Gyorsulásmérő szenzoron alapuló valósidejű gesztusfelismerő áramkör megvalósítása karóra méretben

TDK dolgozat
Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Médiainformatikai Tanszék
2011

készítette:
Szucher Krisztián
IV. éves villamosmérnök hallgató

konzulens:
Prekopcsák Zoltán

Tartalom

1. Bevezetés

Manapság egyre nagyobb igény mutatkozik az ember-gép interakciót (Human-Computer Interaction, HCI) minél egyszerűbbé és természetesebbé tévő megoldások iránt. A technológia fejlettségi szintje lehetővé teszi, hogy változatos módokon avatkozzunk be elektromos berendezéseink működésébe. A mobil eszközök térhódítása révén továbbá egyre kevesebb hely maradt a számítógéphez kapcsolódó alapvető periféria, a billentyűzet kiépítéséhez ezért számtalan egyéb megoldás jelent meg. Érintőképernyővel már szinte bárhol találkozhatunk egy mobiltelefon apró képernyőjétől akár egy asztallap méretű eszközig (ilyen például a Microsoft Surface[1]). Hanggal is vezérelhetjük eszközeinket, például az iPhone 4S új hangvezérlési megoldása[2] ezt teszi lehetővé. Lehetőségünk van akár teljesen természetes módon kinyilvánítani akaratunkat gesztusaink segítségével is. Egy gesztus jól jellemezheti szándékunkat, hozzájuk valamely jelentés társítható, ezáltal funkció rendelhető. Számtalan emberi gesztus létezik, ezek közül egy kézmozdulat egy nagyon egyszerű kifejezési mód, hiszen sokféle változatos formát, alakzatot rajzolhatunk a kezünkkel. Elterjedt kar- ill. kézmozdulatfelismerő rendszerek a Nintendo Wii[3] és a Kinect[4]. A különféle szenzorok lehetővé teszik számtalan adat gyűjtését, kézenfekvő tehát az igény, hogy számítógépünk ezen adatokból valamely feldolgozási folyamat révén detektálni tudja, hogy milyen gesztust szerettünk volna kinyilvánítani. A továbbiakban kézmozdulatok felismerésével fogok foglalkozni, gesztus alatt a továbbiakban kézmozdulatokat fogok érteni.

Kézmozdulatok felismerése lehetséges egy kamera képének elemzésével, gyorsulásmérő szenzor használatával, illetve ezen két módszer ötvözésével. Munkám során kizárólag gyorsulásmérő szenzoron alapuló gesztusfelismeréssel foglalkoztam. A videó-alapú megoldásokkal szemben ennek fő előnye, hogy nem igényli kamerák kiépítését, használata ezáltal sokkal kevésbé helyhez kötött. A jelenleg létező gyorsulásmérő szenzoron alapuló megoldások esetén szükségünk van a karra rögzített szenzoron kívül egy a gyorsulásmérő-adatokon való műveletvégzést és elemzést elvégző számítógépre, ezért beszélhetünk kevésbé helyhez kötöttségről, de a mobilitás itt sem teljesül maradéktalanul, A mögöttes feldolgozó egység sajnos ebben az esetben is korlátozza a szabadságunkat, tehát pl. az adatainkon elemzést végző számítógép közelében kell maradnunk. A számítógép persze akár 90% fölötti hatékonysággal tudja a beérkező adatfolyamból a gesztusainkat felismerni [1]. Elterjedt és népszerű, főként kamera-alapú megoldások, mint a korábban említett Wii Remote és a Kinect is hatékonyan képesek gesztusfelismerésre, azonban a mobilitás itt is korlátozott, amennyiben nem a mozgás szabadságára, hanem a helyhez kötöttségre gondolunk (pl. csak abban a szobában működik a rendszer, ahol a kamerák vannak). Célul tűztem ki egy olyan eszköz megvalósítását, amely önálló adatgyűjtésre és feldolgozásra képes, így teljes mobilitást tesz lehetővé. Különböző mobil eszközökben megjelentek a gyorsulásmérők, itt elmondható, hogy az eszköz teljesen önállóan képes adatgyűjtésre és feldolgozásra, de itt nincs szó komplexebb gesztusok felismeréséről. Ilyen pl. az iPhone készülékekben a forgatás érzékelése, mely mindössze a nehézségi gyorsulás irányának értékéből következtet annak pozíciójára.

[1] http://www.microsoft.com/surface/en/us/whatissurface.aspx
[2] http://www.apple.com/iphone/features/siri.html
[3] http://www.nintendo.com/wii/console/features
[4] http://www.xbox.com/hu-HU/Kinect/GetStarted

A mobilitás és a bárhol alkalmazhatóság feltételének leginkább egy „viselhető” eszköz felel meg. Jelenleg nem elérhetőek olyan eszközök, amelyek karóra méretben az adatgyűjtést mellett az adatfeldolgozást is megvalósítanák valós időben, ebből a szemponból a fejlesztésem újdonságnak számít.
Dolgozatom során egy olyan eszköz fejlesztését mutatom be, mely szabályos geometriai formákon alapuló gesztusokat képes felismerni, és mindehhez olyan minimális az erőforrás-igénye, hogy karóra méretben megvalósítható az adatgyűjtő, az adatfeldolgozó és egy vezeték nélküli kommunikációs egység.

1-1. A dolgozat felépítése

A 2. fejezetben áttekintem a Távközlési és Médiainformatikai Tanszékhez köthető, korábbi kutatásokat a gesztusfelismerés témakörében, melyeket referenciaként használtam a munkám során.

A 3. fejezetben egy újszerű megközelítéssel egy analitikus megoldást javaslok bizonyos gesztusok, jellemzően félkör, ferde illetve szögletes vonalakból álló alakzatok felismerésére. Ezen megközelítés előnye, hogy az eszköz nem igényel tanító mintákat. Minták felvételével persze növelhető a felismerés pontossága, ezáltal lehetőség van a felhasználó mozdulataihoz történő minél jobb adaptálódására bizonyos érzékenységi paraméterek tekintetében.

A 4. fejezetben bemutatom egy olyan algoritmus fejlesztését, mely ezen analitikus módszeren alapulva a bejövő gyorsulásmérő-adatokon valós idejű adatfeldolgozást végez. Ábrák és animációk segítségével az algoritmust működés közben is bemutatom idealizált és valós gyorsulásmérő-mintákon is, továbbá az alkalmazott megoldások lehetőségeit és korlátait. Az algoritmus hatékonyságának tesztelését a saját eszközömmel rögzített, továbbá egy előre rögzített adathalmazból [2] kiolvasott mintákon végeztem. Ezen rögzített adathalmazra a továbbiakban gesztus-adatbázisként fogok hivatkozni.

Az 5. fejezetben bemutatom, ahogy az algoritmust egy nagyobb prototípuson való futtatás és folyamatos tesztelés közben továbbfejlesztettem, és végül egy ténylegesen karóra méretű áramkört hoztam létre. Ezen áramkörben a jelfeldolgozást és a vezeték nélküli kommunikációt egy-egy mikrokontroller végzi. A gesztusfelismerés során kritikus tényező a gyorsulásmérő adatok minél gyakoribb mintavételezése, bemutatom a feldolgozás sebességét korlátozó tényezőket, a jelenleg alkalmazott megoldást, majd egy adatfeldolgozás-párhuzamosítási lehetőséget amely a jelenlegi megoldásnál sokkal pontosabb gesztusfelismerést tehet majd a jövőben lehetővé.

Végül, a 6. fejezetben összefoglalom az eddig elért eredményeket, bemutatok néhány ötletet ezen eszköz felhasználására, majd a későbbi kutatáshoz fogalmazok meg célokat.

2. Korábbi kutatások

A gesztusfelismerés jelenlegi eredményeiről egy kiváló összefoglaló munka érhető el *Testi gesztusok számítógépes felismerése* címmel [3], a továbbiakban nem tekintem célomnak ezen eredmények bemutatását.

Gesztusfelismerő algoritmusok nagy számban érhetőek el, azonban összehasonlításuk nem könnyű feladat. Ezen algoritmusok tesztelésére elkészült egy gesztus-adatbázis 2011 januárjában [2]. Ezen adatbázis objektív tesztelést tesz lehetővé a különféle algoritmusok számára, mindehhez egy általános, 23 elemű gesztus-szótár elemeihez tartalmaz mintákat. Az adatbázis elkészítése 30 fő részvételével készült, több gyorsulásmérésre alkalmas eszközzel. Ezen eszközök közül volt egy bluetooth kommunikáción keresztül gyorsulásmérő adatokat kiküldő eszköz[5], egy gyorsulásmérő szenzort tartalmazó mobiltelefon és a Wii kontrollere.

Egy korábbi, 2007-es kutatás eredményeként létrejött egy szoftver, mely segítségével mobil eszközökben található gyorsulásmérő adatait felhasználva nagy hatákonysággal lehet adatbányászati eszközökkel gesztusokat felismerni. Ezen gesztusok felismeréséhez tanító minták megadása szükséges [1]. Az adatgyűjtés mobil környezetben, az adatfeldolgozás számítógépen valósult meg.

Felmerült az ötlet, hogy egy önálló és programozható készüléket lenne érdemes tervezni. Azokkal az eszközökkel, amelyeket a gesztus-adatbázis felvétele közben alkalmaztak, ez nem lehetséges. Ezen eszköz az elképzelések szerint lehetőleg karóra méretű és viselhető is kell hogy legyen. Az eszközön futnia kell egy programnak, amely egy kisebb számításigényes, de már jó hatékonyságú gesztusfelismerő algoritmust képes futtatni.

2011 februárjában kapcsolódtam be a kutatásba. Először létrehoztam egy áramkört, mely távirányításra használható. Készítettem egy már meglévő mikrokontrollert tartalmazó eszközre csatlakoztatható panelt, melyre egy gyorsulásmérő ültethető, továbbá amelyen infra adatátvitelhez szükséges eszközök és további tesztelési funkciójú elemek találhatóak (erről áttekintést az 5. fejezetben adok). Ezen eszköz az iPhone készülék gyorsulásmérőjéhez hasonlóan a gyorsulásmérő pozícióját tudta csak megmondani. A jobb oldalára fordított eszköz például képes a televízión a hangerő növelését előidéző parancsot kiadni, bal oldalára döntött helyzetében pedig annak csökkentését. Ez azonban nem igazán tekinthető ténylegesen gesztusnak, mivel ennél bonyolultabb elmozdulások érzékelésére nem volt alkalmas.

Munkám során egy olyan eszköz fejlesztését tűztem ki célul, mely önállóan működni képes egy kis számítási kapacitású platformon, ami teljes mobilitást tesz lehetővé. Az algoritmus célja továbbá, hogy ne igényeljen tanító mintákat (legyen személyfüggetlen). A jelenleg létező megoldások a mobilitást nem biztosítják kellőképpen. Ennek oka vagy az, hogy kamerákkal egy térben kell tartózkodni, vagy az adatfeldolgozó műveleteknek túl nagy a számításikapacitás-igénye. Bizonyos elterjedten alkalmazott módszerek tanító mintákat igényelnek, mint például egy HMM [4] vagy SVM alapú osztályozó [5]. Már csak ezen tanító minták beolvasása és tárolása sok erőforrást igényel. Érdemes megemlíteni, hogy létezik személyfüggetlen megoldás is [6], mely nem igényel tanító mintákat de az alkalmazott diszkrét koszinusz transzformáció ugyancsak erőforrásigényes művelet.

[5] http://www.sparkfun.com/products/8563

A gyorsulásmérő szenzoron alapuló felismerés jelenleg nem megoldott ilyen kis platformon. A létező hasonló megoldások [7] közös hátránya, hogy a gyorsulásmérő adatok feldolgozását nem önmaguk végzik.
Egy háttérben futó számítógép megléte azonban szintén korlátozza a mobilitásts. A HCI-kutatások célja viszont minél természetesebb módon lehetőséget biztosítani a felhasználó számára, ezért egyértelműen van igény egy ilyen eszközre.

3. Gesztusok dinamikája

Bizonyos jól definiálható és egymástól jól elkülöníthető gesztusokhoz tartozó adatfolyamok elemzése után felállítottam egy analitikus modellt azok egy lehetséges, minél tömörebb leírására. Tudjuk, hogy az elmozdulás és a gyorsulás egymással szoros kapcsolatban állnak, egy elmozdulás-időfüggvényből a gyorsulás-időfüggvény megkapható annak kétszeri deriválásával, illetve egy gyorsulás-időfüggvényből az elmozdulás-időfüggvény annak kétszeri integrálásával. Mivel rendelkezésünkre állnak gyorsulásmérő szenzorok, ezért kézenfekvőnek tűnhet, hogy a gyorsulásmérő által szolgáltatott jelalak kétszeri integrálásával egyszerűen megkaphatnánk az elmozdulás időfüggvényét. Ez azonban korántsem ilyen egyszerű, mivel a kétszeri integrálás elvégzése után akkorává válik a zajból eredő, kisebb-nagyobb amplitúdók által halmozott hiba, hogy gyakorlatilag nem lehet visszakövetkeztetni arra, milyen elmozdulás is történhetett. Ezen hibák kiküszöbölésére elképzelhető, hogy a szenzorról beérkező kvantált jelet valamilyen polinomfüggvénnyel közelítve végezzük el az integrálást, azonban ezek a műveletek messze meghaladják azt a számításikapacitás-igényt, amit egy karóra méretű eszközben elhelyezhető processzor valósidőben ki tudna szolgálni. Tehát ez nem járható út. Dolgozatomban a másik lehetőséget veszem alapul, vagyis azt használom fel, hogy egy elmozdulás-időfüggvényből annak kétszeri deriválásával megkapható a hozzá tartozó gyorsulás-időfüggvény. Az ötletem a következő: a gesztusokat mint elemi építőkockákból felépíthető alakzatokat fogom bevezetni, majd ezen elemi építőkockák analitikus leírását követően bemutatok egy lehetőséget ezen építőkockák programozástechnikalag tömör leírására. Ha ez megvan, már csak ezen tömör alaknak megfelelő mintázatokat kell megtalálni a vizsgálandó jelalakban. Egyszerűbben fogalmazva, egy megoldást mutatok arra, hogy az idealizált jelalakot leíró jellemző mintázatot hogyan detektáljuk egy valós adatsoron.
Jelen fejezet első alfejezetében az egyidőben egy tengely mentén, egy dimenziós (adott gyorsulásmérő-tengely irányában értendő) elmozdulásokat és ezekből származtatható gesztusokat tárgyalom. A második alfejezetben bemutatom olyan komplexebb gesztusok analitikus leírásának lehetőségét, melyek esetén felhasználom azt a tényt, hogy két tengely mentén (síkban) végzett mozgás leírható az egyes tengelyekre vett mozgáskomponensek szuperpozíciójaként.

3-1. Egy tengely mentén végzett mozdulatok

Ebben az alfejezetben olyan mozdulattípusokat tárgyalok, melyek esetén adott időszeletben egy tengelyen történik elmozdulás. Megjegyzem, ettől még síkbeli alakzatokról van szó, pl. ha elképzelünk egy L-betű alakú gesztust, és a száraival párhuzamosan vannak felvéve a gyorsulásmérő koordináta-rendszerének tengelyei, akkor a függőleges szárának megrajzolása esetén csak a függőleges, míg a vízszintes szára esetén csak a vízszintes tengely mentén történik elmozdulás. Tehát olyan gesztusokról lesz szó, melyek éptítőkockáit olyan egyenes mozdulatok alkotnak, melyek a gyorsulásmérő koordinátarendszerének tengelyeivel jó közelítéssel párhuzamosak, de legrosszabb esetben is nem több mint 15 fokot zárnak be.

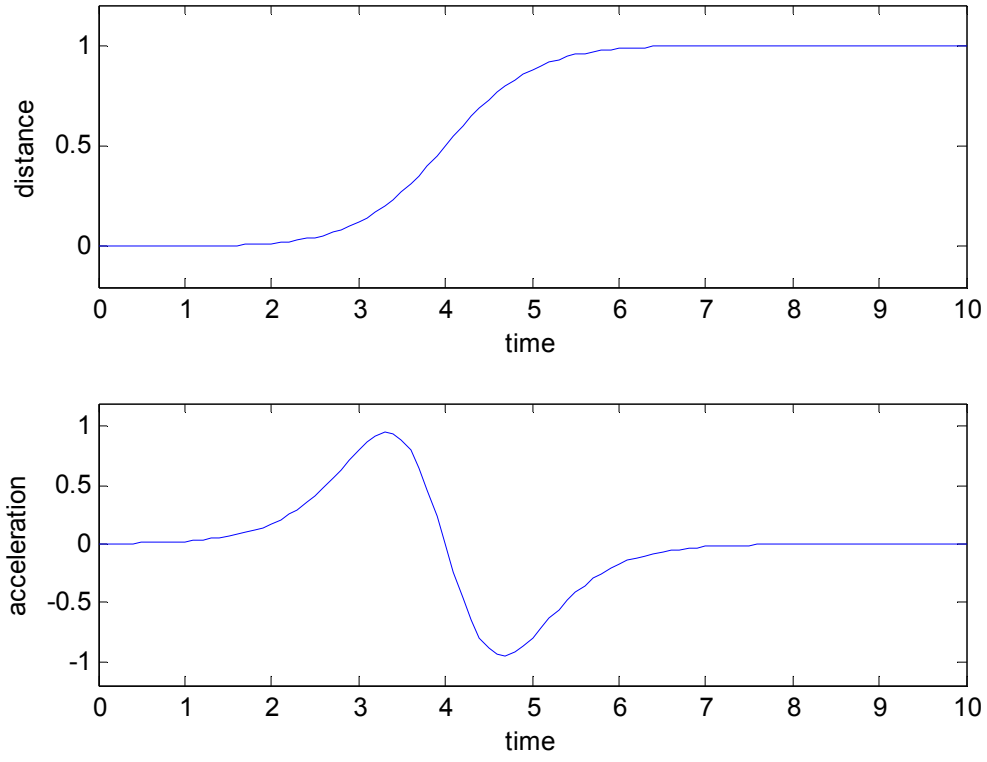
3-1.1. Egyszerű elmozdítás

A gyorsulásmérőnek úgy ajánlatos elhelyezkednie, hogy valamely tengelyével párhuzamos irányú, de attól maximum 15 fokban térjen el a nehézségi gyorsulás iránya, így válhat lehetővé az, hogy a később részletezett ferde vonalú gesztusokkal ne keverhessük össze az ebben a szakaszban részletezett egyenes vonalú gesztusokat. A legegyszerűbb elképzelhető

elmozdulás a gyorsulásmérő szenzor koordináta-rendszerének origójától egy adott síkban valamely tengely pozitív vagy negatív irányába való elmozdítás.

