010 projektet és a Kinect AAL szoftver forráskódjait tartalmazó könyvtár.
  - utils\
    Harmadik féltől származó szoftverek és driverek. A Kinect AAL szoftver futtatásához először az *openni-win32-1.5.2.23-dev.msi*, majd *nite-win32-1.5.2.21-dev.msi*, végül *SensorKinect091-Bin-Win32-v5.1.0.25.msi* fájlokat kell futtatni és feltelepíteni a keretrendszert. A feltelepítés után a *KinectXMLs* könyvtárban található XML fájlokat rendre az OpenNI és NITE telepítési könyvtárain belül a *data* könyvárakba kell másolni.

Média anyagok (képes és videók)

- images\
  - Test rooms\
    A tesztszobák képei és térképei.
  - OpenNI test\
    A keretrendszer tesztelése közben elmentett képernyőképek. Az *OpenNI_test06.jpg* és *OpenNI_test07.jpg* demonstrálják az OpenNI követő algoritmusának hibáit. A *OpenNI_test09.jpg* és *OpenNI_test10.jpg* is két

hibásan felismert esetet ábrázol, amikor a rendszer a forgó széket próbálta meg embernek felismerni.

- Gesture recognition test\
  Szkeleton illesztés hibája tárgyhasználatkor, gesztusfelismerésnél.
- Fall test\
  BME MIT AAL laborban végzett elesés detektálás tesztelése. A képek tipikus esési szituációkat és az azokhoz tartozó mérési adatokat mutatják be.
  - Fall_normal: tipikus esési szituációk.
  - Fall_cling: félig székre esés, vagy székbe kapaszkodásos szituációk.
  - Fall_occluded: kitakart pozícióba való esési szituációk
  - Sitting.jpg: leülés mozdulatsora. Látszódik, hogy normál leülésnél a sebesség (zöld grafikon) jóval kisebb, és a tömegközéppont is lassabban, és magasabbra süllyed.
  - Walk.jpg: egyszerű sétálás grafikonjai. A sebességben és tömegközéppont magasságban láthatók kisebb zajok (lépésdinamika), de ez normálisnak tekinthető.
- Early versions\
  Érdekeség képen két képernyőkép a szoftver korai verziójából.

- videos\
  - ADL activity\
    - ADL_drink.avi (http://youtu.be/oc373BUe2Mc):
      ivás ADL aktivitás demonstrációja
    - ADL_object_usage_activity_demo.mp4 (http://youtu.be/c3c7qpWmDQo): objektum használat és ivás ADL aktivitás tesztelése. 1:15-nél látszódik, hogy a szkeleton hogyan torzul egy ajtó kinyitásakor, és rögtön utána a 3D nézeten látszódik, hogy az „előre” irány (kék vektor) is megfordul egy időre, ezzel megtévesztve a felismerő algoritmust.
    - Gesture_drink.avi (http://youtu.be/82ZOa4qS4DY):
      ivás gesztus tesztelése. A rendszer audiovizuálisan (hanggal és kiírással) jelzi vissza a gesztus bekövetkezését.
    - Gesture_recog_itemuse_fail.mp4: a szkeleton torzulását demonstrálja tárgyhasználat közben.

- Gesture_train_test.avi (http://youtu.be/B8DPq8vadYA):
  gesztus menedzser működését demonstráló videó. Valós időben tanítok meg a rendszernek különböző vezérlő gesztusokat, amiket a rendszer utána rögtön, hiba nélkül felismer.
- Sector_change_first_tests.mp4: térrész váltás első tesztelése. Itt még nem lett a váltások között idő minimalizálva, illetve a 3D nézet sincs szinkronban a valós adatokkal.
- Sector_change_test.mp4 (http://youtu.be/SPSZP3TGq-A):
  térrész váltás demonstrációja. 1:00-nél látható, hogyan reagál a rendszer a látótérbe hirtelen beszaladó emberre.

