

ESZTERHÁZY KÁROLY FŐISKOLA
MATEMATIKAI ÉS INFORMATIKAI INTÉZET

Videófelvételek elemzésére épített statisztikai felmérés

Készítette:

Gregus Tamás

Témavezető:

Dr. Geda Gábor

PhD, főiskolai docens

EGER, 2010

Tartalomjegyzék

1. Bevezetés	3
2. A Motion Capture technika bemutatása	5
2.1. Felhasználás	5
2.2. Passzív markerrel működő rendszerek	6
2.2.1. Aktív markerrel működő rendszerek	6
2.2.2. Egyéb megoldások	7
2.3. A háromszögeléses módszer	8
2.4. Mintavételezési eljárás	9
2.5. Éldetektálás	10
2.5.1. Konvolúció	10
2.5.2. Canney-féle éldetektor	11
2.6. Optical flow	12
2.7. Elterjedt mozgásrögzítő és lekövető rendszerek	13
3. Témához kapcsolódó középiskolások számára előírható feladat	16
3.1. Feladat meghatározása	17
3.2. Célkitűzés	17
3.3. Megoldási menet	17
4. Megoldás	21
4.1. RGB színmodell	21
4.2. A program használata	22
4.3. A lekövetés algoritmus	22
4.4. A program elkészítéséhez használt programozási ismeretek	27
4.5. Felhasználhatóság	27
4.6. Eredmények	29

1. Bevezetés

Napjainkban a közoktatás súlyos gondokkal küzd. Egyik jellemző probléma, hogy a tanulók évről-évre fokozatosan elveszítik érdeklődésüket a természettudományos tantárgyak iránt, ami sajnos a felsőoktatást is elérte. A jelenlegi helyzet több éves folyamat eredményeként alakult ki. A megváltoztatására irányuló törekvések mostanáig csekély eredménnyel jártak. Az ezzel kapcsolatos legátfogóbb minőségügyi törekvésnek a Nemzeti Alaptanterv tekinthető [24],[20],[18],[19],[21],[22],[23],[12],[13],[14][15],[16],[17],ennek bevezetése és többszöri módosítása ellenére sem mutat javulást ezen tantárgyak helyzete. Ezt a folyamatot tényként kell kezelni, amelyről jónéhány publikáció és felmérés számol be, valamint a diákok és hallgatók érdemjegyei is erről tanúskodnak. A fizika tárgy már évek óta a legutolsó helyen szerepel a népszerűségi listán, és sajnos közelítenek hozzá a matematika és az informatika tantárgyak is. Mindezek ellenére van ez így, hogy a '80-as évektől a '90-es évek közepéig az egyik legnépszerűbb tárgy az informatika (számítástechnika) volt.

Nem volna bölcs dolog megvárni azt, hogy az informatika is a matematika és a fizika „sorsára” jusson. A dolgozat, jellegénél fogva nem teszi lehetővé, hogy a problémakört – annak összetettsége miatt – minden aspektusában megvizsgáljuk, ezért csak az informatika tárggyal kapcsolatban vetünk fel egy olyan témát, mely a programozás-oktatással kapcsolatban jobb eredményességgel kecsegtethet. A dolgozatban megfogalmazott probléma aktualitásánál fogva a tanulók nagyobb érdeklődésére tarthat számot, ugyanakkor alkalmas a tananyagban foglalt ismeretek közvetítésére is. A dolgozat egy olyan megoldást mutat be, amely törekszik arra, hogy kicsit szemléletesebben, érdekesebben mutasson be egy olyan módszert, mellyel a tanulók nap, mint nap találkoznak, jelen vannak érdeklődési köreikben, hiszen ők képezik a számítógépes játékok és computeranimációk célcsoportját.

Mozdulatsorok lekövetésére kifejlesztett technikákat a fentiekén kívül is sok más területen alkalmaznak. A különböző területeken való elterjedését az tette lehetővé, hogy a digitálisan rögzített mozdulatsorok számítógépes feldolgozása a korábbi megoldásoknál lényegesen több lehetőséget biztosít, ugyanakkor a technológia fejlődése napjainkra elérhetővé tették ezeket a megoldásokat.

Ezt az eljárást idegen szóval *Motion Capture* technikának, (továbbiakban *MoCap*-nek), digitális mozgásrögzítésnek nevezzük. A dolgozat kitér a technológiára és jelenlegi lehetőségeire, működésére, tulajdonságaira, valamint arra, hogy miképpen lehetne a tanulókhöz közelebb hozni ezeket a kevésbé népszerű tárgyakat.

A korábban megfogalmazott célkitűzések mellett a program lehetőséget biztosít annak bemutatására, hogy ez a metódus hogyan valósítható meg a lehető legjobb módon C# programozási nyelven, ezenkívül magát a .NET fejlesztői környezetet is bemutatva.

2. A Motion Capture technika bemutatása

A módszer lényege, hogy egy vagy több objektum mozgását jellemző pontjai mozgása révén követjük. Ezt úgy érhetjük el, hogy a mozgás sebességének megfelelően egy vagy több kamerával bizonyos időközönként felvételeket készítünk. Így az objektum elmozdulásáról kapott adatokat – igény szerint – egy 2D-s vagy 3D-s modellre vetítik, ami a modellnek a vizsgált alannal megegyező mozgását teszi lehetővé. Ezzel csak és kizárólag a jelenet szereplőinek mozgása kerül rögzítésre. Több digitális kamerával megvalósítható, hogy a jelenetet több pontból rögzítsük, így a végtermékben a nekünk legkedvezőbb nézőpontot használhatjuk. A eljárás akkor végezhető el eredményesen, ha a lekövetés teljes ideje alatt az alanyra helyezett minden egyes kitüntetett pont^{*} helyét legalább két kamera folyamatosan látja, és rögzíteni tudja helyzetüket. A számítógép, mely a méréseket végzi, a felvétel után háromszögeléses módszerrel tudja elhelyezni 3D-s térben a pontokat, vagyis a markerek pozícióit. Abban az esetben, ha a teljes térbeli objektum mozgatsoraira van szükségünk, akkor ajánlott 4 kamerával rögzíteni a felvételt. Ha digitális mozgásrögzítést végzünk, akkor ezt a legkülönbébb módokon meg is tehetjük. Mégis két megoldás a legelterjedtebb. Ezek a megoldások a markerek típusában különböznek jelentősen.

2.1. Felhasználás

A módszert manapság a legkülönbébb helyeken használják. Elég csak a legújabb filmeket, videojátékokat, mozgással kapcsolatos kutatásokat megfigyelni, könnyedén észrevehetjük, hogy mekkora szükség van erre a technológiára.

A számítógépes játékok karaktereinek valódi emberek mozgatai adnak élethű mozgást [1], a filmeket hasonló módon teszik élethűvé, és sportolók teljesítményének, mozgataik helyességének méréséhez kiváló megoldási lehetőséget biztosít. Nem hanyagolható el az orvostudományban sem az alkalmazása [2], hiszen bizonyos sérülések utáni rehabilitációban, mozgászavarok kezelésében és gyógyításában, biomechanika területén is felhasználják a lehetőségeit, például a szív, és az izmok mozgásának méréséhez.