↑
← 0 →
↓

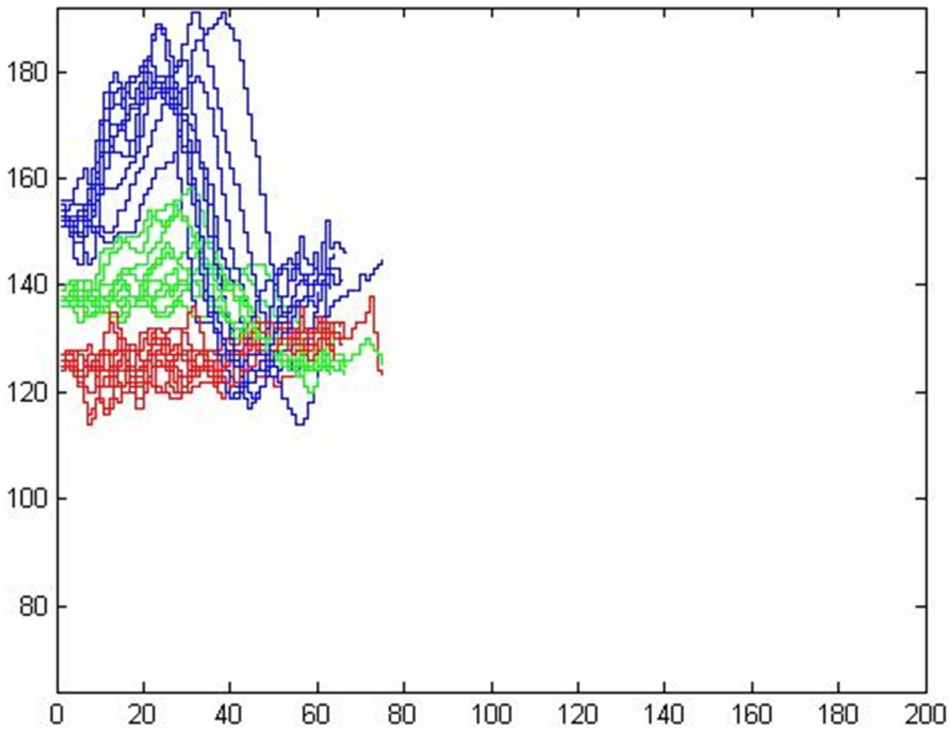
A valóságban az irányok attól függően változnak, hogy a gyorsulásmérő szenzort hogy helyezzük el fizikailag. A tengelyeket szándékosan nem nevezem el x, y, z nevekkel, és irányt sem rendelek hozzájuk, mivel a gesztus-adatbázis mintái esetén az alkalmazott gyorsulásmérő z-x, míg a megvalósított hardver esetén az általam használt gyorsulásmérő y-z síkjában dolgozok, ezért könnyen összekeveredhetnének a jelölések, márpedig a bemutatott módszer független a választott koordináta-rendszertől. Az egyszerűség kedvéért tehát tételezzük fel, hogy a magunk elé képzelt síkban az égtájaknak megfelelően nevezzük el az irányokat, az egyik koordináta-tengely mentén - nevezzük ezt függőleges tengelynek - az északi és a déli irányok vannak, a másik mentén pedig - nevezzük ezt vízszintes tengelynek - értelemszerűen a keleti és a nyugati irányok.
Az északi irányú elmozdítást az ún. *sigmoid* függvénnyel közelítettem, ami a 3-1. ábra felső részén látható, alatta ezen függvény második-deriváltja látható (normalizált egységekben). Ez egy egyszerű modellje az elmozdításnak, mely figyelembe veszi, hogy az elmozdulás véges idő alatt bekövetkező folyamatos jelváltozást jelent. Közelebb állna a valósághoz, ha a beállás során túllövése is lenne a jelalaknak, azonban az csak szükségtelenül bonyolítaná az ebből származtatható gyorsulás-időfüggvényt.



3-1. ábra

Látható, hogy egy pozitív irányú elmozdításnak egy pozitív majd negatív irányban kilengő gyorsulás-időfüggvény felel meg. Könnyen belátható, hogy az adott tengely mentén egy negatív irányú elmozdításnak ilyesformán egy negatív majd egy pozitív irányban kilengő gyorsulás-időfüggvény feleltethető meg. Tekintsük most a gesztus-adatbázis 5. sorszámú gesztusát.

A 3-2. ábrán a gesztus-adatbázis valamely személy 10 gyorsulásmintájának időfüggvénye látható az 5. sorszámú gesztusra vonatkoztatva. Kékkel a számunkra érdekes tengely mentén történő jelváltozások figyelhetőek meg, pirossal és zölddel a gyorsulásmérő koordináta-rendszerének elvileg számunkra érdektelen tengelyei mentén mért jelváltozás látható (ezen tengelyeknek elvileg konstans nyugalmi gyorsulást kellene hogy mutassanak precíz mozdulat esetén). Az egyes tengelyeken a nyugalmi gyorsulás értéke különböző, attól függően, hogy az 1 g nehézségi gyorsulás irányához képest milyen irányba mutatnak. Ez a gyorsulásmérő fizikai elhelyezkedésére utal. Ha jobban megfigyeljük, látható, hogy valamely tengelyen a hozzá tartozó kék színnel jelölt gyorsulásértékek egy negatív majd egy pozitív irányú kilengést mutatnak, ami megfeleltethető a 3-1. ábra alsó időfüggvényének.

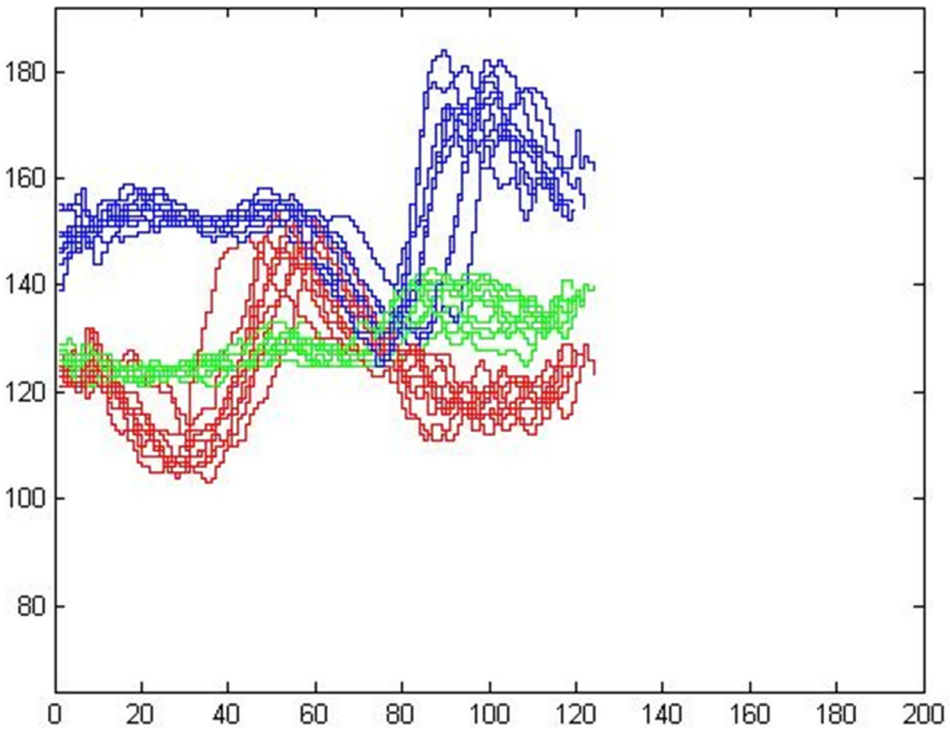


3-2. ábra

Ezekből az elmozdulás-építőkockákból különféle szögletes gesztusok származtathatóak le, pl. egy L betű megfeleltethető egy déli majd az ezt követő keleti irányú elmozdulásnak. Bonyolultabb gesztusminták megalkotása esetén egy építőkocka végpontja tehát a következő origója. A gesztus-adatbázison a 9. sorszámú gesztus egy keleti majd déli irányú egyszerű elmozdításból leszármaztatható.

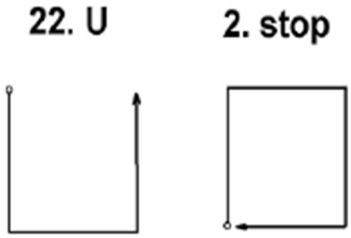
9. right down

Az gesztus-adatbázisból valamely személy ezen gesztushoz tartozó 10 mintjának egymásra rajzolása látható a 3-3. ábrán. A 3-2. ábráról tudjuk, hogy a függőleges irányú elmozdulás a kék színnel jelzett adatokhoz tartozik, tehát nyilvánvaló, hogy a piros színnel jelzett adatok a vízszintes elmozdulást jelentik. A 3-3. ábráról leolvasható, hogy először vízszintes irányban történt elmozdulás, ezt a pirossal jelölt adatsoron a negatív majd pozitív irányba történő kilengésből lehet látni, mindeközben a kékkel jelölt adatok elvileg konstans értéken vannak. Ezt követően a pirossal jelölt adatsoron áll be valamilyen nyugalmi helyzetbe, és ekkor következik be a függőleges elmozdulás, ami az 5. sorszámú gesztus 3-2. ábrán feltüntetett irányításával ellentétes (először a negatív, aztán a pozitív irányú kilengés), tehát ha referenciairányokban gondolkozunk, akkor azt is láthatjuk, hogy azzal ellentétes irányú. Látható, hogy az 5. sorszámú gesztus esetén valóban függőlegesen felfelé történő, a 9. sorszámú esetén pedig lefelé történő elmozdulás történt a kék színhez asszociált tengely esetén.



3-3. ábra

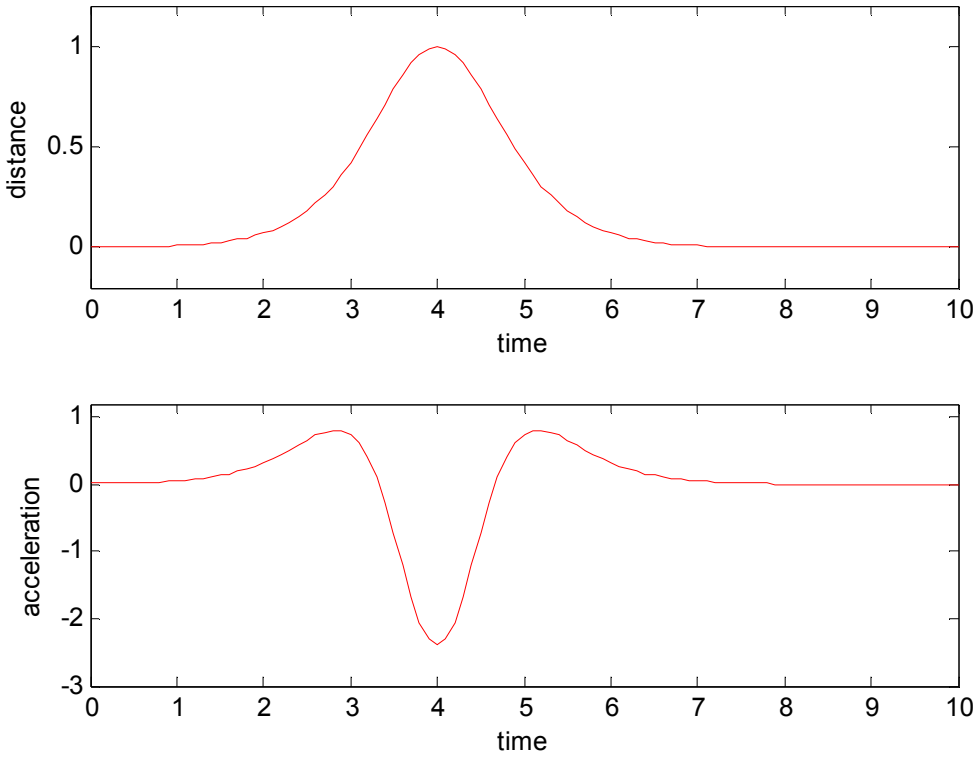
Látható, hogy ilyesformán pl. a gesztus-adatbázis 22. és 2. gesztusa is leszármaztatható egyszerű elmozdításokból, továbbá az adatbázisban nem szereplő, sok egyéb gesztus definiálható.



Ezen gesztusokat hosszúságuk alapján 1, 2, 3 illetve 4 hosszúságú csoportokba lehet sorolni. Úgy gondolom, hogy az 1 hosszúságú, egyszerű elmozdulásokhoz önmagukban nem célszerű gesztust társítani, mivel túl könnyen összekeverhető a felhasználó nem tudatos mozdulataival. 3-nál (vagy 4-nél) hosszabb gesztusokat túl sokáig tart megrajznolni, ezért egyszer funkciók eléréséhez pedig kényelmetlenül hosszúak lennének, és persze a felismerés hatékonysága is jelentősen lecsökken, ha túl hosszúra választjuk egy lehetséges gesztus hosszát.

3-1.2. Oda-vissza elmozdítás

Egy adott tengely mentén oda-vissza elmozdításnak megfeleltethető elmozdulás modellezése, illetve az ehhez tartozó gyorsulás-időfüggvény a 3-4. ábrán látható.



3-4. ábra

Elmondható, hogy a gyorsuláskomponensek két kisebb pozitív irányú kilengésből és közöttük egy nagyobb negatív irányú kilengésből állnak. A gesztus-adatbázisban ilyen oda-vissza elmozdulást tartalmazó gesztusminta nincs, de elképzelhetőek ilyen rázás-jellegű gesztusok. Ezt az elmozdulás-jelalakot itt találtam célszerűnek bevezetni, de a következő alfejezetben tárgyalt görbevonalú alakzatoknál lesz igazán jelentősége, mint egy görbevonalú gesztus egyik tengelyre vetülő komponense.

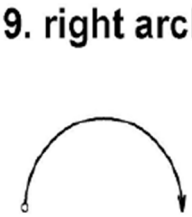
3-2. Két tengely mentén végzett mozdulatok

3-2.1. Ferde elmozdítás

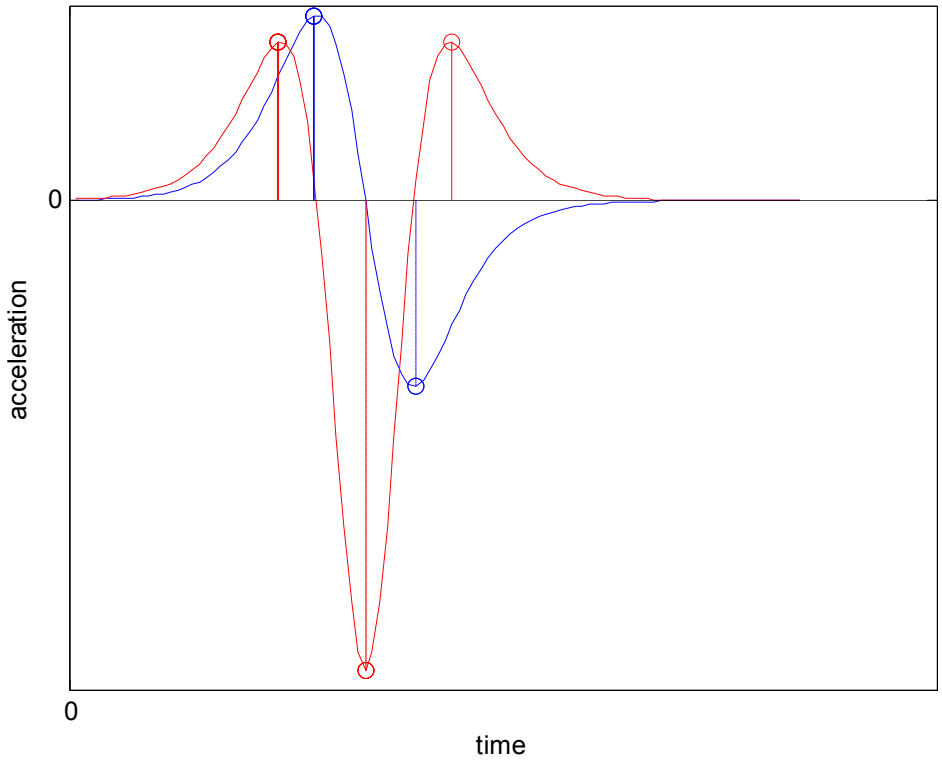
Ferde elmozdítás alatt a gyorsulásmérő koordináta-rendszerében, annak tengelyeihez képest 45°-os elmozdításokat értek. A 3-1. ábrán látotthoz képest könnyű úgy elképzelni ezen típust, hogy mindkét tengelyen egyidőben történik az elmozdulás. Az északkeleti irányú ferde elmozdítás pl. a vízszintes tengelyen keleti irányba való elmozdítás és a függőleges tengelyen északi irányú elmozdítás eredője. Ezen gesztusok felismerése a dolgozat írása idején még nem kellőképpen kidolgozott.

3-2.2. Görbevonalú elmozdítás

A görbevonalú elmozdulások alapeseteként tekintsünk egy félkört. Legyen ez a félkör a gesztus-adatbázis 19. gesztusa.