- Fall detection\
  - Fall_detect_faint.mp4 (http://youtu.be/8lDlOBhgNxM):
    tipikus ájulási mozdulatsor és detektálása
  - Fall_detect_instant_falldetect.mp4 (http://youtu.be/uQVvkltvc0g):
    a kamera látóterébe hirtelen belép alak felismerése és az elesés sikeres detektálása
  - Fall_detect_multiple_falling.mp4 (http://youtu.be/sDbShwa_ANw):
    különböző irányú esési szituációk demonstrálása idősor adatokkal
  - Fall_detect_sitting.avi (http://youtu.be/AhH_tqYI5XA):
    normál (lassú) leülési szituáció, és sikeres detektálása
  - Fall_detect_sitting_fail.mp4:
    gyors leülési szituáció hibás detektálása. Összehasonlítva a Fall_detect_faint.mp4 videóval látszódik, hogy mozgás lefolyása nagyon hasonló, és a kis termetű tesztalany miatt a tömegközéppont is nagyon alacsonyan maradt.
  - Fall_detect_sofa.mp4 (http://youtu.be/jKZqnDqGBng):
    félig kapaszkodó pózba való esést (székre, kanapéra) és sikeres detektálását demonstrálja
  - Fall_detect_various_sleeping_falling.mp4
    (http://youtu.be/dFx7MRhb1aQ): normális mozgások (sétálás, lefekvés) közben elesés detektálás. Ez a video demonstrálja egy normális napi rutint végző alany egyszeri elesési szituációját, és annak sikeres felismerését.

- Fall_detect_various_walking_falling.mp4 (http://youtu.be/H4bDVSi8gE4): Előzőhöz hasonlóan, itt is normális sétálás közbeni elesés demonstrálása látható.

- Other\
  - Calibration.avi: kalibráció menetét bemutató gyorsított (12-szeres) felvétele. A teljes kalibráció körülbelül 15 percet vett igénybe. A videó azt szemlélteti, hogy maga a kalibráció egyszerűen elvégezhető, akármilyen helyszínhez pár marker és mérőszalag segítségével.
  - OpenNI_test.mp4: OpenNI keretrendszer tesztelésekor rögzített videó, melyen a szék hibás felismerése is látszódik.
  - Early_test_v0.1.mp4: érdekesség képen a szoftver korai verzióját bemutató videó. A felvételen már a kontextus modell kezdetleges verziója is látszódik, ahogyan a 3D nézeten a szkeleton, a nézési irány vektora, és a sebességvektor dinamikus változása.

A linkkel is jelzett videók a következő URL-en, lejátszási listaként is megnézhetők: http://www.youtube.com/playlist?list=PLF09D65F098F834B8

5. Függelék – Szoftver forrásfájlok leírása

Ez a függelék a DVD-n található Kinect AAL\source\KinectAAL_WPF\KinectAAL_WPF könyvtárban lévő forrásfájlok leírásait tartalmazza. A teljes forráskód bőségesen kommentezett, és az osztályok részletes leírása is a forráskódokban olvasható.

Gyökérkönyvtár

- MainController.xaml , MainController.xaml.cs:
  Ez a program belépési pontja (fő felhasználói felület ablaka), és ez az osztály felel a többi komponens létrehozásáért és összekapcsolásáért. Feladatai: a fő felhasználói felület biztosítása a többi komponens eléréséhez, és az AAL események (ADL aktvitás, elesés, alany követés) megjelenítése. A MainCotroller végzi a fontosabb események logolását (felületen, és fájlba).
- AboutDialog.xaml , AboutDialog.xaml.cs: szoftverinformációs ablak.

data

- RecordedGestures\
  - DefaultGestures.txt: a program indulásakor ezek a gesztusok fognak betöltődni. Ha új gesztusokat rögzítünk, és következő indításkor ezeket szeretnénk használni, akkor azokat mentsük ebbe a fájlba.
- CalibrationData.xml: kalibrációs adatokat leíró XML fájl.
- KinectConfig.xml: OpenNI futásához szükséges Kinect kamerát paraméterező XML fájl.
- SceneModelData.xml: fizikai kontextust leíró XML fájl.

Components

- Calibration\
  - AOAbsoluteOrientation.cs: kalibrációs algoritmust megvalósító osztály. A public *compute* függvény futtatja a tényleges algoritmust.
  - AOFrame.cs: kalibrációs transzformációt tároló osztály. Lehetőséget ad a transzformáció alkalmazására és módosítására.
  - Calibration.cs: Calibráció a Kinect AAL számára reprezentáló osztály. Ez olvassa és írja a kalibrációs XML fájlt, és a fenti két osztályt használva végzi a kalibrációt.