A hadászatban az ellenséges hadmozdulatok felderítésére, akár nagy távolságokban történő mozgások felismerésére és lekövetésére, felderítésre használják eredményesen.

^{*} a pontok megjelölésére különböző technikákat fejlesztettek ki, erre a markerek szolgálnak

2.2. Passzív markerrel működő rendszerek

A passzív markert használó technikánál az alanyokra olyan eszközöket helyeznek el, melyek nem jelforrásként működnek, csupán külső jeleket vernek vissza, például fényt. A visszaverődés szögéből lehet arra következtetni [2], hogy az adott marker hol helyezkedett el egy adott pillanatban. Ezt a módszert a korábban említett szívmozgás és izommunka mérésekben lehet eredményesen használni, vagy legalábbis olyan esetekben, amikor nincs lehetőség aktív fénykibocsájtó markereket helyezni a vizsgálni kívánt objektumra. A markernek a passzív(nem fényt kibocsájtó) tulajdonsága miatt nem ajánlott olyan felületen lennie, amely jobban visszaveri a rá vetített jelet. Ez ugyanis ahhoz vezetne, hogy a rögzítő eszköz nem tudná elkülöníteni a jelölő, és a felületek közötti különbséget, megzavarhatná a lekövetést. Ezért ezeknél a rendszereknél fokozottan ajánlott a mozgást végző alanynak egy speciális öltözet viselete, melynek színe az előbb említett okok miatt mattnak, sötétnek, és alakot követőnek kell lennie. Erre a célra tökéletesen megfelel a neoprén bűvárruha, vagy valamilyen hasonló speciális ruha.



1. ábra

Passzív jelölőkkel felszerelt speciális ruhába beöltöztetett alany

2.2.1. Aktív markerrel működő rendszerek

Az aktív markerekkel történő megoldás annyiban különbözik a passzív jelölőket használó rendszerektől, hogy fényforrásokat használnak a lekövetés méréséhez. Ezt a módszert csak olyan helyzetekben lehet alkalmazni, amikor szemmel látható mozgulatsorokat kell digitalizálni. Az aktív fénykibocsájtó eszköz lehet egy lámpa, LED

(világító dióda) [1], vagy egyéb jelforrás. Ebben az esetben sem válik lényegtelené a felület, melyen az aktív jelölők vannak, ugyanis ha erős fénnel világítanak aránylag tükröződő, kevésbé matt, vagy egyszerűen világos felületről, könnyen összezavarhatják a mozgásrögzítő rendszereket.



2. ábra
Aktív markerek használata

2.2.2. Egyéb megoldások

Rádiófrekvenciás pozicionáló rendszerek: Itt a markerek rádiófrekvencia segítségével követhetők le. Minél magasabb a frekvencia, annál pontosabb mérést eredményez a módszer.

Létezik olyan alternatív megoldás mozgás lekövetéséhez [2], melynél a vizsgált alany viszonylag „nagy területet” járhat be kis helyen. Ezt úgy teheti meg, hogy egy speciális, erre a célra létrehozott gömbben tud mozogni, melyben belső szenzorok vannak. Így megspórolhatóak az extra kamerák, a plusz költségek, valamint az alanyra helyezett markerek egyaránt. Hátránya ennek a megoldásnak, hogy csak egy irányba tartó mozgást tudunk vele rögzíteni.

Létezik ezen kívül egy 6 szabadsági fokkal rendelkező mozgásplatform, pontosabban egy speciális „taposómalom” elven működő eszköz, mely nagy felbontású optikai szenzorjai révén tud mozgást lekövetni. A módszer segítségével a lekövetett személy képes határtalan területet bejárni, még egyenetlen felszínen is. Itt egy sok helyen alkalmazott eljárásról van szó. Alkalmazási köreibbe beletartoznak az orvosi rehabilitáció egyes területei, ahol a kiegyensúlyozott járás elsajátítását és gyakorlását érik el vele. Használati területei közé tartoznak még a biomechanikai kutatások és a virtuális valóság irányába folyó kutatások is.

2.3. A háromszögeléses módszer

A háromszögeléses módszer egy olyan trigonometriai, geometriai eljárás, mely során, ha ismerjük egy háromszög két csúcsának koordinátáit, valamint a belső szögeket, akkor meghatározhatóak a harmadik csúcs [4] koordinátái. Leginkább a földmérés-tanban alkalmazzák. Képlete:

$$l = \frac{d}{\tan \alpha} + \frac{d}{\tan \beta}$$

Ebből d -t kifejezve megkapjuk a következő összefüggést:

$$d = \frac{l}{\left(\frac{1}{\tan \alpha} + \frac{1}{\tan \beta}\right)}$$

A

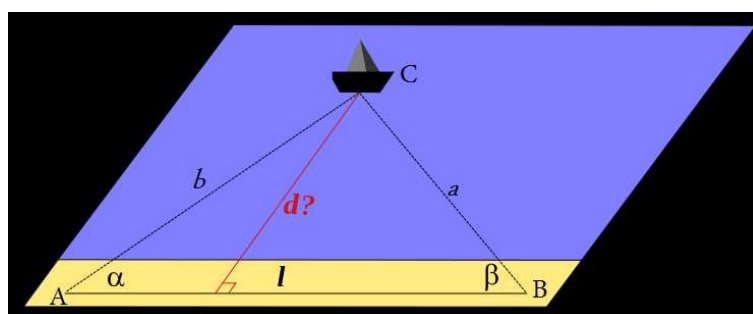
$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha} \quad , \quad \sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta$$

trigonometrikus azonosságokat felhasználva pedig így alakítható át:

$$d = \frac{l \sin \alpha \sin \beta}{\sin(\alpha + \beta)}$$

ahol α a b és l oldal által bezárt szög, β pedig a l és a oldal által bezárt szöget jelöli. l hosszának, vagy A és B csúcs koordinátáinak ismeretében alkalmazhatjuk a párhuzamos szelők tételét, mellyel meghatározhatóak lesznek a C csúcs koordinátái és d hossza, vagyis az l -től mért távolság(3. ábra).

Magasság mérésekre gyakran használják [3], valamint a GPS technológia elterjedése előtti időkben ez volt az egyetlen rendelkezésre álló eszköz a földmérők kezében.



3. ábra

Egy hajó parttól való távolságának meghatározása

2.4. Mintavételezési eljárás

A mintavételezés során egy jelből kiválasztjuk az értékek egy véges halmazát. A kiválasztott értékek lesznek a minták. Két dimenziós esetben, vagyis képeknél ez az xy sík egy rácsba, másnéven pontok rendezett geometriai viszonyába való felosztását jelenti.