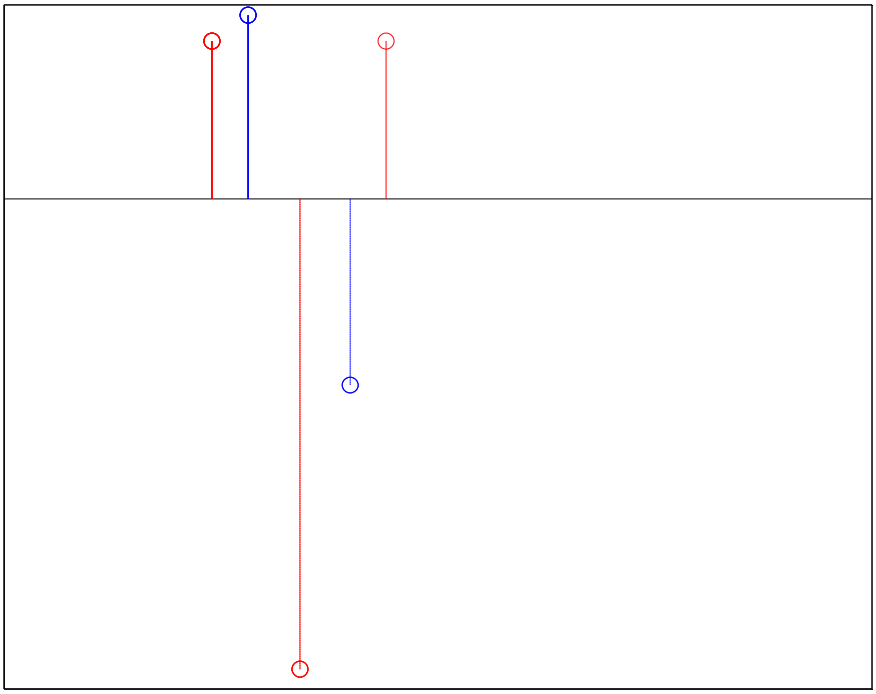


Ezen gesztus tengelyekre bontott elmozdulás-időfüggvényeit elemeztem. A vízszintes tengelyen a 3-1.1. pontban leírt sima elmozdulást, a függőleges tengelyen a 3-1.2. pontban leírt oda-vissza elmozdulást lehet megfigyelni. A modell alkotásához a 3-5. ábrán egymásra rajzoltam a két tengelyen egyidőben lejátszódó két gyorsulás-időfüggvényt (melyek a 3-1. és a 3-4. ábrák alsó időfüggvényei), bejelöltem továbbá a csúcsokat a könnyebb azonosíthatóság kedvéért. Látható, hogy ezen gesztus rajzolásakor a sima elmozdulásokkal ellentétben már bonyolultabb csúcsmintázat jön létre, az elmozdulás 2 csúcsot, az oda-vissza elmozdulás 3 csúcsot eredményez, ezek a csúcsok az ábrán látható módon követik egymást.



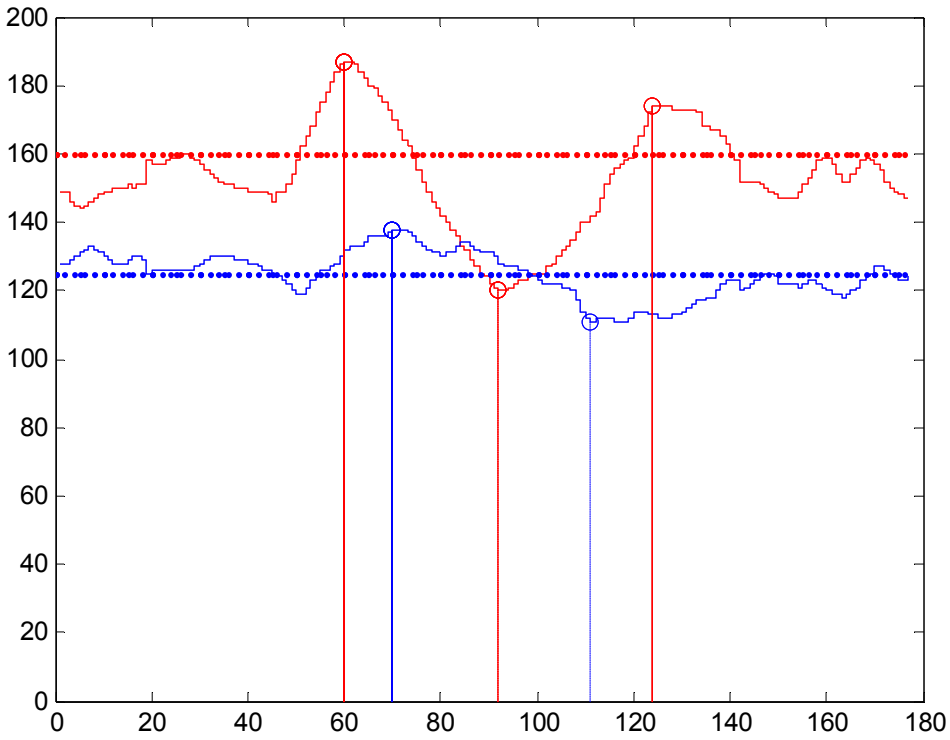
3-5. ábra

A jelalakok tömör leírása csak a csúcsok adatait veszi alapul a 3-6. ábrán kiemeltem a 3-5. ábrán látható jelalakhoz tartozó csúcsokat. Ez a csúcsmintázat egyértelműen leírja ezt a gesztust: a függőleges tengelyen egy maximum után a vízszintes tengelyen is egy maximum következik, ezt követően a függőleges tengelyen egy minimum, a vízszintes tengelyen egy minimum, majd ismét a függőleges tengelyen egy maximum.



3-6. ábra

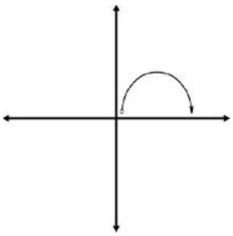
Amennyiben egy adatfolyamon ezt a csúcsmintázatot detektáljuk, onnantól kezdve mondhatjuk azt, hogy a 19. számú gesztust beazonosítottuk. A 3-1.1. és a 3-1.2. alpontban burkoltan, de szintén csúcsmintázatról volt szó, bár ott könnyen el lehetett különíteni az egyes tengelyekre vonatkozó mintázatot, itt pedig ilyesformán „egymásba ölelkezve” jelennek meg a csúcsok. A 3-7. ábrán bemutatom a gesztus-adatbázis valamely ezen gesztushoz tartozó mintáját, kiemelve a jellemző csúcsokat (melyet a később tárgyalt algoritmussal detektáltam). A szaggatott vízszintes vonalak az egyes tengelyekhez tartozó nyugalmi gyorsulás értékek, melyekhez viszonyítva tekintem valamely csúcsot lényeges csúcsnak. Ezen ábra egyben igazolása is annak, hogy a modell működik a valóságban, és tényleg olyan mintázatot kapunk, ami megfelel az előzetes elvárásoknak.



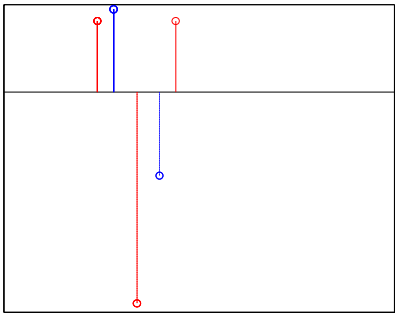
3-7. ábra

A félköröket az összes lehetséges irányítottsággal elképzelve 8-féle félkör-alakú gesztus különíthető el, ezeket és a rájuk jellemző csúcsmintázatokat a 3-1. táblázatban foglalom össze, itt látható, hogy hogyan cserélődnek fel a csúcsok különböző irányban indított félkör-alakú gesztusok esetén. Szögletes alakzatokból még ha max. 4 eleműekre korlátozzuk a lehetséges gesztuskészletet, nagyon sok képzelhető el, ám egy megértése után lehet általánosítani a csúcsmintázatokra vonatkozó állításokat, azonban a félkör-alakzatokat fontos építőkockáinak tartom pl. a köralakú vagy az m- illetve a w alakú gesztusokhoz, ezért a 8 alapesetet célszerűnek tartom bemutatni. Az itt megfigyelhető szabályosságokra épül a később tárgyalt algoritmus görbevonalú gesztusok detektálásáért felelő részének programkódja.

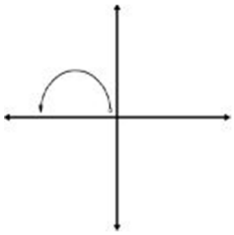
1.



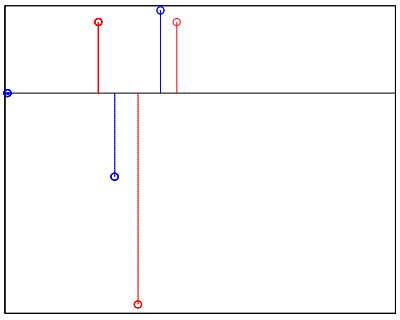
vízsz. tengely: K-i irányban elmozdulás
függ. tengely: É-i irányban oda-vissza lengés



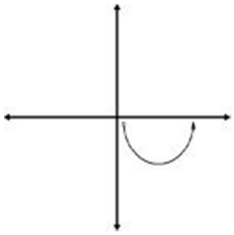
2.



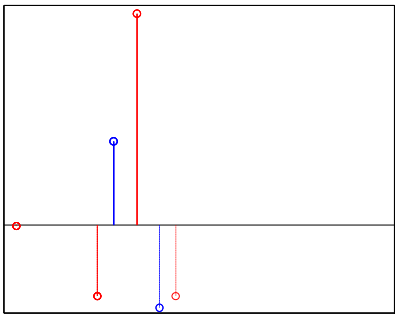
vízsz. tengely: Ny-i irányban elmozdulás
függ. tengely: É-i irányban oda-vissza lengés



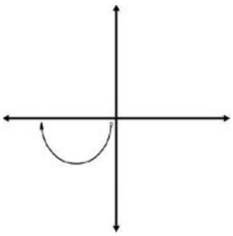
3.



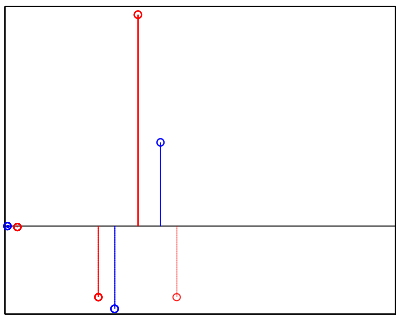
vízsz. tengely: K-i irányban elmozdulás
függ. tengely: D-i irányban oda-vissza lengés



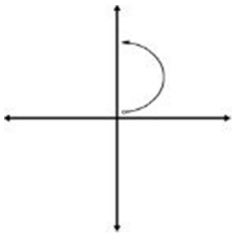
4.



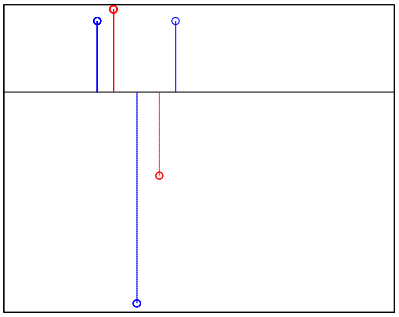
vízsz. tengely: Ny-i irányban elmozdulás
függ. tengely: D-i irányban oda-vissza lengés



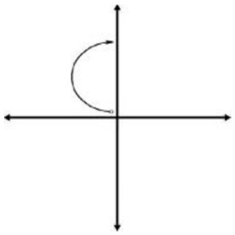
5.



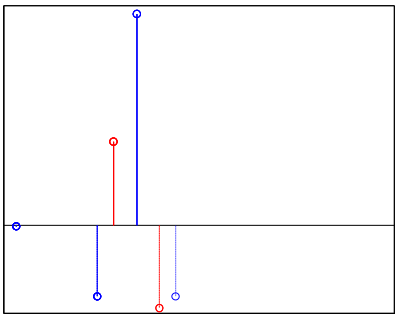
vízsz. tengely: K-i irányban oda-vissza lengés
függ. tengely: É-i irányban elmozdulás



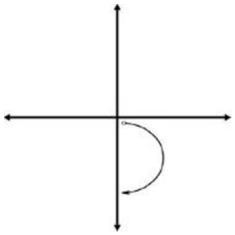
6.



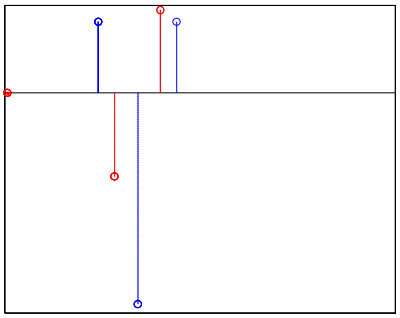
vízsz. tengely: Ny-i irányban oda-vissza lengés
függ. tengely: É-i irányban elmozdulás



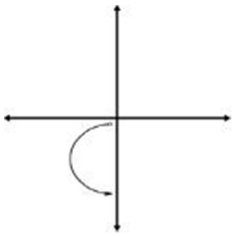
7.



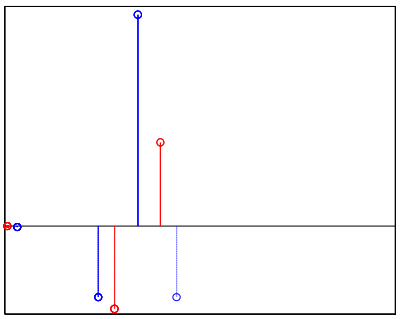
vízsz. tengely: K-i irányban oda-vissza lengés
függ. tengely: D-i irányban elmozdulás



8.

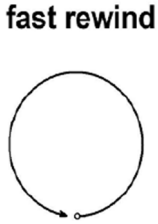


vízsz. tengely: Ny-i irányban oda-vissza lengés
függ. tengely: D-i irányban elmozdulás

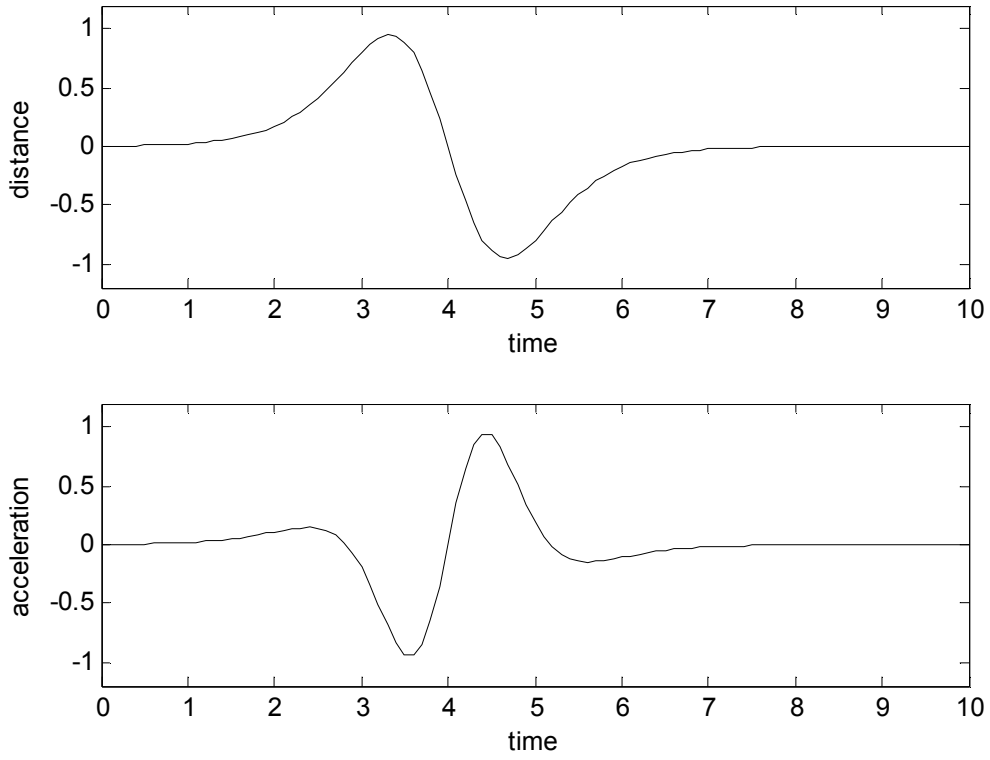


3-1. táblázat

Látható, hogy az egyéni csúcsmintázat alapján egyszerűen el lehet különíteni ezen 8 gesztust.
A következőkben bemutatom, hogy az analitikus módszerrel hogy lehet egy kör alakú gesztust leírni, ezt követően bemutatom, hogy a kör alakú gesztushoz tartozó csúcsmintázatot hogyan lehet bizonyos félkör-alakú csúcsmintázatokból leszármaztatni.
A gesztus-adatbázisban 8. sorszámmal szereplő teljes kör alakzatot veszem alapul.

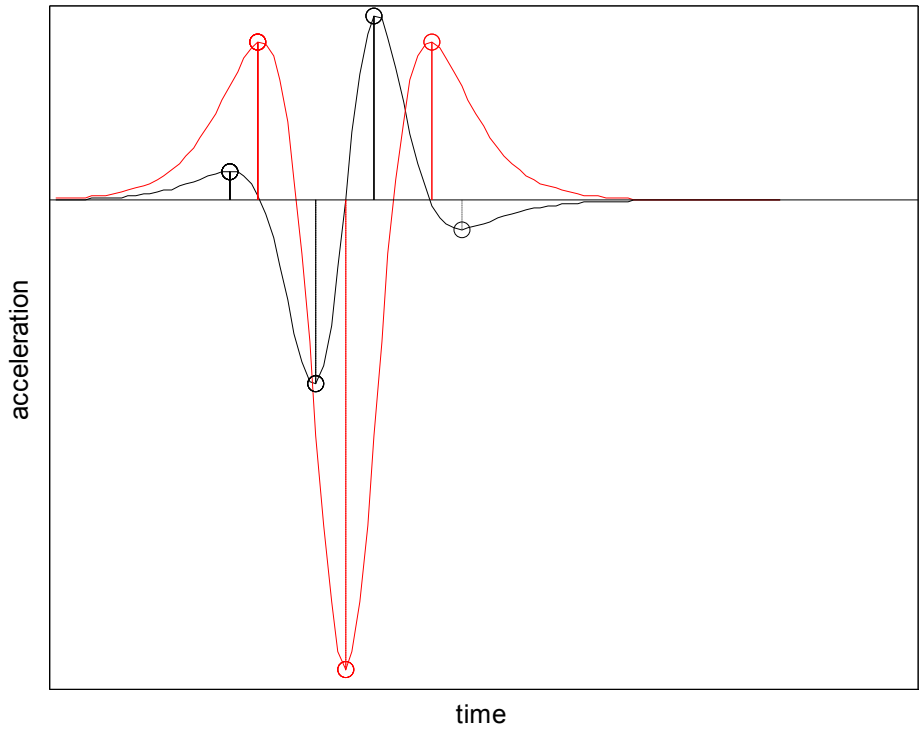


Az analitikus megközelítés szerint északi irányban egy oda-vissza mozdulattal írható le annak függőleges tengelyre vetülő része, melyet a 3-1.2. pontból már ismerhetünk. A vízszintes tengelyen azonban egy új, keleti irányban kitérő, majd ezt követően visszatérve nyugatra kitérő, majd alaphelyzetbe beálló mozgást figyelhetünk meg. Ezen mozgáshoz analitikusan rendelt függvényt a 3-8. ábra felső részén figyelhetünk meg, az ehhez tartozó gyorsulás-időfüggvény a lenti részen látható.