- CalibrationAssistant.xaml, CalibrationAssistant.xaml.cs:
  Egy külön segédeszköz a programon belül, mely felületet biztosít a kalibráló algoritmushoz szükséges referencia pontok rögzítésére, és a kalibrálás elvégzéséhez. A fizikai modell felépítésében is használható a pozíciók meghatározására. Elvégzi a kalibrációs adatok betöltését, ami elengedhetetlen a rendszer működéséhez.
- CalibrationTest.xaml, CalibrationTest.xaml.cs: kalibráció visszaellenőrzését (hibaszámítás) megjelenítő ablak.
- cMatrixLib.cs: mátrixműveleteket megvalósító segédosztály.

- DataProcessor\
  - DataProcessor.cs: Minden adatot, amit a kamera generál az adatfeldolgozó kap meg. Az adatfeldolgozó osztály négy interfésszel rendelkezik, amiken keresztül a többi osztály feladat specifikusan érheti el az adatokat és függvényeket.
    - IDataConverter.cs: Ezen az interfészen keresztül érhető el minden olyan segédfüggvény, ami a Kinect, a kalibrált és a kivetített koordinátarendszer között képes adatokat transzformálni. Funkciói minden olyan esetben használhatók, amikor az adatokat manuálisan transzformálni kell.
  - .
  - IImageData.cs: RGB és mélységkép adatokat tárol, és dedikált függvényeken keresztül biztosítja azok elérését.
  - IMotionData.cs: A követett alany mozgásával kapcsolatos adatokat (pozíció, sebesség, távolságok, irányok) számolja és tárolja, és függvényeket biztosít ezek lekéréséhez. Funkcionalitását elsősorban a következtető rendszer használja fel az elesés detektáláshoz.
  - ISkeletonData.cs: A keretrendszer szkeleton adatait tárolja, és függvényeket és eseményeket biztosít a DTW algoritmus számára az adatok eléréséhez.
- GestureRecognizer\
  - DtwAlgorithm.cs: gesztusfelismerő DTW algoritmus megvalósító osztály. A DTW implementációját a *dtw* függvény tartalmazza.
  - Gesture.cs: gesztust reprezentáló osztály, mely eltárolja a gesztus nevét, és idősor adatait két féle formában: összes adat, és „kimaszkolt” adatok (csak

azon csuklópontok adatait tartalmazza, melyek a gesztusban részt vesznek), amiket így a DTW már elő feldolgozva kap meg.

- GestureManager.xaml, GestureManager.xaml.cs:
  Ez az osztály valósítja meg a gesztus betanító és tesztelő modult. Külön eszközként nyitható meg a főprogramból, és segítségével a meglévő gesztusokat lehet betölteni és módosítani, vagy újakat betanítani, és fájlokba eltárolni. Statikus tagváltozóban tárolja az összes betöltött gesztust, és azok aktuális *confidence* értékeit, így a többi osztály ezen keresztül érheti el azokat. Ez az osztály példányosítja a DtwAlgorithm osztályt, mely a DTW algoritmust futtatja a GestureManager-ben tárolt adat pufferen, és a statikus gesztus-táron.
- GesturePrompt.xaml, GesturePrompt.xaml.cs: új gesztus felvételét megkönnyítő ablak, mely bekéri a gesztus nevét, hosszát és a megfigyelendő szkeleton pontokat.

- InferenceEngine\
  - InferenceEngine.cs: A következtető rendszert megvalósító osztály. A fuzzy rendszert, fuzzy változók definícióit és a szabályokat tárolja. A DataProcessor osztály eseményére feliratkozva minden új frame érkezésekor elvégzi az elesés detektálás és ADL aktivitás felismerés algoritmusát, aminek eredményéről új eseményt generál. Ez az osztály felel minden következtetési folyamat (fuzzy, threshold) elvégzéséért, tehát ez képezi (a gesztusfelismerő után) a Kinect AAL rendszer intelligens részét.
  - SceneModel.cs: fizikai modellt tároló osztály. A modell XML fájlt olvassa be, és ez alapján tárolja el a SceneObject objektumokat, melyeket ezen az osztályon keresztül ér el a szoftver többi része. A SceneModel osztály funkcionalitását elsősorban a következtető rendszer és a megjelenítő használja.
  - SceneObject.cs: egy fizikai objektumot, vagy térrészt reprezentáló osztály. Eltárolja az objektum nevét, pontjait és méreteit, és hozzá rendelt ADL aktivitásokat.
- KinectCamera\
  - KinectCamera.cs, KinectCameraOpenNI.cs:
    A KinectCamera egy absztrakt osztály mely azokat a funkcionalitásokat