Egyenletes mintavételezésnél vegyünk egy $n \times n$ -es mátrixot, melynek elemei képpontok lesznek, és Δx legyen a mintavételezés sűrűsége, vagyis ez az érték fogja jelenteni azt, hogy milyen sűrűn veszünk a digitalizálni kívánt képből mintákat. A mátrix forma a digitális képek leggyakoribb megjelenítési formája. Folytonos kép mintavételezésekor a képen lévő információknak csak a töredékét tudjuk ábrázolni, ebből következik, hogy egy jel mintavételezése mindig információvesztést eredményez, melyből az eredeti jel nem állítható vissza. Gyakorlatilag ebből az adódik, hogy a mintavételezési intervallumot legalább olyan méretűre, vagy kisebbnek kell választani, mint a képen lévő legkisebb, számunkra fontos részlet fele. Milyen sűrű mintavételezés szükséges egy adott képhez? A benne lévő legmagasabb frekvencia határozza meg. Ebben segít minket a *Fourier-transzformált*.

A továbbiakban jelöljük W -vel a függvény frekvenciáját, vagyis a görbéjének az x tengellyel metszett szakaszainak hosszát. *Sávhatárolt függvények* azok az f függvények, amelyek nem tartalmazznak W -nél nagyobb frekvenciát. Ez akkor következik be, ha van olyan W , hogy

$$[Ff](x) = 0$$

ha $|x| \geq 0$. A képek Fourier-transzformáltjai olyan gyorsan csökkenő függvények [6], melyben a magasabb frekvenciák fokozatosan kisebb szerepet játszanak. Így található lesz bennük egy olyan határ, melytől nagyobb frekvenciát már nem veszünk figyelembe. Ez a határ képes olyan közelítést megadni, melynek segítségével az eredeti jel a lehető leghasonlóbb. Ez a frekvencia gyakorlatilag egy súlyozott értéket biztosít a képre vonatkozólag.

Az algoritmusomban is mintavételezés történik, melyben az aktuális képsorozatról a mintavételezés gyakoriságát és a minták méretét a program a felhasználóra bízta, ezzel lehetőséget biztosítva számára a kísérletezésre. Tesztelhetővé válik számára, hogy arányosan hogyan növekszik a feladat bonyolultsága a mintavételezés számának növekedésével.

2.5. Éldetektálás

A térképek digitalizálásának egyik legfőbb célja az automatikus raszter-vektor konverzió. A legtöbb éldetektáló, konverziós eljárás arra törekszik, hogy vonalkövetést valósítson meg, még hozzá úgy, hogy a hasonló intenzitású, színű pixelekből előállt vektoros állomány nyomvonalát jelölje ki minél pontosabban. Az ehhez alkalmazható eljárások közül a fontosabbakat tekintjük át. Képelőkészítéskor alkalmazhatunk Canney-féle éldetektort, medián szűrőt, zaj, és frekvencia szerinti szűréseket, szín szerinti leválogatást és cserét, színikivonást és nem utolsó sorban szegmentáló eljárásokat. Ezek megértéséhez viszont ismernünk kell a *konvolúció* fogalmát.

2.5.1. Konvolúció

Tekintsük egydimenziós esetben a konvolúciót [6]. A következő képletben $h = f_1 * f_2$ a konvolúció, és f_1, f_2 folytonos függvények jelölése.

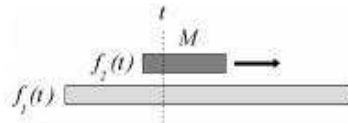
$$h(t) = f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t - \tau) d\tau$$

Digitális jelekre alkalmazva:

$$h(t) = f_1(t) * f_2(t) = \sum_{\tau=-\infty}^{\infty} f_1(\tau) f_2(t - \tau)$$

A konvolúció fogalma könnyen átültethető digitális képekre is, vagyis kétdimenziós függvényekre. A digitális szűrési módszerek egyik legfontosabb fogalma a *kernel* [6], másnéven a *mag*. A szűrési eljárások az aktuális időtartományban a kernellel konvolválják a szűrendő képet. A konvolúció hatása attól függ, hogy milyen függvény értékeit használjuk fel a kernelhez. Ezek az értékek egy $n \times n$ -es mátrixot alkotnak, ebből fog állni a mag. Ha a rendelkezésre álló adatok alapján meg tudjuk adni, hogy milyen átviteli függvényt kívánunk megvalósítani a frekvencia tartományban, akkor annak inverz Fourier-transzformálásával megkapjuk az időtartománybeli függvényt, amelyet megfelelően mintavételezve megkapjuk a szűrőegyütthatókat, vagyis a kernelbe betöltendő számokat. A szűrést ekkor az időtartományban végezzük el a következő módon: A szűrt adatrendszer ($g'(t)$) tehát úgy kapjuk meg, hogy képezük a t időtartománybeli eredeti adatrendszer ($g(t)$) és a kernel ($s(t)$) szorzatát.

$$g'(t) = g(t) * s(t)$$



4. ábra
A konvolúció szemléletes jelentése

A kernel megállapítása nemcsak a frekvencia szerinti szűrők esetén játszik fontos szerepet. Az élmegőrző, élkimelő szűrők létrehozásakor is kulcsfontosságú.

A kernel és a kép konvolúciójakor a következő történik: Vegyünk egy 3×3 -as pixel-méretű kernelt. A kernel végighalad a képen, és a középső, sötét mezője hat az alatta levő képpixelre. A kernel első mezőjében levő számot megszorozzuk az alatta levő pixel RGB színével, majd tároljuk az eredményt. A kernel második mezőjének számát szorozzuk meg az alatta fekvő képpont színével, majd a szorzás produktumát adjuk hozzá az előző eredményhez, és így tovább. Tehát minden szorzatot hozzáadunk az előzőhöz, majd az így létrejött összeget elosztottuk a kernel elemszámával. A művelet eredménye egyetlen pixelnek a szűrt színe lesz. Ez a műveletsor minden képpont felett végrehajtandó a teljes kép szűréséhez.

2.5.2. Canney-féle éldetektor

Egyik legegyszerűbb módja az élsimításnak (*antialiasing*) a kép és egy Gauss-függvény konvolúciója:

$$g_{\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$$

A következő képletben legyen h az f és g függvények konvolúciója. Bebizonyítható, hogy:

$$h = (f * g)' = f * g'$$

vagyis egy f jel g Gauss-függvénnyel való konvolúciójának deriváltja megegyezik az f és a g deriváltjának konvolúciójával. Ebből következik, hogy az éldetektálás egy- és kétdimenziós esetekre nézve a következő algoritmussal idézhető elő:

1. Konvolváljuk f jelet g Gauss-függvény deriváltjával
2. Számítsuk ki az így keletkezett h abszolútértékét
3. definiáljuk éleknek mind azokat a helyeket, ahol $|h|$ abszolút értéke meghalad egy előre beállított küszöbértéket.