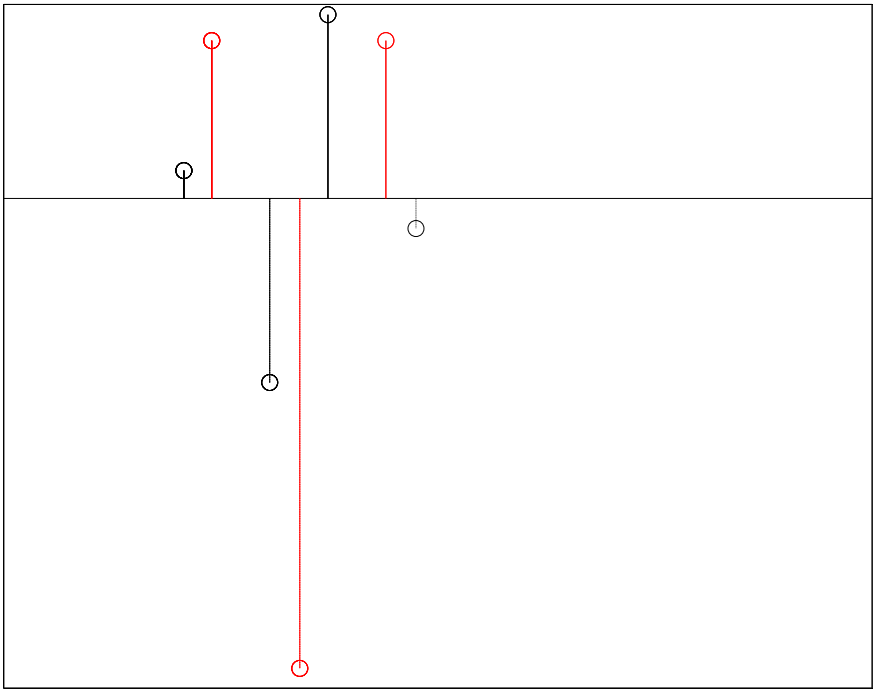


3-8. ábra

A gyorsulás időfüggvény esetünkben modellezhető egy kisebb, pozitív irányba történő kilengést követően egy nagyobb, negatív irányba, majd egy nagyobb, pozitív irányba történő, végül egy kisebb, negatív irányba történő kilengéssel.
A 3-4. ábrán és a 3-8. ábrán található gyorsulás-időfüggvények fognak tehát egyidejűleg bekövetkezni, ezen projekció eredménye a 3-9. ábrán látható, a 3-10. ábrán a csúcsmintázat.

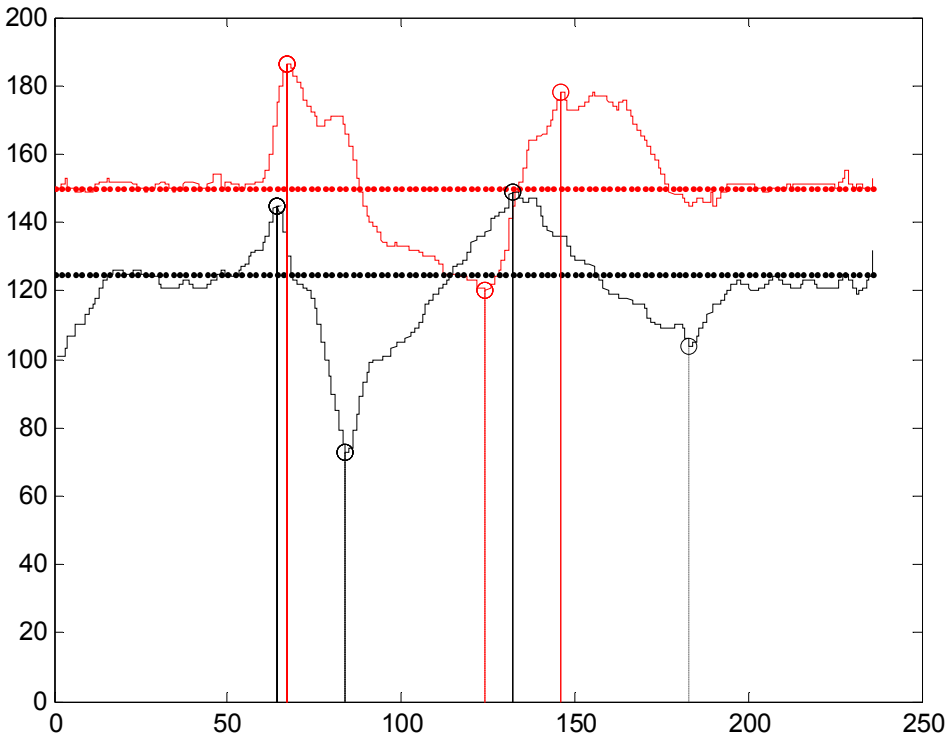


3-9. ábra



3-10. ábra

A 3-11. ábrán a 3-7. ábrához hasonlóan egy a gesztus-adatbázisból vett valós adatfolyam látható, ahol ezt a csúcsmintázatot detektáltam, és megtaláltam a 8. számú kör alakú gesztust.



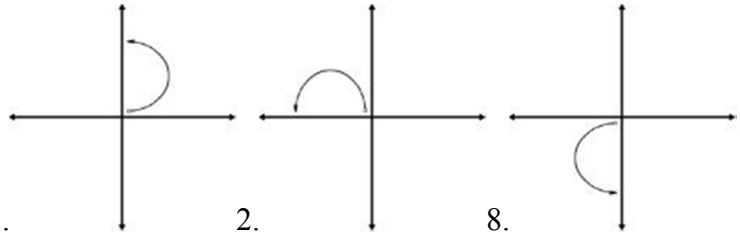
3-11. ábra

Mielőtt bemutatnám, hogyan lehet a félkör-alakú gesztusokhoz tartozó csúcsmintázatok alapján egy körhöz tartozó csúcsmintázatot leszármaztatni, bevezetek egy egyszerű leírási módot. Amennyiben a 3-1. táblázatban leírt 8 félkör-alakú gesztust szeretnénk lekódolni, akkor élhetünk a következő egyszerű választással: az x-tengelyen található maximum illetve minimum kódja legyen 1-es illetve 2-es, az y-tengelyen található maximum illetve minimum kódja pedig 3-as és 4-es.
Az így adódó kódokat a következő táblázatban foglalom össze:

félkör sorszáma	csúcsok sorrendje
1.	3,1,4,2,3
2.	3,2,4,1,3
3.	4,1,3,2,4
4.	4,2,3,1,4
5.	1,3,2,4,1
6.	2,3,1,4,2
7.	1,4,2,3,1
8.	2,4,1,3,2

3-2. táblázat

Tekintsük most a 3-10. ábrán látható mintázatot, mely kódja a félkörökre alkalmazott szabály alapján 1,3,2,4,1,3,2. A vizsgált körünket próbáljuk leírni az ötödik, a második majd a nyolcadik félkör alakú gesztusból.



5. 2. 8.

Megyfigyelhető átlapolódás ezen építőkockák között, figyeljük meg ezt az átlapolódást a csúcsok kódja esetén!

5.	1	3	2	4	1		
2.		3	2	4	1	3	
8.			2	4	1	3	2

Olvassuk most össze az egyes oszlopokban szereplő elemeket: megvan az 1,3,2,4,1,3,2!
Ilyesformán 8 különböző irányban képzett körök leírása gyorsan megalkotható.

4. Szoftveres fejlesztés

A 3. fejezetben leírtak figyelembe vételével elmondható, hogy az algoritmus alapjában véve a rendelkezésre álló adatfolyamon megvalósít

- csúcsdetektációt
- a detektált csúcsokon elemzést, mintakeresést

4-1. Definíciók

Az alábbiakban néhány definíciót vezetek be, melyet az algoritmus leírásához szükségesnek tartok. Ezek a definíciók egymásra is erősen épülnek.

4-1.1. Vizsgálati sík

A feladat megoldása során 3 tengelyű gyorsulásmérővel dolgoztam. A gyorsulásmérő origójához képest, annak x, y- ill. z tengelye mentén történhetnek elmozdulások, melyet a szenzorral érzékelni tudunk. Azt a síkot, amit a vizsgálat szempontjából kitüntetettnek veszünk, nevezzük vizsgálati síknak, innentől kezdve csak az ezen síkot meghatározó két tengely gyorsulás-adatait vesszük figyelembe. Vizsgálati síknak a szenzor tengelyeivel párhuzamos – ill merőleges tengelyek által meghatározott síkot választottam, de lehetséges ferde koordináta-rendszerben is dolgozni. Ha a feldogolzó eszközt a karunkra rögzítjük, ésszerű lehet számításba venni, hogy bizonyos testhelyzetekben pl. mindig egy adott dőlésszögben van a gyorsulásmérő a karunkhoz képest.

4-1.2. Tengelyre vonatkoztatott nyugalmi gyorsulás

Miután kijelöltem a vizsgálati síkot, tudnom kell, hogy mozdulatlan alaphelyzetben milyen irányból és milyen mértékben hat a nehézségi gyorsulás az általam választott tengelyekre, és ezt mint eltolás tudom a számításaimba belekalkulálni. Pl. egy egyszerű választás esetén:
x tengely: felfelé mutat, nyugalmi gyorsulása -1 g, y tengely, jobbra mutat, nyugalmi gyorsulása 0 g.

4-1.3. Csúcs

Egy csúcsot a következő 3 adatával tekintem egyértelműen leírtnak:

- típus, mely megadja, hogy a csúcsot a vizsgálati sík melyik tengelyén detektáltuk, és hogy minimumként vagy maximumként jegyeztük-e fel, ezeket az alábbi (4-1.) táblázatban bevezetett egyszerű számozással különíthetjük el

	első tengely	második tengely	(harmadik tengely)
min	1	3	(5)
max	2	4	(6)

4-1. táblázat

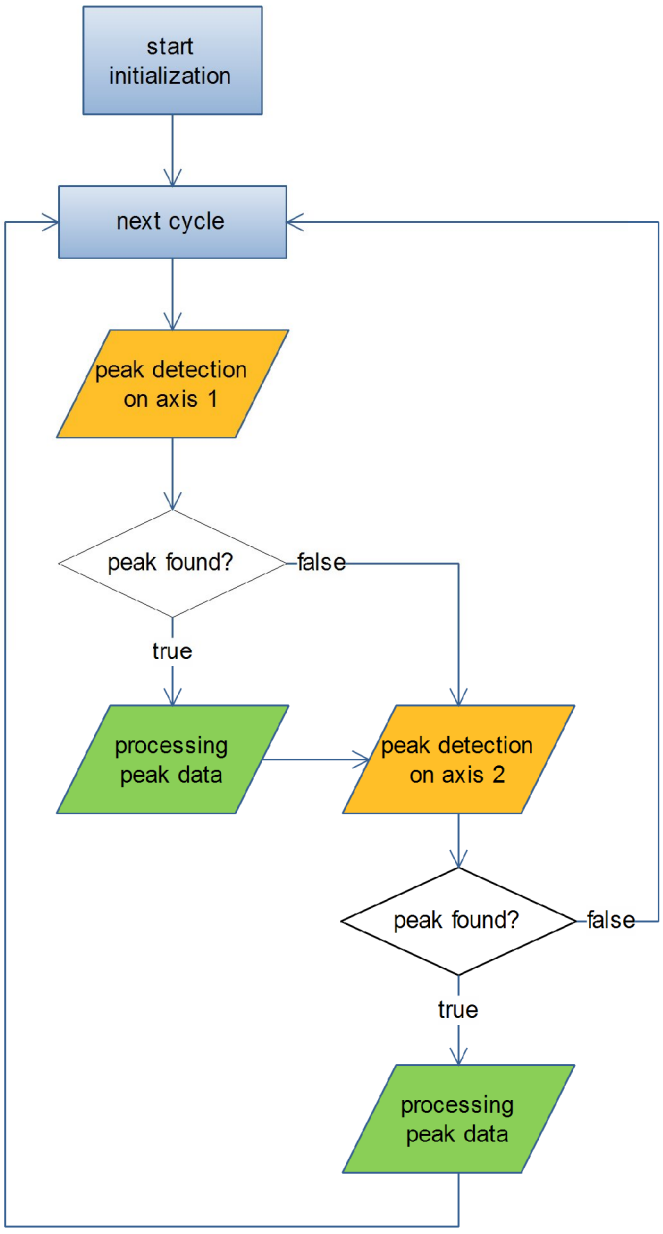
- nagyság, melyet célszerűen az adott tengelyre vonatkoztatott nyugalmi gyorsulástól vett abszolút eltérés fogja megadni
- időbeli megjelenés, azaz az indulástól számított hányadik fő ciklusban ismertem fel (a fő ciklus egyszeri futása során tengelyenként egy minta vételezése történik meg, a fő ciklus pedig adott időközönként fut le)

4-1.4. csúcsmintázat

A 3. fejezetben leírtak alapján a gesztusokat egy analitikusan meghatározott szabályrendszer alapján egymást követő csúcsok mintájaként egyértelműen meghatározottnak tekintem.

4.2. Az algoritmus legáltalánosabb vázlata

Ezen definíciók után tekintsük az algoritmus legáltalánosabb vázlatát!



4-1. ábra

A kezdeti inicializáló műveletek elvégzése után megkezdődik a fő adatfeldolgozó ciklus. Amennyiben az elemzést statikus, valamely adatbázison rendelkezésre álló adatokon végezzük, a feldolgozás akkor ér véget, amikor a rendelkezésre álló adatsor végére ért a program, amennyiben viszont az algoritmus egy hardveres impementációját tekintjük, akkor mindaddig, amíg az eszközt ki nem kapcsoljuk, folyamatosan futni fog a feldolgozó rész.

4-2.1. Inicializáló műveletek

A program indításakor meg kell adni kezdeti inicializáló értékeket. Ezek a később részletezett csúcsdetektáció és csúcsfeldolgzó szubrutin érzékenységi paramétereinek kezdeti értékeiből, továbbá hardveres alapbeállításokból állnak.

4-2.2. Fő ciklus

A fő ciklus egyszeri lefutásakor a vizsgálati síkot meghatározó két tengelyen egy-egy minta vételezése történik meg.
A tengelyeken a mintavételezés a *csúcsdetektáló szubrutinban* történik meg. Amennyiben csúcsot detektáltunk valamely tengelyen, ezt követően elvégezzük a *csúcsfeldolgozó szubrutint*. A *csúcsfeldolgozó szubrutin* elsődleges feladata eldönteni egy újonnan detektált csúcsról, hogy az őt megelőzően detektált csúcsokhoz hozzávéve az együttesen rendelkezésre álló csúcsmintázat egy gesztust határoz-e meg. A másodlagos feladat találat feljegyzése esetén értesíteni erről a felhasználót.

4-3. Csúcsdetektálás

Ebben a szakaszban a 4-1. ábrán narancssárgával jelölt, csúcsdetektáló szubrutint mutatom be. Ahhoz, hogy egy kis számításigényű platformon, valós időben lehessen csúcsokat detektálni, egy olyan algoritmusra volt szükségem, amely a csúcsdetektációt minél egyszerűbben, de ugyanakkor hatékonyan hajtja végre. A témában előzetes kutatás után találtam Eli Billauer peakDet nevű MATLAB-függvényét [8], melyben alkalmazott algoritmust használtam fel a munkám során. Fontos előnye, hogy inkrementálisan találja meg a csúcsokat egy adatfolyamon, tehát nem szükséges az egész adatsor előzetes ismerete, lehetővé téve a folyamatos jelfeldolgozást. Továbbá előnye ezen algoritmusnak az egyszerűsége, és az, hogy semmilyen komolyabb, számításigényesebb műveletet (pl. deriválás) nem igényel. Hogy hogyan működik maga a csúcsdetektáció, feltétlenül szükségesnek tartom bemutatni, ugyanis a későbbiek során a felmerülő problémák és azok megoldásának bemutatásához, egyszóval a saját feladatomhoz történő integrálásához fog a következő elemzés hivatkozási alapot jelenteni.

4-3.1. Az algoritmus alapgondolata

Miután elemeztem az algoritmus működését, arra jutottam, hogy két lényegi pontban megfogalmazható a detektáció megvalósításának alapgondolata:

4-3.1.1. Az algoritmus lokális minimum után lokális maximumot keres, lokális maximum után pedig lokális minimumot. Ez mindig így lesz, még akkor is, ha olyan a jelalak, hogy két lokális minimum követné egymást, ezen minimumok között ebben az esetben ekkor is detektálni fog lokális maximumot. Van, hogy ez rossz mellékhatásként jelentkezik, de egy később részletezett módon az ilyen lokális csúcsokat ki fogom szűrni és zárni a további vizsgálataimból (ld. 4-5.1. pont).

4-3.1.2. Az algoritmus akkor jelenti ki egy pontról, hogy lokális minimum, ha utána legalább egy bizonyos mértékben (nevezzük *deltának*) megemelkedett a jel, és amíg ez az emelkedés bekövetkezett, a kérdéses pontnál kisebb pontot nem észleltünk. Ezzel analóg módon, egy

pont akkor lokális maximum, ha utána legalább deltával lecsökkent a jel szintje, és a csökkenés során a kérdéses pontnál nagyobb pontot nem észleltünk.