deklarálja, amik egy tetszőleges szoftver keretrendszert (OpenNI, KinectSDK) használó program Esetében kötelezően implementálandók. Ezzel biztosítható az, hogy a program többi része csak keretrendszer független funkciókat használjon.
A KinectCameraOpenNI az absztrakt osztály leszármazottja, mely OpenNI keretrendszer specifikus módon valósítja meg az általános kamerafunkciókat. Ez az osztály futtatja a kameraadatokat lekérdező szálat, és generálja az új adatokat, alanyt és szkeleton követést jelző eseményeket, amiket az adatfeldolgozó kap el.

- View\
    - Mesh3D.cs, Bar3D.cs, Ellipse3D.cs, Pyramid3D.cs, Triangle3D.cs: háromdimenziós alakzatok adatait (pontok, méretek, színek) tároló osztályok. A Mesh3D osztályból származik le a többi osztály.
    - Model3D.cs: háromdimenziós megjelenítés modell osztálya, mely több alakzatot reprezentálhat.
    - TextureMapping.cs: különböző textúrázásokat megvalósító osztály, mely a modellek megjelenését határozzák meg a 3D nézeten.
    - MotionViewer.xaml, MotionViewer.xaml.cs: idősor adatok és a következtető rendszer kimenetét megjelenítő ablak (részletes leírása az 5. fejezetben).
    - Scene2D.cs, Scene3D.cs:
      Két külön osztály, mely a fő megjelenítőn, és a segédeszközökön képes a Kinect képét és egyéb adatokat vizualizálni. A 2D nézet az RGB kameraképet, követett alany szkeletonját és további információkat jelenít meg. A 3D nézet jeleníti meg a fizikai virtuális modellt (SceneModel elemei), és a szkeleton háromdimenziós változatát a virtuális térben. Ezen a nézeten nyomon követhetjük az objektumokkal való interakciót, megjeleníti a nézési irányt, és a dinamikusan változó sebességvektort.
    - Trackball.cs: a háromdimenziós ablak nézeti transzformációit számoló osztály.

Ábrajegyzék

Táblázat jegyzék

Irodalomjegyzék

[1] Valuch Tibor, *Magyarország társadalomtörténete a XX század második felében*. Budapest: Osiris, 2001.

[2] Wolfgang L. Zagler, Paul Panek, and Marjo Rauhala, "Ambient Assisted Living Systems - The Conflicts between Technology, Acceptance, Ethics and Privacy," Vienna University of Technology, Vienna, Austria, Szeminárium 2008.

[3] http://www.aal-europe.eu/. (2011, Sep.) Ambient Assisted Living Joint Programme Catalogue of Projects. PDF doumentum.

[4] Reiner Wichert and Birgid Eberhardt, *Ambient Assisted Living: 5. AAL-Kongress 2012 Berlin, Germany, January 24-25, 2012*, Reiner Wichert and Birgid Eberhardt, Eds. Berlin: Springer, 2012.

[5] Marius Pflüger, Julia Kroll, and Barbara Steiner, "Automatic Recognition of Emergencies with the Help of Optical and Acoustic Sensors," in *Ambient Assisted Living: 5. AAL-Kongress 2012 Berlin*. Berlin: Springer, 2012, pp. 29-41.

[6] Diego López-de-Ipiña, Xabier Laiseca, Ander Barbier, and Unai Aguilera. (2012, May) Infrastructural Support for Ambient Assisted Living. [Online]. http://paginaspersonales.deusto.es/dipina/publications/ZAINGUNE-UCAMI08_v7.pdf

[7] Winifred W. Logan, Alison J. Tierney Nancy Roper, *The Roper-Logan-Tierney model of nursing.*: Elsevier Health Sciences, 2000.