2.6. Optical flow

Az *optical flow* egy olyan technika [25], amellyel megvalósítható egy képsorozaton nem csak egy, hanem akárhány mozgásrészlet lekövetése, pontok kijelölése nélkül. Ez viszont igen nehéz és időigényes feladat, ugyanis óriási – egy aránylag kisméretű képekből álló sorozatnál is – a számítás komplexitása. A másik gond a módszerrel, hogy – mivel az emberi látás nem analízisen, hanem felismerésen alapszik, – nem mindig egyértelmű, hogy lekövetéskor mi a jó megoldás (ami számunkra egyértelmű, az a programoknak nem mindig az). Egy ilyen elven működő program számára az is nehéz helyzet, hogy például sok hasonló pixel milyen irányba mozdulhatott el.

Az optical flow eljárással gyakorlatilag az eredeti kép mására egy olyan átalakított képet kapunk, amely kétdimenziós irányvektorokat tartalmaz. Elviekben azokat a pixeleket, – melyek kezdőpontjaik az irányvektoroknak – mozgatjuk az általuk mutatott pixel helyére. Ekkor megkapjuk a képsorozat következő elemét. Ezt viszont úgy tehetjük meg, hogy minden képet *kép-piramissá* alakítunk. Mivel az eredeti, nagyfelbontású képen rengeteg számítást venne igénybe minden pixel új helye a következő képen, ezért először az eredeti kép egy nagyon alacsony felbontású mását vizsgáljuk, amin kevés számítást igényel az új helyzet meghatározása. A továbbiakban valamivel élesebb képet vizsgálunk ugyanígy, és ezt addig végezzük el, míg az eredeti képpel megegyező képet vizsgálunk.

Abban az esetben, ha ismerjük az összes képpont mozgását, akkor azt sokféle speciális *effekthez*, másnéven hatáshoz használhatjuk fel. Az egyik legelterjedtebb ilyen hatás a *retime*, aminél gyakorlatilag *slow motion* hatást, belassított mozgást érnek el. Ezt úgy érjük el, hogy a képsorozat elemei közé újabb frame-eket, képkockákat generálunk, melyeket úgy hozunk létre, hogy az optical flow-t tartalmazó eredeti képünkön az irányvektorok hosszát felezzük; vagy még kisebbre vesszük, ha nem csak egy képet kívánunk beszúrni két kép közé. A legismertebb példák erre a Mátrix filmek.

Szintén elterjedt hatás még a *motion blur*, azaz utólagos bemozdulás létrehozása is. Ezzel érhetjük el azt, hogy kihangsúlyozzuk valaminek a gyors mozgását. Egyszerűen elmoszuk a képet a mozgásvektorok irányába.

Ha olyan képsorozattal dolgozunk, amelyben a tárgyak külön nem mozognak a képen, tehát ha mondjuk a felvételt nézve térben haladunk például egy utcán, akkor lehetségessé válik az is, hogy utólag módosítsuk a kép mélységélességét.

A modern videótömörítő kodec-ekeknél használatos a *block matching* algoritmus a hierarchikus megközelítés helyett, mivel ebben az esetben a sebesség fontosabb, mint a pontosság. A képet úgynevezett blokkokra osztjuk fel, és a vektorok kiinduló pontjai nem a kép egységelemei, hanem ezek a blokkok lesznek. Ebből következik, hogy ezeket a vektorokat felhasználva nem tudnánk előállítani slow motion effektet, de még így is fel tudjuk gyorsítani a videótömörítést.

Az optical flow-t, mint módszert gyakorlatilag a programomban is alkalmazom, annyi különbséggel, hogy a felhasználó jelöli be a követni kívánt részeket.

2.7. Elterjedt mozgásrögzítő és lekövető rendszerek

Korábban említettük, hogy videójátékokban, és általában a szórakoztatóiparban is használatos a digitális mozgásrögzítés. Köztudott, hogy jelenleg alig van játékkonzol gyártó cég, viszont azokból egyiknek sincs monopóliuma az ipar ezen területén. Ezek a cégek a japán Nintendo, a szintén japán székhelyű Sony és az amerikai Microsoft.

Ezek a vezető cégek évről évre próbálják megújítani az elektronikus szórakoztatást különféle eszközökkel. Mivel gyakorlatilag pár év alatt a játékok *grafikája*, vagyis a realisztikusságuk kétdimenzióban a határokat feszegeti – tehát a jelenleg rendelkezésre álló eszközökkel alig fejleszthető tovább, vagy legalábbis lelassult a „javuló” tendencia – , ezért valami olyan ötlettel kellett előrukkolniuk, ami ténylegesen megreformálja a játékipart. Mivel akkoriban, pontosabban 2004 környékén még nem terjedt el a manapság mindenhol hangzó 3D megjelenítés, ezért a Nintendo ötlete az volt, hogy készít egy olyan játékkonzolt, a Nintendo Wii-t, melyhez mozgásérzékelős kontroller tartozik. Ez a konzol egyébként 2006-ban jelent meg, mint a hetedik játékkonzol-generációban az első mozgásérzékelős kontrollerrel rendelkező konzol, és magát az irányító eszközt 2005-ben mutatták be.

A kontroller neve Wii Remote, másnéven Wiimote, amely lehetővé teszi használója számára, hogy virtuális tárgyakkal, dolgokkal kapcsolatba lépjen mozgásrögzítés és lekövetés segítségével, amit a konzolba épített infravörös optikai szenzor és egy gyorsulásmérő valósít meg. Ezenkívül a kontrollernek van egy párja is, a Nunchuk, amely annyival egészíti ki a Wiimote-ot, hogy több, gamepad-okhoz hasonlító, funkciójukban megegyező gombot tartalmaz.

A Wii Remote ezenkívül elnyerte a hackerek kitüntető figyelmét is, akik különféle kreatív megvalósításokat ötlöttek ki hozzá. Az ötletek közé tartozik például a

Roomba nevű takarító robot, melynél a robot irányítható a kontrollerrel, valamint ujjlenyomat leolvasó eszköz is készült a segítségével. Az utóbbi ötlet egyébként a *Különvélemény* című filmben látottakon alapult.

A Wiihez azóta is rengeteg kiegészítő jelenik meg, ezek közül talán a legjelentősebbek közé tartozik a Wii MotionPlus, ami gyakorlatilag megtalálható beépítve a Wii Remote Plus-ba. Ez az eszköz 2010 november 5-én fog előreláthatólag megjelenni Európában, és sokkal precízebb, pontosabb irányítást tesz lehetővé, mint a normál Wiimote. Egy hangvilla alakú két tengellyel rendelkező beépített giroszkópja van, amivel körforgó mozgást is le tud követni. A kapott információt egy szög-arány szenzor dolgozza fel, amely megkülönbözteti a szabályos egyenes vonalú, lineáris mozgást a sebességmérőtől kapott jeltől. Ennek hatására a vetélytársaknak is lépést



5. ábra

Bal oldalon Wii MotionPlus, középen bal oldalon Nunchuk, mellette Wii Remote, jobb oldalon Nintendo Wii többféle színben

kellett tartaniuk a Nintendo-val, így a Sony kifejlesztette a PlayStation Move nevű kontrollerét, amely kiegészítője a PlayStation 3 nevű konzolnak. A PS3 is a hetedik konzolgenerációban jelent meg még 2006-ban, viszont a Move csak 2010-ben.