Nyilvánvaló, hogy ha egy pontot mindkét oldalról jelszint-csökkenés övez, akkor egy lokális maximumról van szó. A 4-3.1.2. pontban megfogalmazott állítás viszont csupán a pontot követő emelkedést írja elő. Azt, hogy a pontot megelőzően is emelkedés történt, az biztosítja, hogy fennáll a minimumkeresés ténye. Az első pont alapján ugyanis, maximumot egy megtalált minimumot követően fog keresni az algoritmus, de egy megtalált minimum meglétéből már következik az, hogy ezen minimumot követően a jelszint legalább deltával magasabbra került, mielőtt a kérdéses maximum-pontba kerültünk volna. Tehát, látható, hogy biztosítva van a kérdéses pontot övező, elvárható jelszintváltozás. A lokális minimumkeresés jósága belátható a maximumkereséssel való analógiából.

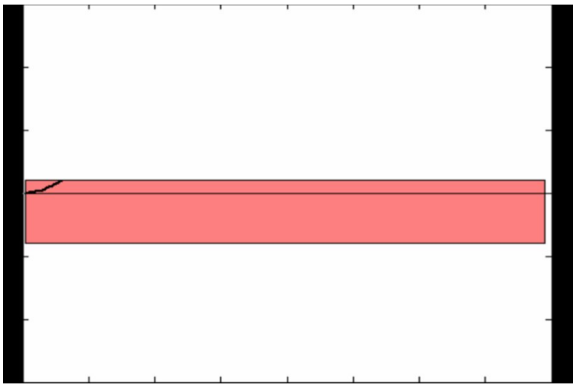
4-3.2. A csúcsdetektáció bemutatása animáció segítségével

A csúcsdetektáció lényegi pontjainak bemutatásához az *f(x)=x*sin(x)* próbafüggvényt választottam, melynek az origótól kiindulva néhány periódusig terjedő szakaszát vettem alapul. Azért esett a választásom erre a függvényre, mert ezen bemutatható a csúcsdetektáció egyetlen érzékenységi paraméterének – a korábban definiált *deltának* – a hatása. Ezen paraméter felelős azért, hogy meghatározhassuk, mi az a küszöb, aminél nagyobb csúcsokat el lehet fogadni lényeges csúcsnak, és aminél kisebb csúcsokat már zajnak, elhanyagolható méretű kilengésnek kell tekintenünk.
Az animáció bemutatása előtt a következő szakaszban dőlttel szedett fogalmakat és a hozzájuk társított jelentést, valamit az animációval való szemléletes kapcsolatot az alábbi (4-2.) táblázatban összefoglaltam.

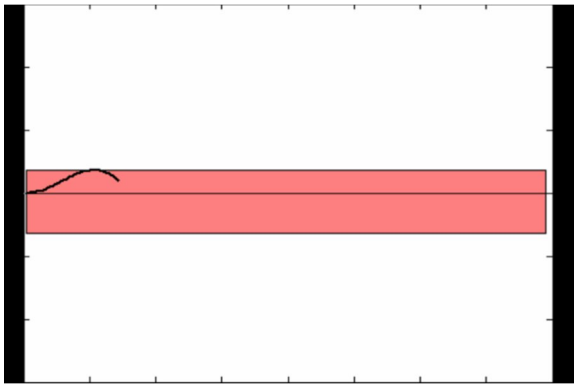
fogalmak	jelentés	szemléltetés az animációkon
delta	A csúcsdetektáció érzékenységi tényezője, a lehetséges csúcsok közül ez alapján lehet kiszűrni a *nagyság* alapján nem számottevő csúcsokat (megjegyzem, később más szempontok alapján is fogok további csúcsokat kiszűrni)	A vízszintes, színezett sáv (*delta*-sáv) konstans vastagsága
maximum-keresés	A 4-2.1.1. pont alapján az algoritmus azon fázisa, mely időben egy lokális minimum detektációjától kezdődően egy lokális maximum detektálásáig tart	A *delta*-sáv pirosra van színezve
minimum-keresés	A 4-2.1.1. pont alapján az algoritmus azon fázisa, mely időben egy lokális maximum detektációjától kezdődően egy lokális minimum detektálásáig tart	A *delta*-sáv kékre van színezve
lehetséges maximum	Az a legnagyobb érték, melyet a maximumkeresés fázisa alatt beolvastunk	A pirosra színezett *delta*-sáv felső határa
lehetséges minimum	Az a legkisebb érték, melyet a minimumkeresés fázisa alatt beolvastunk	A kékre színezett *delta*-sáv alsó határa

4-2. táblázat

Ehhez a szekcióhoz a 4-1. és 4-2. animáció[6] tartozik. A két animációban a csúcsdetektáló algoritmus mindössze a *delta* értékében tér el egymástól. Tekintsük először a 4-1. animációt, annak is az elejét (4-1-a ábra). A *maximumkeresés* állapotát a pirosra színezett sáv – nevezzük ezt *delta*-sávnak – jelzi. Mivel az animáció legelején emelkedik a jelalak, ezért a 4-3.1.2. pontban leírtak alapján mindaddig, amíg ez az emelkedés tart, addig a *lehetséges maximum* az újonnan beolvasott érték lesz, és ezen emelkedés alatt a *delta*-sáv is együtt mozog a jelalakkal. Amint a jel elér egy pontra, mely pontot követően elkezd csökkenni, onnantól kezdve a *lehetséges maximum* a *delta*-sávval együtt "leragad" a csúcsnál, ahol a növekedés átváltott csökkenésbe (4-1-b ábra). Ezt követően mindaddig, amíg a *delta*-sávban tartózkodik a jel, addig nem változik a *lehetséges maximum.*

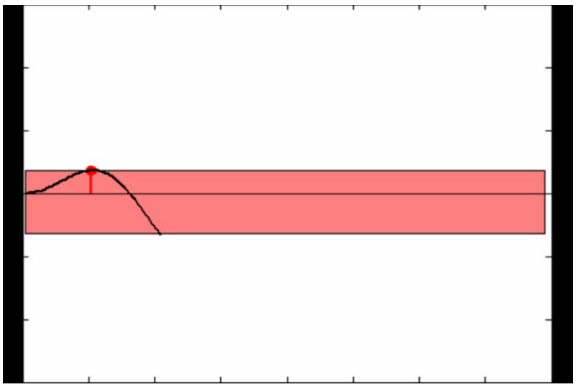


4-1-a ábra

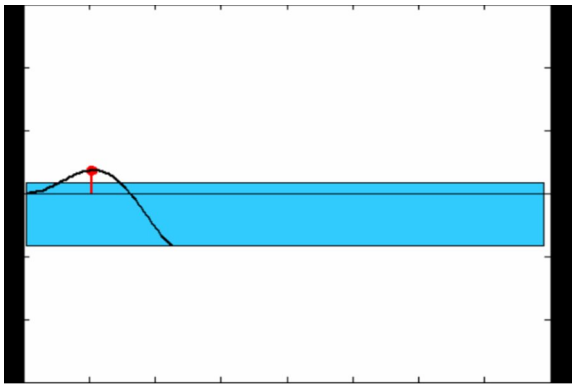


4-1-b ábra

Amennyiben a jel a delta sávot "alulról" hagyja el, akkor teljesül a 4-3.1.2. pontbeli feltétel (a *deltával* lejjebb kerülés), ezáltal a *lehetséges maximumot* most már tényleges, detektált maximumként rögzíthetjük (4-1-c ábra, a detektált maximum piros oszloppal kiemelve).



4-1-c ábra

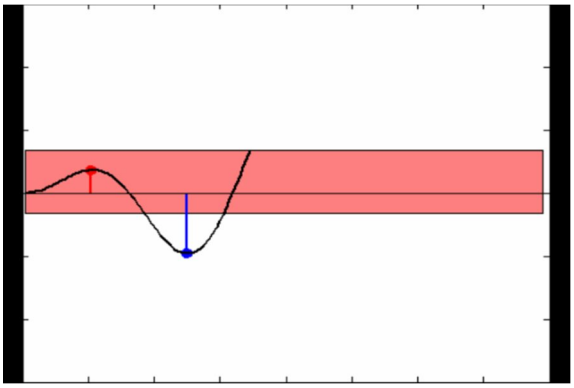


4-1-d ábra

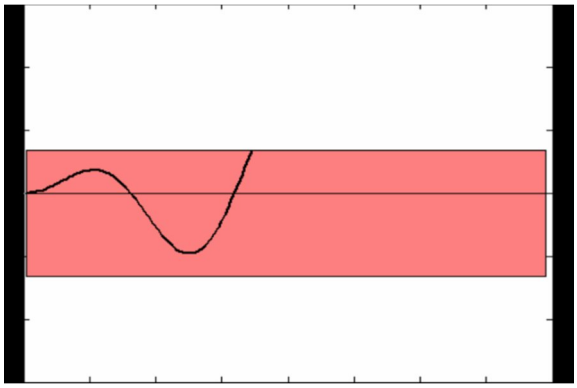
Amennyiben viszont a jel a delta sávot felülről hagyja el, akkor a *lehetséges maximum* értéke, ezáltal a *delta*-sáv feljebb tolódik. Az animáció legelején a jelalak növekedésével a *lehetséges maximum* feljebb tolódása tekinthető ez utóbbi eset legegyszerűbb változatának, amikor is minden újonnan beolvasott érték felüldefiniálja a *lehetséges maximumot* az általa kijelölt *delta*-sávval együtt. Abban a pillanatban, hogy egy maximumot detektáltnak rögzítünk, a

[6] http://szucher.com/tdk2011/4-1.avi | http://szucher.com/tdk2011/4-2.avi

detektáció a 4-3.1.1.pont alapján minimumkereséssel fog folytatódni (4-1-d ábra). Minimumkeresés esetén a *delta*-sáv vastagsága nem változik meg, viszont értelemszerűen megfordulnak a sávhatárok: a sáv alsó határa lesz a *lehetséges minimum* értéke, és a felső határa az ehhez képest *deltával* magasabb szint. Az analógia a minimumkereséshez innentől nyilvánvaló.
A 4-1. és a 4-2. animáció összehasonlítása esetén megfigyelhető, hogy a *delta* érzékenységi tényező hogyan befolyásolja a detektálást. A 4-1. animáció esetében a kisebbre választott *delta*-sáv nagyobb érzékenységet jelent a csúcsokra, a jelalak az első maximumát követően a sávot még elhagyja alulról (4-1-c ábra), mielőtt az első minimumát eléri, ezáltal bekövetkezik a detektáció. Azonban, a 4-2. esetében a jel első maximuma és első minimuma egyaránt kiszűrődik, mivel az első csúcs ugyan az elérése után *lehetséges maximummá* vált, azonban a jel az ezt követő minimum elérésének pillanatában a *delta*-sávban tartózkodott, nem pedig alatta, tehát nem teljesült 4-2.1.2. pontban megfogalmazott feltétel a maximumdetektálásra vonatkozóan (4-2-e ábra).

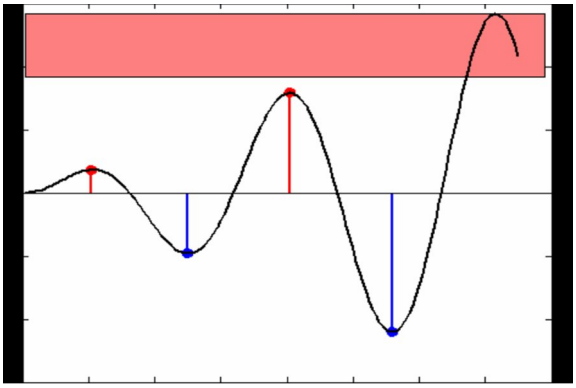


4-1-e ábra

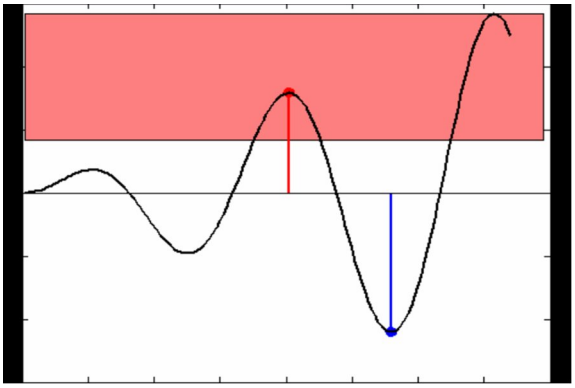


4-2-e ábra

Látható, hogy a 4.2 esetben (4-2-e ábra) jelünk a *delta*-sávot az abban eltöltött, vizsgálatunk szempontjából lényegtelen változása (csökkenés majd emelkedés) után végül felülről hagyta el, így ebben a pillanatban a *lehetséges maximum* is feljebb tolódik. Ez az emelkedés folytatódhat mindaddig, amíg a jel el nem éri a második maximumát, amit már ténylegesen maximumnak fogunk detektálni. A *delta* hatása tehát szemmel látható, a 4-1. esetben meghagyta, míg a 4-2. esetben kiszűrte a kisebb csúcsokat (4-1-f és 4-2-f ábra).



4-1-f ábra



4-2-f ábra

Nagyon fontosnak tartom kiemelni, hogy egy csúcs előfordulásának ideje és detektálásának ideje nem azonos, a detektálás az előfordulást követően valamennyi idő múlva következhet be, attól függően, hogy mennyit tölt a jel a *delta-sávban* a detektálást megelőzően.

4-3.3. Implementáció

4-3.3.1. Változóhasználat

Az előző pontban leírt működés megvalósításához szükséges egyrészt **statikus** változókat bevezetni. Ilyen például a *delta*, mellyel a csúcsdetektáció érzékenységét határozhatjuk meg. A hardveres megoldás során a későbbiekben szeretném a *delta* állítását fizikailag egy potméterrel is lehetővé tenni a felhasználó számára, mivel bizonyos helyzetekben nagyobb amplitúdójú kézmozdulatok felismerésére lehet szükség, más helyzetekben viszont kényelmesebb, ha már kisebb mozgásra is aktiválódik egy parancs (például ülve kényelmesebb kisebb karmozdulatot tenni, állva pedig nagyobbat, álló testhelyzetben sokkal inkább elképzelhető olyan szabadabb mozgása a karnak, amit a szándékos gesztusnyilvánítástól el kell különítenünk). A későbbi fejlesztések során lehetségesnek tartom azt is, hogy a *deltát* dinamikusan válassza meg a program, azaz ha kisebb amplitúdójú mozdulatokat kezd el a felhasználó használni, akkor a felismerés képes legyen ehhez adaptálódni, és fordítva. Statikus változó továbbá az adott tengelyre vonatkoztatott nyugalmi gyorsulás értéke. Erre egyszerűen csak *average* elnevezéssel fogok hivatkozni, mivel meghatározni is legegyszerűbben a nyugalmi gyorsulásértékek átlagolásával lehet. Ennek értékkészlete az eszköz nyugalmi pozíciójától függően -1 g és 1 g közötti érték lehet, mely egyszerűen megállapítható akár méréssel, attól függően hogy a felhasználó a karjára rögzített eszközt milyen pozícióban tartja, mielőtt a gesztus megrajzolását elkezdené. A további fejlesztések során ezt a változót is célszerű lehet dinamikusan változóvá tenni, ez ugyanis lehetővé tenné a felhasználó számára, hogy kényelmesebben tudjon bizonyos gesztusokat megrajzolni. Egy vízszintes félkört (pl. U) kényelmesebb a csukló olyan helyzetéből indulva megrajzolni, amikor ha kinyújtjuk a kezünket, az ujjaink vízszintesen helyezkednek el egymás mellett, egy függőleges félkört (pl. C) pedig úgy, hogy kinyújtott kezünkön az ujjaink függőlegesen helyezkednek el egymás alatt. Érdemes egy **logikai** változót bevezetni, mely egy egyszerű választással legyen például *lookForMax*. Amennyiben *lookForMax* igaz, a maximumkeresés fázisában vagyunk, ellenkező esetben pedig a minimumkeresés fázisában, más állapot a 4-3.1.1. pont alapján nem lehetséges. Szükségünk van továbbá **dinamikusan változó**, megfigyelő-típusú változókra a *lehetséges minimum* ill. a *lehetséges maximum* eltárolására. Egy beolvasott adatot az időbeli előfordulása és értéke határoz meg, ezért ezen megfigyelő-változók egy-egy változópárt jelentenek. Mindezek után a csúcsdetektációhoz szükséges változókat a következő (4-3.) táblázatban foglalom össze:

mx	Lehetséges maximum értéke
mxPos	Lehetséges maximum előfordulásának ideje
mn	Lehetséges minimum értéke
mnPos	Lehetséges minimum előfordulásának ideje
lookForMax	Maximumkeresés/minimumkeresés
delta	Érzékenységi tényező
average	Tengelyre vonatkoztatott nyugalmi gyorsulás

4-3. táblázat

4-3.3.2. Kezdeti feltételek

Mihelyst van egy csúcsunk, onnantól kezdve a csúcsdetektálás ezen megfontolások alapján jól működik, a kezdeti feltételek megadása azonban nem egyszerű. Az alkalmazott megoldás: legyen kezdetben *mx* a mínusz végtelenben, *mn* a plusz végtelenben. Így az első tényleges mintavett adat egyből *lehetséges minimummá* és egyben *lehetséges maximummá* lép elő (a korábban megfigyelt *lehetséges minimumnál* ugye alacsonyabban, a korábban megfigyelt *lehetséges maximumnál* pedig magasabban van). Azonban, még mindig el kell dönteni, hogy minimumot vagy maximumot keresünk-e először. Bármelyik döntés lehet jó vagy rossz, attól függően, hogy milyen jelalak érkezik. Ennek sajnos van egy hátránya: ha a gesztusmintázat amit rögzítünk pl. detektálandó minimummal kezdődik, előfordulhat, hogy ezt a minimumot azért nem detektálom, mert alapértelmezésben maximumdetektálásra állítom a programot. Persze, utána minden jól működik, csupán indításkor egy csúcsot elvesztek. Ezt követően már jól fog működni a detektálás, többször ilyen nem fordul elő újra, ez mégis problémát jelenthet. A csúcsok eltárolásánál a főciklus-számláló értéke fogja a csúcs időbeli előfordulásának helyét megadni. Mire „elfogy” a főciklusszámláló, az lehet egy hosszabb idő, pl. unsigned long használata esetén, de elegáns lenne, ha meg lehetne oldani mindezt úgy, hogy bizonyos ideig tartó tétlenség észlelése után újrakezdődjön a számlálás, és minden „resetelődjön”. Ez azonban az algoritmus ezen hiányossága miatt nem lehetséges, mivel az első csúcsot téveszthetí. Az unsigned long persze nagyon sokára fogy el, 5 msec-os mintavételezéssel számolva kb. 248 nap alatt. Ami a szűk keresztmetszetet jelentheti, az a fogyasztás, hiszen ha állandóan bekapcsolva kell tartanunk az eszközt, hogy mindig jól működjön, akkor az akku nem fogja bírni túl sokáig, ezért az algoritmus ezen hiányosságának kiküszöbölése még várat magára a későbbi fejlesztések során.