[8] Thomas D. Downs,Helen R., Robert C. Grotz Sidney Katz, "Progress in Development of the Index of ADL," *Gerontologist*, vol. 1., Apr. 1970.

[9] (2011) PRIME SENSE LTD PATENTS. [Online]. http://www.faqs.org/patents/assignee/prime-sense-ltd/

[10] PrimeSense. (2010) Range mapping using speckle decorrelation. [Online]. http://www.freepatentsonline.com/7433024.html

[11] PrimeSense. (2012, May) NITE Controls 1.3.1 User Guide. [Online]. https://bitbucket.org/manctl/nite/src/87978e57f6ed/doc/NITE%20Controls%201.3.1%20User%20Guide.pdf

[12] Microsoft. (2012, May) Kinect for Windows Programming Guide. [Online]. http://msdn.microsoft.com/en-us/library/hh855348.aspx

[13] Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake Jamie Shotton. (2011) Real-Time Human Pose Recognition in Parts from a Single Depth Image. [Online]. http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf

[14] (2010) OpenNI Google Group. [Online]. http://groups.google.com/group/openni-dev

[15] OpenNI. (2012, May) OpenNI Programmers Guide. [Online]. http://openni.org/Documentation/ProgrammerGuide.html

[16] PrimeSense. (2012, May) NITE 1.3 Algorithms notes. [Online]. http://pr.cs.cornell.edu/humanactivities/data/NITE.pdf

[17] (2011) Kinect Wikipedia. [Online]. http://en.wikipedia.org/wiki/Kinect

[18] Microsoft. (2011) Kinect Games. [Online]. http://www.xbox.com/hu-HU/kinect/games

[19] Ryan Orendorff. (2011) Esoma Exercise System. [Online]. http://newmed.media.mit.edu/blog/jom/2011/03/17/esoma-exercise-system-cardiac-rehab-using-kinect

[20] Wenqing Dai, Jay Eggert, Jarod T. Giger, James Keller Zhongna Zhou, "A Real-time System for In-home Activity Monitoring of Elders," , 31st Annual International Conference of the IEEE EMBS, 2009.09.

[21] Zhihai He, Derek Anderson, James Keller, Marjorie Skubic Xi Chen, "Adaptive Silouette Extraction and Human Tracking in Complex and Dynamic Environments," *Image Processing, 2006 IEEE International Conference on*, pp. 561 - 564, Oct. 2006.

[22] Zhongna Zhou et al., "Activity Analysis, Summarization, and Visualization for Indoor Human Activity Monitoring," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 18, no. 11, NOVEMBER 2008.

[23] Erik E. Stone and Marjorie Skubic, "Evaluation of an Inexpensive Depth Camera for Passive In-Home Fall Risk Assessment," Pervasive Health Conference, Dublin, Ireland, May 21-23, 2011, pp 71-77 2011.

[24] Robert H. Luke, James M. Keller, Marjorie Skubic Derek Anderson, "Modeling Human Activity From Voxel Person Using Fuzzy Logic," *Fuzzy Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 39 - 49 , Feb. 2009.

[25] Liyanage C De Silva, "Audiovisual Sensing Of Human Movements For Home-Care And Security In A Smart Environment," *International Journal On Smart Sensing and Intelligent Systems*, vol. 1, Mar. 2008.

[26] Caroline Rougier, Edouard Auvinet, Jacqueline Rousseau, Max Mignotte, and Jean Meunier, "Fall Detection from Depth Map Video Sequences," in *Towards Useful Services for Elderly and People with Disabilities*. QC, Berlin: Springer, 2011, pp. 121–128.

[27] S. Rajko, Gang Qian, T. Ingalls, and J. James, "Real-time Gesture Recognition with Minimal Training Requirements and On-line Learning," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, Minneapolis, 2007, pp. 1-8.

[28] Bo Peng, Gang Qian, and S. Rajko, "View-invariant full-body gesture recognition from video," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, Tampa, 2008, pp. 1-5.