Ajánlott együtt használni a PlayStation Eye nevű eszközzel, ami egy webkamerához hasonlít, annyi különbséggel, hogy mozgást is képes követni. A mozgás követése mellett színérzékeléssel képes arra, hogy a felhasználók „hatást” gyakoroljanak a játékokban különféle virtuális dolgokra, például tereptárgyakra.

A PlayStation Eye négyszer nagyobb felbontású és kétszer gyorsabb, mint az EyeToy, ami a PlayStation 2 kiegészítője volt. 60 hertzen 640×480 felbontású, és 120 hertzen 320×240 felbontású képet képes rögzíteni. Összességében kétszer pontosabb az EyeToy-nál. Tartalmaz egy két-beállítású rögzített fókusszal rendelkező lencsét.

Manuálisan állítva a lencsét elérhető 56°-os látószög, ekkor fokban tudta vizsgálni környezetét az EyeToy is. Ezenkívül beállítható 75°-os látószög is, amelyet chat programoknál, vagy interaktív fizikával rendelkező játékprogramoknál ajánlott használni. Habár a legtöbb PlayStation Eye szoftver az éldetektálás, színkövetés és a Digimask nevű *avatar* technológiát, illetve módszereket használja, a PS Move bejelentése óta számos újfajta alkalmazási módszert találtak ki. Ezek közül a jelentősebbek a Vision Library, amely képes arcot felismerni, és a felhasználó fejének mozgása is részt vesz a lekövetésben. A Sony szerint ez a technológia a szemeket, a szemöldököt, az orrot és a szemüveget is fel tudja ismerni, kivenni a kapott képből a száj alakját, és azonosítani a mosolyt. Ezenkívül még a kor és a nem meghatározására is képes.

A PlayStation Move is egy mozgáskontroller, ami pálcához hasonlít, és a mozgás során kapcsolatba tud lépni a PlayStation 3-al, a PlayStation Eye-on keresztül. A Move tetején egy LED lámpákból álló golyó helyezkedik el, amely képes sokféle színben világítani. Így a PS Eye dinamikus kiválasztja az éppen aktuális színű fényt, és ennek a segítségével leköveti a Move mozgását. Ez a színes fény aktív markerként szolgál a PS Eye számára, amely a képsíkon leköveti a kontroller aktuális helyzetét. A fény képének a méretéből könnyedén meghatározhatja az Eye a világító gömb távolságát, így 3D-ben ábrázolhatóvá válik a kontroller pontos helyzete. A többi PS3-hoz tartozó kamerán alapuló irányítási technikával szemben ez aránylag kevés számítási kapacitást igényel, ami azt eredményezi, hogy alacsony késleltetési idővel működik a Move. Egy pár tehetetlenségi szenzor, egy három tengellyel rendelkező gyorsulásmérő, és egy háromtengelyű elhelyezkedési szög érzékelő is helyet kapott a kontroller belsejében, amikkel a kontroller 3D-s környezetben teljes mértékben képes meghatározni önmaga pontos helyzetét. A mérési számításokból következő esetlegesen megnövekvő hibamennyiséget egy magnetométer korrigálja, a Föld mágneses mezőjét figyelembe véve. A PS Eye egyszerre négy különböző Move kontrollert tud lekövetni, ha különbözőek a golyók színei.

A Microsoft először 2005-ben jelentette meg az Xbox360 nevű konzolját, amely az Xbox utódja. Üzleti okokból tartalmazza a neve a 360-as számot, mert a 3-ast tartalmazza, és így a vásárlók felé azt az üzenetet közvetíti, hogy egy szinten van a PlayStation 3-al. Kezdetben a konzol neve egyébként Xbox 2 volt.

Ez a konzol szintén a hetedik konzolgenerációba tartozik. Teljesítményileg a Nintendo Wii és a PlayStation 3 között helyezkedik el, viszont előnyre válik, hogy PlayStation 3-ra állítólag nehezebb fejleszteni, mint Xbox360-ra. Az Xbox360-hoz is jelentek meg



6. ábra

Balra PS Move kontrollerek, középen PS Eye, jobbra egy PlayStation 3 konzol, mellette a normál kontrollere hátulról

kiegészítők, az egyik jelentősebb az Xbox Live Vision, ami egy webkamera, és például videóchathez is lehet használni. A másik kiegészítő a 2010-ben megjelent Kinect, másnéven Kinect for Xbox360. A Kinect-ről sokáig Project Natal kódnéven lehetett információt szerezni. A felhasználó számára lehetővé teszi azt, hogy fizikai kontroller és irányító eszköz nélkül tudjon kapcsolatba lépni a konzollal csupán kimondott parancsszavakkal és mozdulatokkal. A Microsoft 2010 decemberében jelentette be, hogy a Windows operációs rendszert tartalmazó PC-kre is elhozza a Kinect technológiát, bár már köztudott, hogy egy pár apró, szoftveres módosítással a Kinect képes asztali PC-kkel működni. A kiegészítő egy távolsági kamerával rendelkezik, amely az előtte elhelyezkedő dolgokra képes folyamatos infravörös „rendezett fényt” vetíteni információszerezés céljából. A „rendezett fénnel” a kamera a megvilágított objektumokról mélységi és felszíni adatokat kap. A térbeli helyzetmeghatározáshoz tartalmaz egy fénykódoló 3D scannert is, amely az előbb említett infravörös fényből kapott adatokból dolgozik. Ezek a szenzorok automatikusan állítódnak, a külvilágtól és a játszott játéktípustól függően. A hangfelvételhez több sornyi mikrofont használ. Ezekkel a beépített eszközökkel teljes mértékben képes célját elérni, vagyis ezek teljes emberi mozgásrögzítésre, arcfelismerésre, és hangfelismerésre is képessé teszik. A Kinect legfeljebb hat ember mozgását tudja egyszerre lekövetni.

3. Témához kapcsolódó középiskolások számára előíráható feladat

A fentiekből látható, hogy a téma megfelelő motivációs erővel rendelkezik, ugyanakkor megértése és megvalósítása lényegesen mélyebb matematikai és programozási



7. ábra

Balról Kinect, középen Xbox Live Vision, a jobb szélén jobb oldalon a régi típusú Xbox360 konzol, mellette bal oldalt az új Xbox360 Slim

ismereteket igényel, mint ami elvárható a legtöbb középiskolástól. Ezek miatt szükséges egy olyan megoldási módszert találni, amely jobban illeszkedik a korosztály számára előírt ismeretanyaghoz.

3.1. Feladat meghatározása

A programnak úgy kell működnie, hogy a lehető legpontosabban, leggyorsabban és leghatékonyabban végezze el a feladatát. Csak és kizárólag annyi memóriát használjon fel, amennyi a tényleges probléma megoldásához feltétlenül szükséges, vagyis jól optimalizált legyen. Képes legyen a program a mozgó objektum éppen aktuális helyzetét felismerni, és ezt a legelső képkockától a legutolsóig minél precízebben lekövetni. A háttér érdekes elkülönítenie a látványosabb, jobban kivehető eredmény érdekében. Szövegesen elemezze, és értékelje a vizsgált objektum típusától függően a mozgását, és fontos dolog, hogy minél több nyelven tudja ezt megtenni.