4-4. A csúcsdetektálás párhuzamosítása

Az eddigiekben az Eli Billauer által megvalósított programot mutattam be, a továbbiakban a saját fejlesztések leírása következik.
Mivel a gesztusfelismerés szempontjából valamely síkot szeretnénk kitüntetettként kezelni, ezért egyszerre párhuzamosan a gyorsulásmérő két tengelyén kell a csúcsdetektáló algoritmust futtatnunk. Ez a 4-1. ábrán is látható volt. A csúcsdetektációt ha úgy tekintjük, hogy a fő ciklus egyszeri lefutásakor egy minta vételezése történik meg, akkor könnyen tudjuk úgy párhuzamosítani két tengelyre a detektálást, hogy a fő ciklus egyszeri lefutásakor egyszer a gyorsulásmérő egyik tengelyről veszünk mintát, és elvégezzük a detektálást, majd ezt elvégezzük a másik tengelyről beolvasott mintával is. Egy adott tengelyen végzett csúcsdetektáláshoz szükséges változókat a következő táblázatban foglaltam össze. Erre a táblázatra asszociatív tömbként tekinthetünk, mely tömb egy adott tengelyre vonatkoztatott csúcsdetektáláshoz szükséges változókat tartalmazza. Mivel (legalább) két tengelyünk van, ezért ezt az asszociatív tömböt az egyes tengelyekhez hozzárendelve egy asszociatív mátrixot lehet definiálni, mely a következőképpen néz ki:

x-tengely	y-tengely	z-tengely
mx	**mx**	**mx**
mxPos	**mxPos**	**mxPos**
mn	**mn**	**mn**

mnPos	**mnPos**	**mnPos**
lookForMax	**lookForMax**	**lookForMax**
delta	**delta**	**delta**
average	**average**	**average**

4-4. táblázat

A táblázat azonos soraiban, de különböző oszlopaiban szereplő értékek egymástól függetlenek, a *delta* esetén azonban célszerű ugyanazt beállítani mindhárom tengely esetén. Az *average* tengelyenként egyszerűen megállapítható, jellemzően például valamelyik tengely a nehézségi gyorsulás irányába mutat 1 g nehézségi gyorsulással, a másik kettő pedig ezen irányra merőleges, 0 nyugalmi gyorsulással, persze ez a nyugalmi pozíciótól függően általában ettől valamelyest eltérő érték.
Ezzel a tömbösítéssel a módszer egyszerűen alkalmazható egyidejűleg 3 tengelyen végzett elemzésekre is, azonban a továbbiakban 2 tengelyt fogok használni, mivel egy kitüntetett síkban végzett vizsgálatokról lesz szó. Az általánosítás kézenfekvő 3 tengely esetére.
Legyen a vizsgálatunk szempontjából kitüntetett sík két tengelye az x- és az y-tengely.
A 4-1. ábrán látható vázlaton a csúcsdetektáció egy adott tengelyre való egyszeri meghívása tehát az itt szereplő asszociatív mátrix adott tengelynek megfelelő oszlopában szereplő változók átadását igényli. Ezzel a módszerrel nem keverednek össze a különböző tengelyekhez tartozó változók.
A csúcsokat a detektációt követően el is kell tárolni. Mivel a valósidejű feldolgozás a cél, ezért célszerű mindig csak a legutolsó *n* beérkezett csúcsot figyelembe venni. Ehhez egy *n* hosszú tárolóba helyezem az utoljára beérkezett csúcsok adatait. A *csúcs* 4-1.3 pontbeli definiálása alapján egy csúcs meghatározásához tudnunk kell egyrészt annak nagyságát. Ezt a nagyságot az adott tengelyre vonatkoztatott nyugalmi gyorsulástól való abszolút különbség jelenti. Tudnunk kell, hogy a csúcs időben mikor következett be, ezt egy a program elején elindított számláló, a korábban említett főciklus-számláló tudja megadni, melynek értékét a detektálás esetén jegyzem fel. A nagyságon és időbeliségen kívül végül tudnunk kell, hogy melyik tengelyen, és hogy minimumként vagy maximumként detektáltuk-e a szóbanforgó csúcsot.
A tároló a következőképpen képzelhető el:

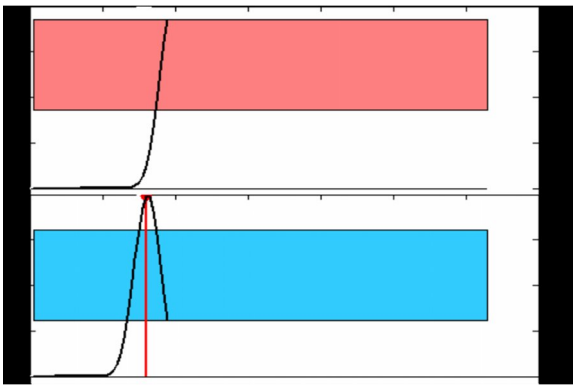
1. csúcs	2. csúcs	3. csúcs	…	n. csúcs
előfordulási idő	**előfordulási idő**	**előfordulási idő**	**…**	**előfordulási idő**
típus	**típus**	**típus**	**…**	**típus**
absz. differencia	**absz. differencia**	**absz. differencia**	**…**	**absz. differencia**

4-5. táblázat

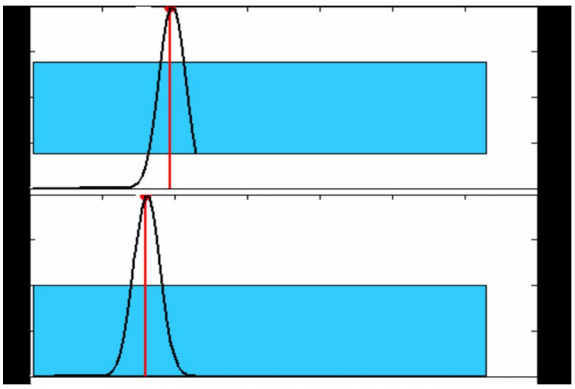
Első ránézésre azt gondolhatnánk, hogy ha az éppen detektált csúcsokat sorban írnánk be a tároló következő helyére, akkor folytonosan kerülnek be az adatok, ez azonban nem feltétlenül van így.
A következő részhet a 4-3. és 4-4. animáció[7] tartozik. A következő ábrákon láthatjuk, hogy a 4-3. animáció esetében két tengelyen végzett párhuzamos elemzést mutatok be, ahol először

[7] http://szucher.com/tdk2011/4-3.avi | http://szucher.com/tdk2011/4-4.avi

az egyik tengelyen, majd röviddel utána a másik tengelyen észlelünk egy csúcsot (4-3-a és 4-3-b ábra). Látható, hogy a két csúcsot egymás után detektáltunk.

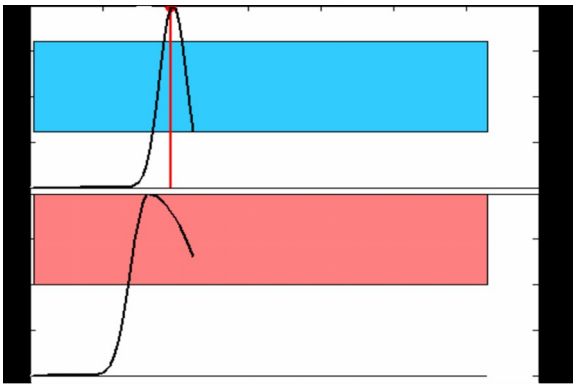


4-3-a ábra

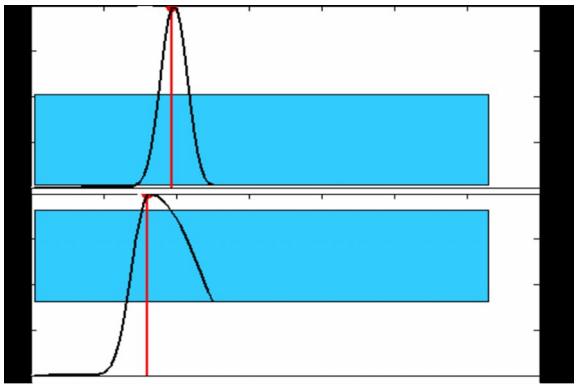


4-3-b ábra

Mint azt már korábban is fontosnak tartottam kiemelni, a csúcsok észlelése a csúcs előfordulásának ideje *után* következik be, és hogy mennyivel utána, azt a *delta*-sávban eltöltött idő határozza meg. A 4-4. animáción egy olyan esetet mutatok be, ahol szintén két egymást követő csúccsal van dolgunk, azonban előbb történik meg egy időben később előforduló csúcs detektálása, mint az azt követő csúcsé. Az animáción látható, hogy mivel az alsó ábrán - időben korábbi - csúcs lecsengése sokkal lassabb, mint a felsőn - az időben későbbié -, ezért a felső csúcsot követően a *deltával* lejjebb kerülés (a *delta*-sávból való alulról kilépés) előbb fog bekövetkezni (4-4-a ábra), mint az alsó csúcs esetében (4-4-b ábra).



4-4-a ábra



4-4-b ábra

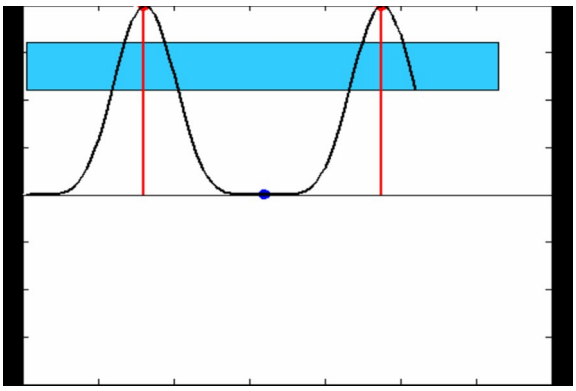
Tehát azt, hogy az *n* elemű tárolónkba kronológikus sorrendben kerüljenek be a csúcsok, biztosítani kell. Ehhez meg kell vizsgálni, hogy egy újonnan eltárolni kívánt csúcs valóban időben később következett-e be, mint az őt megelőző csúcs. A tárolás során tehát egy csúcs detektációja után összehasonlítjuk a tárolónkban szereplő elemekkel, és sorrendhelyes címzéssel szúrjuk be abba. A tárolót periodikusan frissíteni, újraírni kell minden egyes új csúcs érkeztével. Mindezek megoldása után következhet a tárolóban szereplő csúcsok feldolgozása.

4-5. Csúcsfeldolgozás

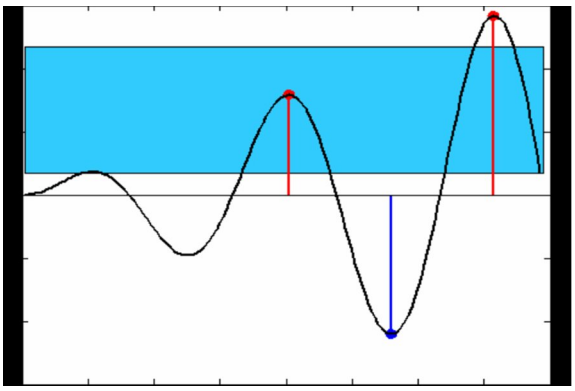
Ebben a szakaszban a 4-1. ábrán zölddel jelölt, csúcsfeldolgozó szubrutint mutatom be. A csúcsokon végzett elemzések minden újonnan detektált csúcs esetében lefutnak. Először egy általános problémát tekintek át, majd miután ennek megoldását bemutatom, a továbbiakban a különböző alakú gesztusok elkülönítését végző, alakzatspecifikus szűrőket mutatom be. Elkülönítek szögletes, görbe és ferde vonalú gesztusmintázatokat, a további pontokban az egyes geometriai mintázatokra jellemző szabályosságokról fogok írni.

4-5.1. Csúcsdetektálás általános problémája és megoldása

Amint a 4-4. pontban leírtam, a beérkezett csúcsok adatai egy periodikusan frissülő tárolóba kerülnek. A 4-3.1 pontban leírtak alapján a maximumkeresés és a minimumkeresés mindig egymást követik. Felmerül azonban az a probléma, hogy előfordulhat olyan mintasorozat, amikor tudjuk, hogy pl. két maximum követi egymást, azonban az algoritmus ebben az esetben is detektál egy minimumot közöttük. A 4-5. animación[8] bemutatott esetben (4-5-a ábra) a jelalak összekeverhető egy olyan csúcsmintázattal, ahol ténylegesen van egy minimum a maximumok között (emlékezzünk vissza a 4-2 animációra, ezt idézi fel a 4-2-g ábra).



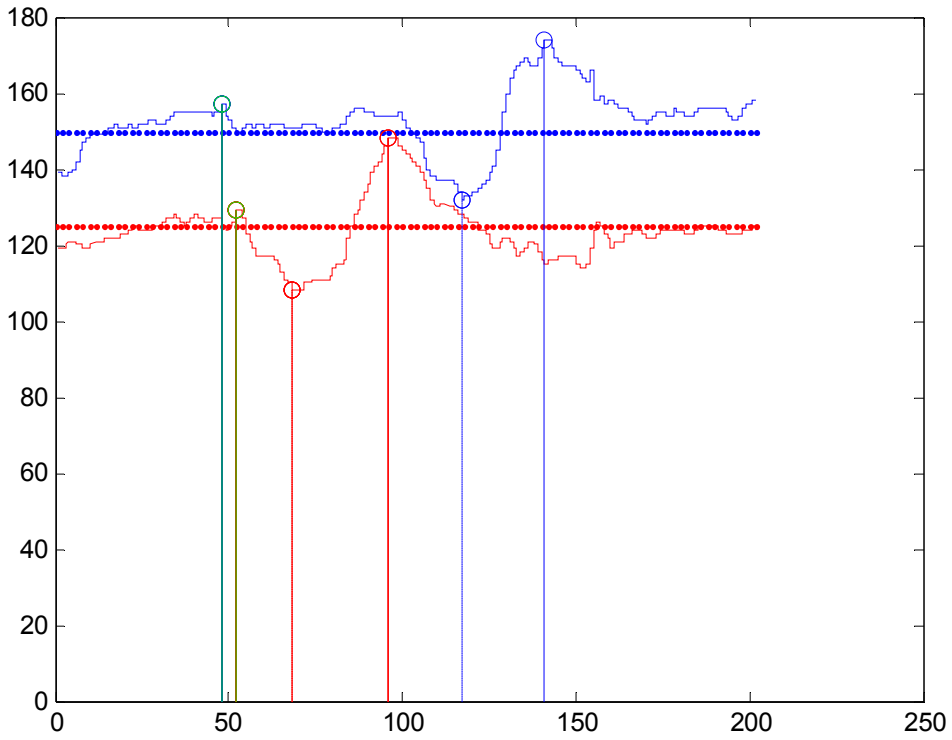
4-5-a ábra



4-2-g ábra

Látható, hogy a két csúcs közötti völgy alja (4-5. ábra) noha matematikailag minimumnak tekintendő, de a 3. fejezetben leírtak miatt a gyorsulásértékekből másfajta mozdulat következtethető vissza attól függően, hogy van-e minimum a maximumok között.
Azt, hogy az ilyen hamis minimumokat ne vegyünk számításba, biztosítani kell egy szűrővel. Ez a szűrő nem tesz mást, mint minden olyan csúcsot semmisnek vesz, mely a tengelyre vonatkoztatott nyugalmi gyorsuláshoz közel van (a tőle vett távolság abszolútértéke nem halad meg egy küszöbértéket). A 4-6. ábrán a gesztus-adatbázis 9. gesztusához tartozó valamely mintára elvégzett csúcsdetektációt követően ezen szűrőt futtatva a zölddel jelölt csúcsok kiszűrésre kerültek, és az elmozdulás 4 jellegzetes csúcsából (a 3-1.1. pontban leírtak alapján) a gesztust detektáltam.