[29] François Bremond, "Scene Understanding: perception, multi-sensor fusion, spatio-temporal reasoning and activity recognition," University of Nice, Sophia Antipolis, Doktori disszertáció 2007.

[30] L.A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, June 1965.

[31] L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *Systems, Man and Cybernetics*, pp. 28–44, Jan. 1973.

[32] Stuart Russel and Peter Norvig, *Mesterséges Intelligencia Modern Megközelítésben*, Második, bővített ed. Budapest: Panem, 2005.

[33] Jan Jantzen. (1998, Aug.) Tutorial On Fuzzy Logic. [Online]. http://www.imamu.edu.sa/Scientific_selections/abstracts/Math/Tutorial%20On%20Fuzzy%20Logic.pdf

[34] Khandelwal Piyush and Peter Stone, "A low cost ground truth detection system for RoboCup using the Kinect," Proceedings of the RoboCup International Symposium 2011, Istanbul, Turkey, 2011.

[35] A. Lorusso, R.B. Fisher D.W. Eggert, "Estimating 3-D rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, pp. 272–290, Mar. 1997.

[36] Horn BKP, "Closed-form solution of absolute orientation using unit quaternions," *J Opt Soc Am Ser A*, vol. 4, no. 4, pp. 629–642, 1987.

[37] Ziv Yaniv. (2012, Apr.) High Quality Affordable Health Care for All. [Online]. http://isiswiki.georgetown.edu/zivy/

[38] Iván Gómez-Conde, David Olivieri, Xosé Antón Vila, and Stella Orozco-Ochoa, "Simple Human Gesture Detection and Recognition Using a Feature Vector and a Real-Time

Histogram Based Algorithm," *Journal of Signal and Information Processing*, pp. 279-286, Feb. 2011.

[39] Pálfalvi József, "Mikroszkopikus méretű partikulumok morfológiai paramétereinek mérése," BME-MIT, Budapest, BSc szakdolgozat May 2010.

[40] Rhemyst and Rymix. (2011, June) Kinect DTW. [Online]. http://kinectdtw.codeplex.com/

[41] HandVu. (2012, May) HandVu: Vision-based Hand Gesture Recognition and User Interface. [Online]. http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html

[42] H Sakoe and S Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43- 49, Feb. 1978.

[43] Stan Salvador and Philip Chan, "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space," *Intelligent Data Analysis*, vol. 11, pp. 561-580, Oct. 2007.

[44] Meinard Müller, "Dynamic Time Warping," in *Information Retrieval for Music and Motion*.: Springer, 2007, ch. 4, pp. 69-84.

[45] AForge. (2012, May) AForge.NET Framework. [Online]. http://code.google.com/p/aforge/

[46] Mikhail Brinchuk. (2009, Apr.) WPF DynamicDataDisplay. [Online]. http://dynamicdatadisplay.codeplex.com/

[47] Charles Petzold. (2012, May) 3D Programming for Windows. [Online]. http://www.charlespetzold.com/3D/index.html

[48] Anas S. A. (2012, May) Advanced Matrix Library in C#. NET. [Online]. http://www.codeproject.com/Articles/51470/Advanced-Matrix-Library-in-C-NET

[49] Qiang Li, "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information," in *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, Univ. of Virginia, Charlottesville, VA, USA, 2009, pp. 138 - 143.

[50] Mars Lan et al., "SmartFall: An Automatic Fall Detection System Based on Subsequence Matching for the SmartCane," University of California Los Angeles, USA, Konferencia beszámoló ICST 978-963-9799-41-71, 2009.

[51] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," in *Lecture Notes in Computer Science*, első ed. Berlin: Springer, 2004, pp. 158-175.

[52] J. Spehr, M. Gövercin, S. Winkelbach, E. Steinhagen-Thiessen, and F. M. Wahl, "Visual Fall Detection in Home Environments," *GERONTECHNOLOGY*, vol. 7, no. 2, p. 114, May 2008.

[53] Daniele Babusci. (2012, May) Introducing: Through the Keyhole. [Online]. http://mydeco.files.wordpress.com/2009/10/3d-floor-plan-emma1.jpg

[54] Andrewe1. (2012, May) Kinect With Nightshot. [Online]. http://www.youtube.com/watch?v=nvvQJxgykcU