3.2. Célkitűzés

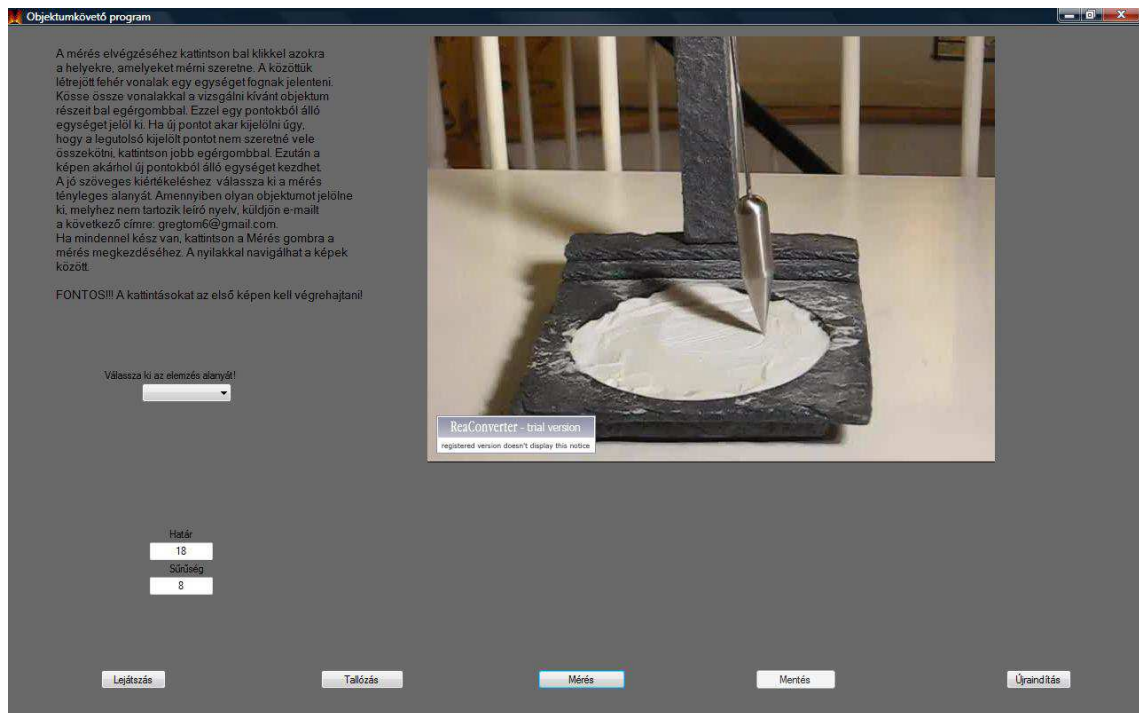
A cél egy olyan algoritmus létrehozása, amely nem használ fel külső eszközöket, markereket a mozgások lekövetéséhez, sokkal inkább az adott objektum természetes mintázatát veszi mérésének alapjául. Elemi módszerekkel valósítja meg ezt a műveletet, legalábbis elemibb lépéssorozattal, mint más hasonló MoCap program.

3.3. Megoldási menet

Keresnünk kell egy olyan fejlesztői környezetet, amely ennek a megvalósításához a lehető legpraktikusabb eszközöket tartalmazza. Általa biztosított lehetőségek alapján a Microsoft .NET Framework 4.0-ás verziójú fejlesztői környezet, C# programozási

nyelvvél bizonyult legmegfelelőbbnek a feladat megvalósítására.

Az univerzalitás megvalósulásának érdekében választható 3 különböző nyelv



8. ábra

Példa a program kinézetére

(magyar, angol, német), amelyen a program üzenetei megjelennek. (A program nyelvét közvetlenül a program indítása után választhatja meg a felhasználó).

A programnak a korai hibák gyors kiküszöbölésére, későbbi tesztelések céljából ajánlott egy alapértelmezett mappát tartalmaznia. Ebből a mappából fog beolvasni egy előre elkészített képsorozatot, mellyel a felhasználó és a programozó megbizonyosodhat programjának megfelelő működéséről. Ha ezt a mappát, vagy ennek a tartalmát valamilyen okból nem kívánjuk felhasználni, akkor Tallózás gomb lehetővé teszi, hogy a programhoz saját képsorozatot töltsünk be egy megadott mappából.

A beolvasott képsorozat első elemét a program megjeleníti a felhasználó számára, hiszen a legelső képen kell kijelölni azokat a pontokat, melyeknek a követését szeretnénk elvégezni a program segítségével. A képeket eltároljuk a memóriában, hogy felkészítsük a programot a *frame*-ekre vonatkozó folyamatok háttérbeli futtatására. (A képek meglétére különféle feltételeket vezetünk be, melyek kizárják, hogy üres, vagy képeket nem tartalmazó mappa kerüljön tallózásra.) A kijelölt *pixeleket*

(képpontokat) vörös megnagyobbított pont jelzi a képeken. A pontok között egy fehér vonal fogja jelezni, hogy azok a pontok egy egység részei, vagyis összetartozó, szomszédos pontok lesznek. (Lényegében egy gráfot jelölhetünk ki a képen.)

Érdemes az összes képhez olyan adatszerkezeteket társítani, amely alkalmas arra, hogy a felhasználó által kijelölt gráfot tárolja. Ezt az adatszerkezetet célszerű listákból felépíteni.

Ha a felhasználó úgy dönt, hogy a két legutóbb kiválasztott pont ne tartozzon egymáshoz, vagyis ne legyenek vonallal összekötve, – a mérésük egy másik pontokból álló egység részeként fog megtörténni – akkor a felhasználónak jobb egérgattintást kell elvégeznie, ami jelezni fogja a programnak az adott egység lezárását. Ennek az adatszerkezetben is tükröződnie kell.

Ha a felhasználó még az első listát tölti fel, akkor a pont helye eltárolódik a listában, viszont abban az esetben, amikor már kettő vagy több lista van feltöltve, a programnak meg kell találnia az adatszerkezetben azt a pontot, amely az új lista csatlakozási pontja lehet. Tehát ha az új pont valamelyik korábbi pont meghatározott környezetében helyezkedik el, akkor azt a pontot egy közös pontnak tekintjük, és az aktuális pont koordinátái egyenlők lesznek a korábbi közeli pont koordinátaival. Ezzel biztosíthatjuk azt is, hogy a program ténylegesen egységként kezeljen több pontsorozatot tartalmazó listát.