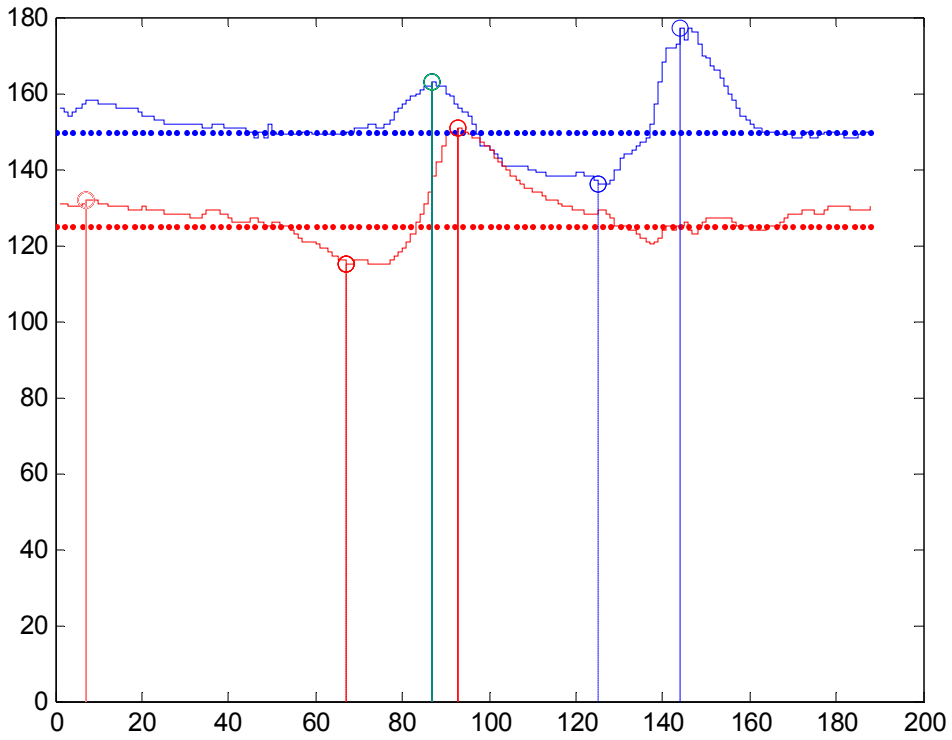
[8] http://szucher.com/tdk2011/4-5.avi



4-6. ábra

4-5.2. Szögletes gesztusok felismerése

Ahogy a 3. fejezetben leírtam, a vizsgálati sík origójához képest a négy égtájnak megfelelő irányba vett mozdulatokat tekintem a szögletes típusú gesztusok alap-építőkockáinak.
Ezen építőkockák egymás után való alkalmazása komplexebb gesztusokat jelenthet.
Két tengely esetén a minimumokra és a maximumokra egyszerű jelöléseket vezethetünk be a kódolás szempontjából, a korábban leírtak analógiájára. Mindezekről együttesen elmondható, hogy nem keverednek össze a csúcsok, egy időben egy tengelyen valamilyen elmozdítást egy minimum-maximum csúcs-pár jelez, ezek sorrendjéből az irány egyértelműsíthető. Mivel csúcs-párok nem keveredhetnek össze, ezért elég egy egyszerű szabályrendszert felállítani: akkor detektálunk egy elmozdulást, ha olyan csúcsmintázatot látunk, ahol csak azonos tengelyhez tartozó csúcspárok vannak (például egy L-alakú gesztus esetén két pár). Korábban említettem, hogy legalább 2 mozdulatból álló gesztusokat célszerű elkülöníteni, ezért a minimum 2 csúcspárt megkövetelem, tehát legalább 4 csúcs meglétét írom elő egy gesztus detektálásához. A 4-6. ábrán is ilyen gesztus volt látható.
A gesztusok ilyen jellegű felismertetése esetén egy további probléma merül fel. Mivel ugyan idealizált esetben egy időben csak az egyik tengelyen történik elmozdulás, de már a tengely irányához képest minimális eltérés esetén is megjelenik egy csúcs közel egyidőben a másik tengelyen is, melyet szintén detektál a csúcsdetektáló. Ebben az esetben megvizsgálom, hogy a két közeli csúcs közül melyik a domináns, ez azt jelenti, hogy a nyugalmi gyorsulás értékétől vett abszolút eltérésüket összehasonlítom, és ha az egyik legalább másfélszerese a másiknak, akkor dominánsként fogadom el, és a másik csúcs adatát egyszerűen eldobom. Ezt a 4-7. ábrán mutatom be, a gesztus-adatbázis 9. gesztusának egy mintáján (a kiszűrt csúcs zölddel).



4-7. ábra

4-5.3. Ferde gesztusok felismerése

Az előző alpont konklúzióját alapul véve, amennyiben időben nagyon közel, szinte egyidőben két csúcsot detektáltunk két tengelyen, és egyik csúcs sem domináns a másikkal szemben olyan mértékben, hogy egyenes elmozdulásról lehessen beszélni, akkor ebben az esetben ferde elmozdulásról lehet szó. A ferde elmozdulást a dolgozat írásakor még nem tudtam elég jól elkülöníteni az egyenes gesztusoktól, részben a rendelkezésre álló adatfolyamon korlátozott felbontású minták miatt (mind A/D átalakítás bitszámát tekintve, mind mintavételezés gyakoriságát tekintve), ezért egyelőre nehezen tudtam csak elkülöníteni az ilyen mintákat a szögletes elmozduláshoz tartozó mintáktól.

4-5.4. Görbevonalú gesztusok felismerése

Amint az a 3. fejezet 3-2. táblázatában is látszott, a görbevonalú gesztusok a szögletes gesztusoktól teljes mértékben elhatárolhatóak, mivel itt nemhogy egy adott tengelyre vonatkozó minimum-maximum párok nincsenek, de ha megnézzük a szabályosságot, épp azt lehet előírni, hogy egy adott tengelyen soha nem követi egymást két csúcs, mindig egymásba fonódnak a csúcsok. Megfigyelhető továbbá, hogy egy adott tengelyen mindig minimumot maximum követ (egy a másik tengelyen található közbeékelt csúcs után), valamint fordítva: maximumot mindig minimum követ (megjegyzem, jelen megállapítás teljesen független a csúcsdetektáció egyik alapelvétől). Ezen szabályosság lekódolása teljesen diszjunkttá teszi a görbevonalú és a szögletes gesztusok felismerését.

4-5.5. Alakzatspecifikus szűrők

Az előző alpontokban leírtak megvalósításához szűrőkre volt szükségem. Ilyen szűrő egyrészt a 4-5.1. pontban leírt problémát megoldó, nyugalmi gyorsuláshoz közel lévő csúcsokat kiszűrő szubrutin. Ez a szűrő tekinthető a csúcsok vertikális (nagyság) adatait vizsgáló szűrőnek. Ami a horizontális (időbeli) szűrőket illeti, figyelni kell a beérkezett csúcsok közötti időbeli távolságot, ha ez egy megadott maximális küszöbértéket meghalad, akkor szegmentálni kell a gesztusokat, "új gesztus kezdődött". Amennyiben egy másik, minimális küszöbértéket nem lép át, akkor két beérkezett csúcsot egyidejűleg detektáltnak kell tekintenünk, itt két lehetőség van: az egyik csúcs domináns a másikhoz képest, ekkor a dominánsat megtartom, a másik adatait eldobom, vagy ha közel azonosak, akkor ferde elmozdulást tételezek fel. A gesztusfelismerésre hivatott szűrőkhöz tartozó, kalibrálható paramétereket végezetül a 4-6. táblázatban foglaltam össze.

abszolút differencia limit	az az eltérés a nyugalmi gyorsulástól, ami fölött egy csúcsot el lehet fogadni
szegmentációs limit	az az időbeli távolság, amely időnél ha több telik el két beérkező csúcs között, akkor már úgy értelmezem, következő gesztus kezdődött
közelségi limit	az az időbeli távolság, amely időnél ha kevesebb telik el két csúcs észlelése között, akkor egyidejűnek tekintem őket
dominancia faktor	az az arány, amely meghatározza, hogy minimum mennyivel nagyobbnak kell lenni két azonos idejű csúcs közül az egyiknek ahhoz, hogy domináns legyen a másikhoz képest

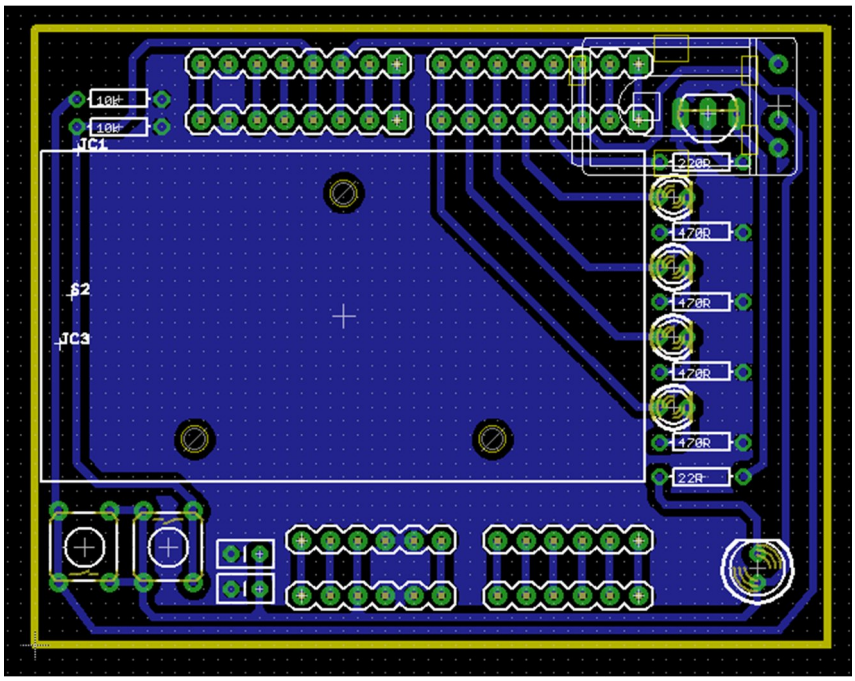
4-6. táblázat

5. Hardveres fejlesztés

Ebben a fejezetben áttekintem a megépített hardverre vonatkozó néhány fejlesztési lépést. A 4. fejezetben leírtak alapján a fejlesztést párhuzamosan MATLAB és Arduino környezetben végeztem.

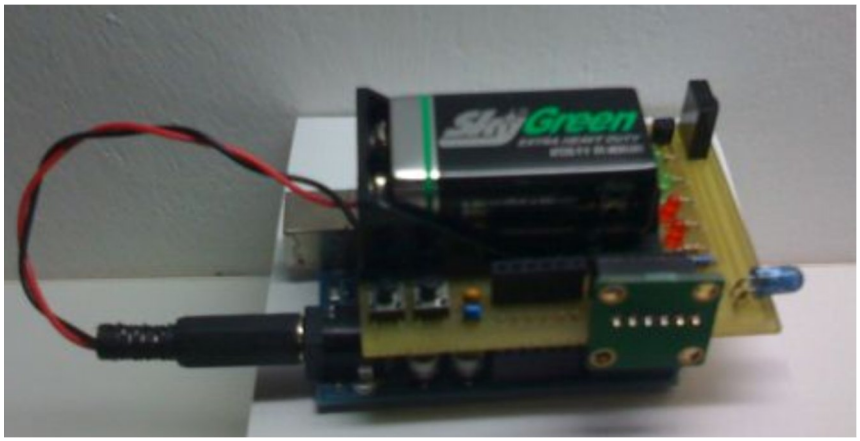
5-1. Első prototípus

Az Arduino egy nyílt forráskódú rendszer, mely az Atmel cég Atmega328 típusú mikrokontrollerén[9] alapul, mely tökéletesen alkalmas egyszerű áramkör-prototípusok elkészítésére. Az általam használt, Duemilanove[10] típusú Arduino egy önálló mikrokontrollerhez képest annyival több, hogy az alkalmazást megkönnyítő alapvető kiegészítő-áramköri elemeket tartalmazza (tápfeszültség-szabályozás ill. védelem, kvarckristály, FTDI chip az USB-n keresztüli felprogramozáshoz és soros kommunikációhoz). Az alkalmazott gyorsulásmérő az Analog Devices cég ADXL335-ös gyorsulásmérője[11], fejlesztőpanel kivitelben. Mivel a kézmozdulatokat a levegőben végzem, ezért szükség volt a tápellátás biztosítására, ehhez az Arduino alapáramköréhez rögzíthethető kiegészítő panelt készítettem, amelyre a tápellátást biztosító 9V-os elemtartó részen kívül tesztelési célokran elhelyeztem ledeket, kapcsolókat, egy infra ledet és egy infra vevőáramkört. Ez a kompakt eszköz szolgáltatta az alábbi képen látható első prototípust, melyen az algoritmust fejlesztettem. A kiegészítő panelláramkör layoutja, melyet a CADSoft cég EAGLE programjával terveztem az 5-1. ábrán látható, majd ezen panellel kiegészített Ardunio-prototípus az 5-2. ábrán látható.



5-1. ábra

[9] http://www.atmel.com/dyn/products/product_card.asp?part_id=4720
[10] http://arduino.cc/en/Main/arduinoBoardDuemilanove
[11] http://www.analog.com/en/mems-sensors/inertial-sensors/adxl335/products/product.html

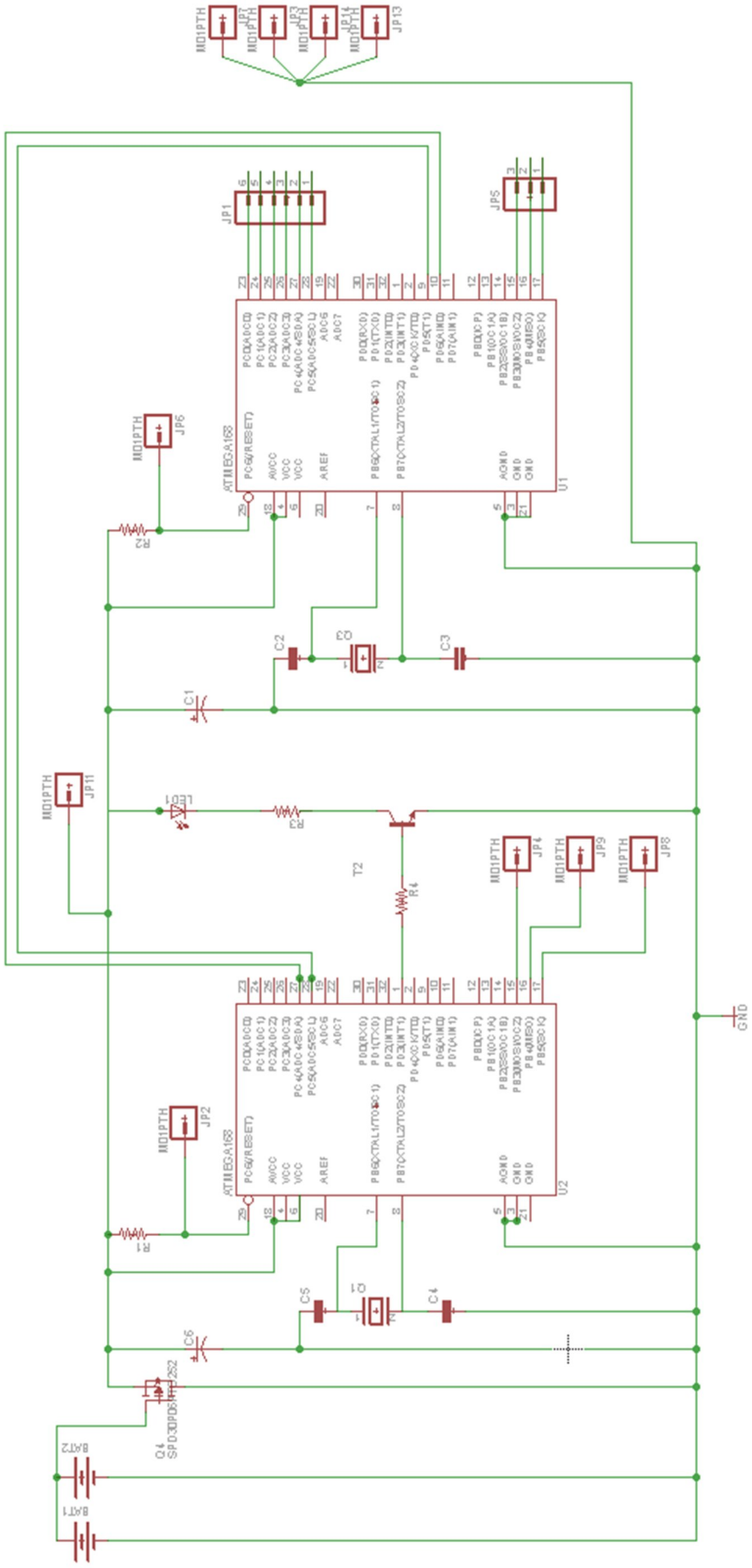


5-2. ábra

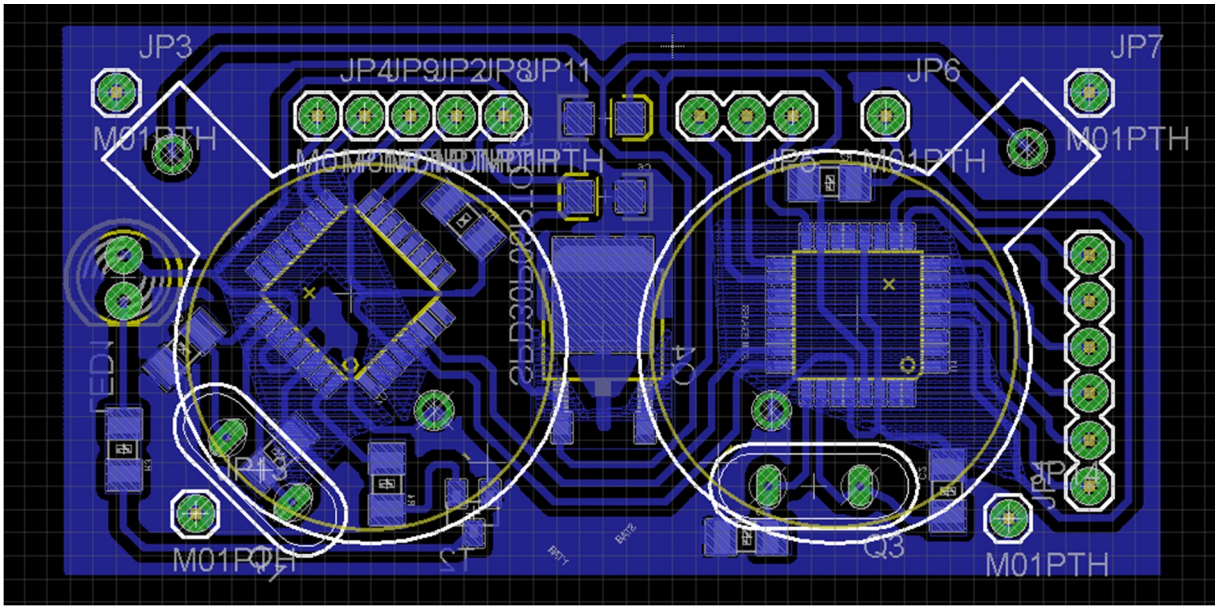
A használt analóg gyorsulásmérő-értékeket a mikrokontroller beépített, 10 bites A/D-átalakítójával olvassa be. A gesztus-adatbázisban alkalmazott 8 biten kvantált adatokhoz képest így 4-szer pontosabb függőleges felbontást kaptam. A mintavételezést is gyorsabban, kb. 5 milliszekundumonként végeztem, de a programkód futásától függően ennél lassabban történik a tényleges mintavételezés, ami még mindig nem elég gyors a vízszintes felbontás pontosságának szempontjából.
Alapvető célom volt, hogy a két mintavételezést között lefutó programkód minél egyszerűbb legyen, hogy minél gyorsabban tudjon lefutni, ezáltal minél pontosabb adatsoron téve lehetővé a valós idejű elemzéseket.
Igyekeztem szoftveresen a lehető legegyszerűbb és leggyorsabb kódot megírni, de szeretném ennél is jobban megnövelni a pontosságot, ami kritikus tényezője a komplexebb gesztusok felismerésének.

5-2. Karóra méretű változatok

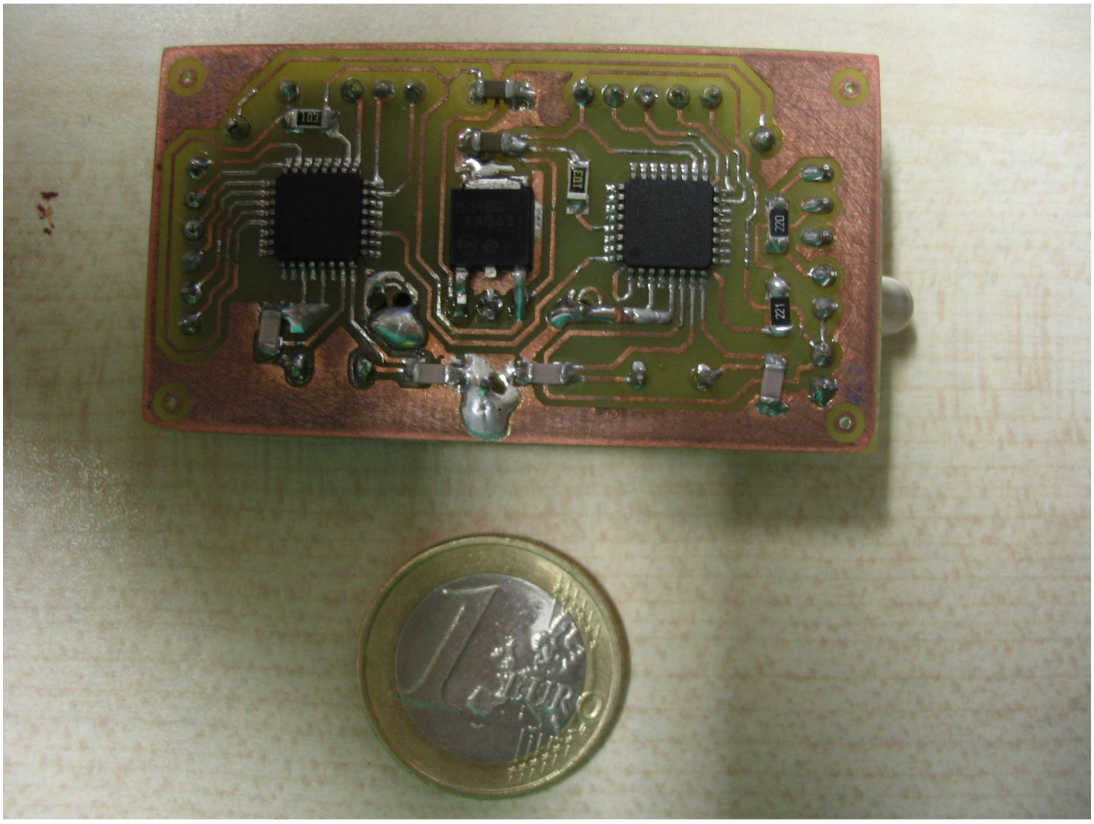
Az algoritmus működését megfigyelve láthatjuk, hogy az adatfeldolgozás lényegében a detektált csúcsokon történik. Mivel csak a beolvasott mintaértékek töredékéből lesz csak csúcs, ezért célszerűnek adódik a feldolgozás hardveres párhuzamosítása: legyen egy mikrokontroller, ami csak a csúcsdetektációt végzi a beérkező adatfolyamon, és ezek adatait továbbítsa egy másik mikrokontroller felé, ami ezen csúcsokon elvégzi a feldolgozó műveleteket. Ez azt jelenti, hogy két mintavétel között nem kell az egész kódnak lefutnia, csak a csúcsdetektáló kódnak, két csúcs detektálása között pedig kényelmesen elvégezheti a másik mikrokontroller a detektált csúcsokon az elemzést. Ezen ötlet megvalósításához az Arduinoban alkalmazott mikrokontrolleren alapuló áramkört fejlesztettem, karóra méretben. Ennek persze az az ára, hogy el kellett hagyjak minden kényelmi és extra elemet, ami a méretet növelhetné, tehát külső pl. külső programozó használata vált szükségessé. Az alkalmazott vezetéknélküli átviteli megoldás a kis erőforrásigénye miatt az infrára esett, ez azonban a jelenlegi tesztelések során rossznak bizonyult, ezért – noha a panel képes az önálló jelfeldolgozásra – de vezeték nélkül egyelőre nem tudja elküldeni a detektált gesztushoz tartozó kódot.
A jelenlegi kapcsolás és áramköri terv az 5-3. és 5-4. ábrákon láthatóak, egy korábban készült verzió megépítve pedig az 5-5. ábrán látható.



5-3. ábra



5-4. ábra



5-5. ábra

Az áramkörök elkészítése a Schönherz Elektronikai Műhelyben valósult meg.

6. Összefoglalás

6-1. Eredmények

A gesztus-adatbázis mintái közül korábban említett megfontolásokból a 2 hosszú szögletes gesztusokat tartom legcélszerűbbnek elemezni, mivel az 1 hosszúak könnyen összekeverhetőek nem szándékolt mozdulatokkal, 3 vagy 4 hosszú gesztusok esetén pedig jelentősen romlik a felismerés.
2 hosszú „sarkokból” a gesztus-adatbázisban kétféle szerepel, de ezekből nyolcféle irány képzelhető el.
Görbevonalú gesztusok alap-építőkockája a félkör. A gesztus-adatbázisban háromféle szerepel belőlük, de nyolcféle irány képzelhető el velük. A 6-1. táblázatban egy átfogó teszt eredménye látható 10 felhasználó 100 mintájára, az egyes cellák esetén. Az oszlopfő a keresett gesztust mutatja, a sorokban az látható, hogy hány esetben melyik gesztust detektálta az algoritmus a keresés közben. Látható, hogy a főátlóban jók az eredmények.

	9. right down	10. left down	17. C	18. left arch	19. right arch
9. right down	**42/100**	**0/100**	**0/100**	**0/100**	**0/100**
10. left down	**0/100**	**57/100**	**2/100**	**1/100**	**0/100**
17. C	**6/100**	**5/100**	**39/100**	**2/100**	**0/100**
18. left arch	**6/100**	**1/100**	**12/100**	**36/100**	**1/100**
19. right arch	**1/100**	**2/100**	**0/100**	**0/100**	**49/100**

6-1. táblázat

Ezek a félkörök és a „sarkok”. Egy „sarok” építőkockája egy négyzetnek, egy félkör egy körnek, ezért nem megfelelő szegmentálás esetén ezek könnyen összekeverhetőek, ezért csak az alap-építőkockákat tudtam jól felismertetni. Ezen javíthatna gyorsabb mintavételezés.
Általánosan elmondható, hogy

- az adatbázisban alkalmazott mintavételezés alacsony, ennél sűrűbb mintavételezésre van szükség, az A/D átalakító felbontása szintén nem elég
- a felhasználók nagy többsége nem ügyelt a gesztusok precíz végrehajtására, ez a módszerem esetén viszont kritikus kérdés

Megvizsgáltam jobban a félkörökre kapott eredményeket, ennek részletes kifejtése felhasználókra bontva a 6-2. táblázatban található. A *delta* érzékenységi paraméter és a tengelyekre vonatkoztatott nyugalmi gyorsulásértékek állandóak.

felhasználó	2	4	10	17	20	25	31	33	34	35	
delta	11	11	11	11	11	11	11	11	11	11	
x átlag	152	152	152	152	152	152	152	152	152	152	
z átlag	124	124	124	124	124	124	124	124	124	124	
gesztus: 19.	7/10	4/10	4/10	4/10	4/10	9/10	5/10	4/10	0/10	8/10	sum 49%
gesztus: 18.	8/10	2/10	3/10	1/10	5/10	4/10	3/10	2/10	1/10	7/10	sum 36%
gesztus: 17.	6/10	0/10	5/10	2/10	8/10	4/10	1/10	3/10	3/10	7/10	sum 39%
három félkörre átlag	sum 70%	sum 20%	sum 40%	sum 23%	sum 57%	sum 57%	sum 30%	sum 30%	sum 40%	sum 73%	sumsum **44,0%**

6-2. táblázat

A 4-3.3.1. szakaszban már írtam arról, hogy a *delta* paraméter állítását fizikailag egy potenciométerrel is érdemes lenne lehetővé tenni, mert a felhasználók egyénenként eltérő lendülettel rajzolták a gesztusokat. Ugyanígy más-más pozícióban tartották a kezüket a rajzoláskor, és erre a nyugalmi gyorsulás érzékeny.
Ésszerű lehet a felhasználót megkérni, hogy használat előtt kalibrálást végezzen, ami abból áll, hogy a számára kényelmes pozícióban tartja egy ideig a kezét (ezzel beállítható a nyugalmi gyorsulás), valamit a *delta* állítását maga végzi. A 6-2. táblázatban szereplő mintákra felhasználófüggő beállításokat alkalmaztam, ami azt jelenti, hogy az intenzívebben rajzoló felhasználók mintáihoz nagyobb *deltát* állítottam be (manuálisan), és megfigyeltem, mi a nyugalmi gyorsulás kezdeti értéke gesztusaik rajzolását megelőzően. Az eredmények javulása ennek megfelelően megfigyelhető a 6-3. táblázatban.

felhasználó	2	4	10	17	20	25	31	33	34	35	
delta	10	12	20	5	15	15	8	8	8	8	
x átlag	152	152	152	152	152	152	152	152	152	152	
z átlag	120	124	120	122	122	127	127	127	127	127	
gesztus: 19.	8/10	9/10	9/10	6/10	6/10	9/10	8/10	6/10	4/10	9/10	sum 74%
gesztus: 18.	6/10	4/10	4/10	5/10	7/10	6/10	5/10	4/10	2/10	8/10	sum 51%
gesztus: 17.	6/10	0/10	3/10	6/10	9/10	6/10	3/10	5/10	6/10	8/10	sum 53%
három félkörre átlag	sum 66%	sum 43%	sum 53%	sum 56%	sum 73%	sum 70%	sum 53%	sum 50%	sum 40%	sum 83%	sumsum **58,7%**

6-3. táblázat

Ezzel tehát hangsúlyozom a kalibrálás szerepét. Saját magamra elvégezve a kalibrációt, a demonstrációs videóban[12] a létrehozott eszközzel bemutatom, hogyan ismertetem fel a prototipussal a nyolcféle félkör ill. a nyolcféle „sarok” alakzatot.

6-2. Áttekintés

Összességében elmondható, hogy a munkám során létrehoztam egy algoritmust, mely személyfüggetlen gesztusfelismerésre képes. Az adatbányászati értelemben vett tanító mintákra nincs szükség, ugyanakkor egyénekre elvégzett kalibráció jelentős mértékben javít a felismerési eredményeken. Az algoritmus alapján MATLAB környezetben írtam egy programkódot, és egy előre rögzített adatbázison mérések eredményeként leszűkítettem a lehetséges gesztusok körét egy 16-elemű halmazra, melyen a felismerés a lehető legjobban tud működni. Az elkészült kódot egy mikrokontrolleres rendszerbe ültetve egy saját prototípust hoztam létre, melyen valós idejű teszteket tudtam futtatni, és létrehozni egy olyan programkódot, ami a demonstrációs videón bemutatott módon és ott látható válaszidővel illetve hatékonysággal tud működni. Létrehoztam egy ténylegesen karóra méretű eszközt, mely jelen munka leadásakor fejlesztés alatt áll. A videón láthatóan a „vezetékes” prototípust alkalmazom, mert a vezetéknélkül működő verzió esetében

- infra átvitel esetén könnyen elvesznek parancsok
- 802.11 szabvány alapon működő szinkron átvitel esetén jelentős (2 másodperces) késéssel érkeznek be az adatok, és ez elvonhatja a figyelmet arról, hogy a gesztusfelismerésnek megfelelő parancskiadás ténylegesen valós időben történik

A gesztusfelismeréshez szükséges elméleti alapokat a gyorsulás fizikai modellje alapján dolgoztam ki, a beérkező adatfolyamon egy hatékony tömörítési eljárást mutattam be a csúcsdetektáció alapján. Egy létező csúcsdetektáló algoritmust továbbfejlesztettem, alkalmassá tettem párhuzamos feldolgozásra, és szűrőket vezettem be a valós időben jelentkező problémák megoldására, melyek valós idejű teszteléskor derültek ki.
Munkám célja alapvetően annak bemutatása, hogy lehetséges rendkívül kis méretben is egy viszonylag nagy gesztushalmazt felismerő rendszert megalkotni, anélkül hogy külső adatgyűjtő (kamera) vagy adatfeldolgozó (számítógép) egység legyen mindehhez szükséges. Ez az, amitől a munkám újszerű, és a HCI irányelvei alapján közelebb visz egy minél természetesebb interakciós lehetőség megvalósításához a számítógép és az ember között.

6-3. Future work

Szoftveres szempontból célszerűnek tartom adaptívabbá fejleszteni a jelenlegi algoritmust. Ezalatt a *delta* értékének és a nyugalmi gyorsulás, valamint egyéb szűrőparaméterek dinamikus állításának lehetőségét értem, melyek sokkal hatékonyabbá tehetik a felismerést.
Hardveres szempontból az 5-2. szakaszban felvetett processzoros párhuzamosítás jelenthet nagy előrelépést, mivel sokkal gyorsabb adatfeldolgozást tehet lehetővé. Célszerű lehet egy új adatbázis felvétele, melyen nem csak statikusan rögzített adatok, de folyamatos jelfolyam is elérhető lenne változatos gesztusokkal, ez a valósidejű feldolgozó teszteléséhez jelenthet lehetőséget. A minták továbbá lehetnének sokkal sűrűbbek, nagyobb felbontással sokkal jobb eredményeket lehetne elérni mind A/D-átalakítás, mind mintavételezés tekintetében.

[12] http://szucher.com/tdk2011/demo.wmv

Adatbányászati szempontból ugyan aggodalomra adhat okot a minták sűrűségének többszörözése, viszont az általam alkalmazott csúcsdetektáció módszere esetén arra van szükség, hogy a csúcsok adatait (időbeliség és nagyság) minél pontosabban meg tudjuk mondani, és minden más beolvasott gyorsulásértéket el lehet dobni.
A vezetéknélküli kommunikáció megvalósítása szintén komoly fejlesztési problémákat vet fel, ilyenek a tápellátás biztosítása, a válaszidő minimálisra csökkentése, és a szinkronitás megtartása (ne vesszen el adat).
Szükség van továbbá átfogó teszteléseket végezni különböző mikrokontrollerek és processzorok használatával, mind sebesség mind pontosság tekintetében.
Lehetőség van továbbá saját eszköz fejlesztése mellett létező, gyorsulásmérőt tartalmazó mobil eszközökben, mobil platformon kipróbálni az algoritmust, egyszóval elemezni egy kis-erőforrású kód alkalmazási lehetőségeit mobil környezetben.

Irodalomjegyzék

[1] Zoltán Prekopcsák (2008)
Accelerometer Based Real-Time Gesture Recognition
http://prekopcsak.hu/papers/preko-2008-poster.pdf

[2] Róbert Varga, Zoltán Prekopcsák (2011) *Creating a Database for Objective Comparison of Gesture Recognition Systems* in POSTER 2011: Proceedings of the 15th International Student Conference on Electrical Engineering. Prague, Czech Republic, May 2011.

[3] Makara Csaba: *Testi gesztusok számítógépes felismerése* (2011)
http://dea.unideb.hu/dea/bitstream/2437/105608/1/Diplomamunka_Makara_Csaba_vedett.pdf

[4] Jinjun Rao; Tongyue Gao; Zhenbang Gong; Zhen Jiang;
Low Cost Hand Gesture Learning and Recognition System Based on Hidden Markov Model
Information Science and Engineering (ISISE), 2009 Second International Symposium on
Digital Object Identifier: 10.1109/ISISE.2009.53
Publication Year: 2009 , Page(s): 433 - 438
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5447266

[5] Gesture recognition based on 3D accelerometer for cell phones interaction
Gesture recognition based on 3D accelerometer for cell phones interaction
Zhenyu He; Lianwen Jin; Lixin Zhen; Jiancheng Huang;
Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on
Digital Object Identifier: 10.1109/APCCAS.2008.4745999
Publication Year: 2008 , Page(s): 217 – 220
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4745999

[6] Xu, R.; Zhou, S.; Li, W.;
MEMS Accelerometer Based Non-Specific-User Hand Gesture Recognition
Sensors Journal, IEEE
Volume: PP , Issue: 99
Digital Object Identifier: 10.1109/JSEN.2011.2166953
Publication Year: 2011 , Page(s): 1
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6009159

[7] Jungsoo Kim; Jiasheng He; Lyons, K.; Starner, T.;
The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface
Wearable Computers, 2007 11th IEEE International Symposium on
Digital Object Identifier: 10.1109/ISWC.2007.4373770
Publication Year: 2007 , Page(s): 15 – 22
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4373770

[8] http://billauer.co.il/peakdet.html