Alapjában véve az egységeknek a legfőbb szerepük, hogy láthatóan elkülönítsék a vizsgálat alanyának részeit, ha esetleg az nem járható be *Euler*-útként. Ennek a megvalósításához eleve olyan listákat definiálunk, melynek minden eleme újabb lista lesz, és ezek egy adott képkockához legyenek hozzárendelve. Így biztosítható lesz az, hogy egyszerűen lekérdezhető és módosítható lesz az i -edik kép j -edik listájának (pontokból álló egységének) k -adik pontja, ahol $1 \leq i \leq n$, n a képek számossága, és i az aktuális kép. $1 \leq j \leq m$, ahol m az adott i -edik kép összes listájának, egységének számossága, és $1 \leq k \leq o$, ahol k az adott i -edik kép j -edik listájának aktuális pontja, és o ugyanebben a pontokból álló egységlistának a maximális pontszáma.

A program elemzi a pontok elhelyezkedését a képen, és ezek alapján jelöli ki a gráffal leírt új objektum helyzetét. Vagyis egy lista tartalmazni fogja egy adott képsorozat feldolgozása utáni, az egyes képhez tartozó, a feldolgozás eredményeként adódó gráfok (amelyek jellemzik az objektum mozgását) leírását. Alapértelmezésben a program az előre, mintaként beállított képsorozatok alanyainak mozgását tudja



9. ábra
A kijelölés utáni kép

szövegesen elemezni és kiértékelni, pontosabban az ember elhelyezkedését egy adott képen, de ez később új típusú objektumok kiértékelésére is alkalmazható.

A pontok kijelölése után a helyes leíráshoz a program egy mozgóképet mutat meg, valamint szövegesen is leírja, hogy a felhasználónak hogy kell kijelölnie az objektumot. Természetesen, ha nem meghatározott mennyiségű, típustól függő pont van kijelölve a képsorozat első elemén, akkor a szöveges elemzés aktiválási szándéka ellenére sem lesz látható a mérés végén a kiértékelés. Ezt közöljük a felhasználóval egy információs ablakban.

A program lehetőséget biztosít az eredeti képsorozat és a lekövetett pontok mozgá-



10. ábra
A mozgókép egy képkockája

sának lejátszására, és a feldolgozás eredményének mentésére egy, a felhasználó által kiválasztott mappába.

Abban az esetben, ha a felhasználó elrontotta a kijelölést, vagy a mérés után újabb

mozgás feldolgozást akar végezni ugyanazon a képsorozaton, lehetősége van kezdőállapotba állítani a programot. Ekkor az éppen használt képsorozat eredeti képkockáit tölti vissza a program, majd a kijelöléseket és a szöveges elemzővel kapcsolatos beállításokat alapértelmezett állapotra állítja vissza.

4. Megoldás

4.1. RGB színmodell

Ez egy olyan *additív* színmodell, amely a vörös, a zöld és a kék színek kombinációjával határozza meg az összes többi színt. Az *RGB* rövidítés [6] is a 3 additív elsődleges szín angol megfelelőjének kezdőbetűiből ered. (red, green, blue). A 3-féle szín erősségét 0-tól 255-ig mérjük, ahol 0 jelenti az üres színt (százalékos értéknek tekintve 0%), és 255 (100%) az adott alapszínt a legerősebb fényerővel. Vagyis mindhárom színhez 1 byte tárolódhat el maximum. Elterjedtek az input RGB eszközök [5], például a TV, videó kamera, scanner, digitális kamera, és output (kimeneti) RGB eszközök például a CRT, LCD, plazma, LED kijelzők, ezeken használatos a színek megjelenítésének ilyen formája. A nyomtatók például nem RGB eszközök, *szubsztraktív* színmodellt használnak, a *CMYK* modellt [10], melyben cyan, magenta, yellow és black, magyar nevükön kék, rózsaszín, sárga és fekete színek kombinációjával jönnek létre a különböző színek. Azért szubsztraktív modell a CMYK, mert a fehér szín fényességét vonja el [9], ugyanis a fehér színt fehér lapra nem lehet nyomtatni. Minden eszköz máshogy jeleníti meg az azonos értékű színeket, ezért szükség van *színvezetésre* (color management). Ez a módszer teszi lehetővé azonos RGB értékű színek ugyanolyan megjelenését eltérő megjelenítő eszközökön. Ha veszünk két legerősebb fényerejű komponenst az RGB modellből, akkor megkapjuk a következő színeket: (red, green, blue)

vörös és zöld színekből $(255, 255, 0) \rightarrow$ sárgát,

vörös és kék színekből $(255, 0, 255) \rightarrow$ magenta színt (rózsaszínt),

zöld és kék színekből $(0, 255, 255) \rightarrow$ cián színt (világoskék).

Ha összeadjuk mindhárom alapszínt, akkor fehér színt $(255, 255, 255)$ kapunk. Ha nem használunk fel egyik alapszínből sem, akkor ez fekete színt $(0, 0, 0)$ eredményez. Néhány digitális készüléknél a színeket 0-tól 65.535-ig a nevükkel lehet beazonosítani, így nem byte-okat, hanem 16-bites szavakat tárolunk el minden alapszínhez.

Létezik az RGB-nek egy kiegészítő komponense is, az [8] *alfa csatorna* (alpha chan-

nel). Ez az érték az átlátszhatóságért [7] felel. Alapértelmezésben 0 az alfa, ami az átlátszhatatlanságot jelenti, de ha például 255-re módosítjuk az értékét, akkor lényegtelené válik a többi komponens, mivel úgyszólván átlátszó lesz a szín.

4.2. A program használata

A programot elindítva egy olyan ablak indul el, mely két fő szereppel is bír. A felhasználó információt kap a program nevéről és készítőjéről, valamint az aktuális verziószámáról. Ezenkívül kiválasztható a program aktuális nyelve, mely lehet az alapértelmezett magyar, angol vagy német. Ezt követően az Indítás gombbal indítható a program fő ablakja. Itt a felhasználó megannyi lehetőség közül választhat. Amennyiben nem kívánja az alapértelmezett képsorozatot használnia, manuálisan képekkel töltheti fel, vagy tallózhat be egy új mappát. A Lejátszás gombbal képes lejátszani az éppen aktuálisan megjelenített képsorozat elemeit. (Ez lehet az átalakított és a normál képsorozat is.) Képes lementeni az átalakított képsorozatot. (a normál képeket nem) Az Újraindítás gomb arra használható, hogy alapértelmezett állapotba helyezzük a programot.

A nyilakkal lehet a képeket léptetni, a jobbra és felfelé mutató nyilakkal lehet a következő, a balra és lefelé mutató nyilakkal pedig az előző kép jelenítődik meg.

Ennél a programnál a felhasználó is a működéshez nélkülözhetetlen szerephez jut, mégpedig úgy, hogy az aktuális képsorozaton a vizsgálni kívánt objektumra bal egérgattintásokkal pontokat helyezhet el, melyek a vizsgálat fő mérő eszközei lesznek. Csak és kizárólag az első képen lehet kijelölni pontokat, ez a bal egérgomb segítségével tehető meg. Minden új pontot összeköt a program egy egyenessel. Abban az esetben, ha nem kívánjuk összekötni a legutóbb kijelölt pontot az új ponttal, jobb egérgombbal szükséges kattintani. A legutolsó egységet lezárva van lehetőség a lekövetést elindítani.

Az objektum helyzetének szöveges kiértékeléséhez a kijelölés előtt vagy után szükséges kiválasztani a lekövetni kívánt objektum típusát.

Lekövetés után új mérés nem kezdeményezhető, csak akkor, ha új képsorozatot találunk be, vagy használjuk az Újraindítás opciót.

4.3. A lekövetés algoritmus

A Mérés gombra kattintva a program elemzéseket végez, például azzal kapcsolatban, hogy annyi pont van-e kijelölve, amennyi a szöveges kiértékelés aktiválásához kell,

valamint két felhasználó által beállított érték helyességét vizsgálja meg, a *határ* és a *sűrűség* értékét. A *határ* értéke a lekövetési terület méretét, a *sűrűség* pedig egy adott pont lehetséges új pozícióit határozza meg a rákövetkező képen.

Az egymást követő képeket úgy kell elképzelnünk, mint egy négyzetrácsos lapot, itt a négyzeteket a pixelek jelentik. Ezek közül választ a felhasználó. A program itt veszi figyelembe a mérés előtt beállított *határ* változó értékét. A kattintás helye egy több pixelből álló négyzet, vagyis egy pixelekből álló mátrix középső képpontja lesz. Ennél a pixelnél lesz érvényes az a kijelentés, hogy az indexét a mátrixban található oszlopok számának, és a sorok számának fele határozza meg. A *határral* adhatjuk meg a kijelölt pont azon környezetét, amelyet felhasznál a pont követéséhez. Azt mondhatjuk, hogy ha például a *határ* értéke 4 volt, akkor egy kijelölt pont körül egy olyan létrehozott mátrix pixeleit vizsgáljuk, melynek minden oszlopának és sorának számossága a *határ* kétszerese, tehát egy ilyen négyzetben 64 darab képpontot kell átvizsgálunk. Ezeket a pixeleket listába fűzzük, vagyis lineárisan haladhatunk a mátrix elemein, és majd így érjük el az elemeket folyamatainkhoz. A *sűrűség* változó szerepéről a későbbiekben lesz szó.

A képek elemi részeiknek tulajdonságaik vannak, melyek közül egyetlen, a probléma szempontjából fontos tulajdonságot emelünk ki. Ez pedig a pixelek színe.

A következő lépésben vesszük a *határ* által meghatározott mátrixot az i -edik képen, és eltároljuk a pixelek színeit egy „Color” típusú, vagyis színeket eltároló listában. Érdekes ilyen típusú listát használni, mivel itt egy változó tartalmazza egyszerre a pixelek R, G és B komponenseit, megkönnyítve a munkát velük a későbbiekben. Ahhoz, hogy nagyjából egyszínű, kevés színbeli eltéréseket tartalmazó objektumok mintázatát is eredményesen lehessen lekövetni, a mátrix közepéhez közelebb álló pontok RGB értékeit súlyozzuk, vagyis nagyobb mértékben vesszük figyelembe a belsőbb pontokat, mint a külsőket. Ezt követően az $i + 1$ -edik képen vizsgáljuk az i -edik képen használt tartomány középpontjával megegyező koordinátájú pont *határ* által meghatározott környezetét hasonló módon, mint ahogy az i -edik képen vizsgáltuk a pont körüli kétdimenziós tömböt alkotó pontok színeit. Az $i + 1$ -edik kép tartományának RGB értékeivel feltöltünk egy másik szín típusú listát. Azokat a színeket, melyeket az imént tároltunk el, összehasonlítjuk az előző (i -edik) képen vett, azonos mátrixkoordinátájú pontok színeivel. Az összehasonlítást úgy tesszük meg, hogy az $i + 1$ -edik képen vett B mátrix j, k -adik eleméből abszolútérték alatt kivonjuk az i -edik képen vett A kétdimenziós tömb j, k -adik elemét, és ez az elem lesz a C eredménymátrix j, k -adik eleme. A j jelentse a sorok számát, k pedig az

oszlopok számát.

$$C_{j,k} = \sqrt{B_{j,k} - A_{j,k}}$$

Ezt a műveletet végrehajtjuk a $i + 1$ -edik mátrixot alkotó összes pixelen.

Ha ezzel kész vagyunk, kapunk egy olyan eredménymátrixot, melynek - a kétdimenziós mátrixokkal való kivonás műveletének axiómáit figyelembe véve - azonos számú sora és oszlopa lesz, mint az i -edik vagy az $i + 1$ -edik mátrixnak.

Az eredménymátrix elemeit átlagoljuk külön R, G, és B értékekre. Ez úgy értendő, hogy végigmegyünk a tartományt alkotó összes pixelen, és három különböző változóban összegezzük az összes R, G és B értékeket, majd elosztjuk mindhárom változó értékét a vizsgált területet alkotó képpontok számosságával. Ennek a műveletnek az eredményét ugyanúgy RGB kódként tároljuk el, mint különbozati értéket.

Alkalmazzuk a *minimumkiválasztás elvét* [11], vagyis vesszük a most létrehozott mátrix átlagos RGB értékét, mely mátrixnak az i -edik és $i + 1$ -edik képen vizsgált mátrixok különbségeként jöttek létre elemei. Feltételezzük azt, hogy ennek a pontnak a környezete a leghasonlóbb az i -edik képen az azonos koordinátájú pont környezetéhez. Tehát ezt a pontot vesszük az első lehetséges pozíciónak az $i + 1$ -edik képen.

Kiolvassuk a sűrűség értékét, ezt a határ változó értékéhez hasonlóan szintén a felhasználó állíthatja be. A sűrűség értéke gyakorlatilag itt jut szerephez, olyan módon, hogy meghatározza, az $i + 1$ -edik képen az eredeti pont koordinátaival megegyező pont körüli pixelekből álló kétdimenziós tömbben milyen gyakorisággal választódjanak ki a lehetséges új pontpozíciók. Például azt mondjuk, hogy a legelső képpont a 0. pixel, a sűrűség értéke 5, és indítunk egy ciklust, ami az összes képponton végigmegy, melyben az iterációs változó 0-tól n -ig tart, ahol n a pixelek száma a mátrixban-1. (Azért indul 0-tól, mivel a C#-ban és más programozási nyelvekben is a referencia típusú változókban egy adott elemre történő hivatkozásnál a legelső elem mindig a 0. elem.) Itt a lehetséges új pontpozíciók (a középső ponton kívül, amelyet a kiválasztott pontnak tekintünk legelső esetben) azok a pixelek lesznek, melyeknél az iterációs változó osztható 5-el maradék nélkül.

Az ilyen új pontok körül úgy vizsgáljuk meg az aktuális pont és az eredeti pont körüli pixelek színeinek különbségéből képzett mátrix átlagos színtartományát, mint ahogy azt korábban tettük. Amelyik korábban említett lehetséges pont környezete a leghasonlóbb az i -edik képen kiválasztott pont körüli környezetéhez, az lesz az $i + 1$ -edik képen a kiválasztott pont.

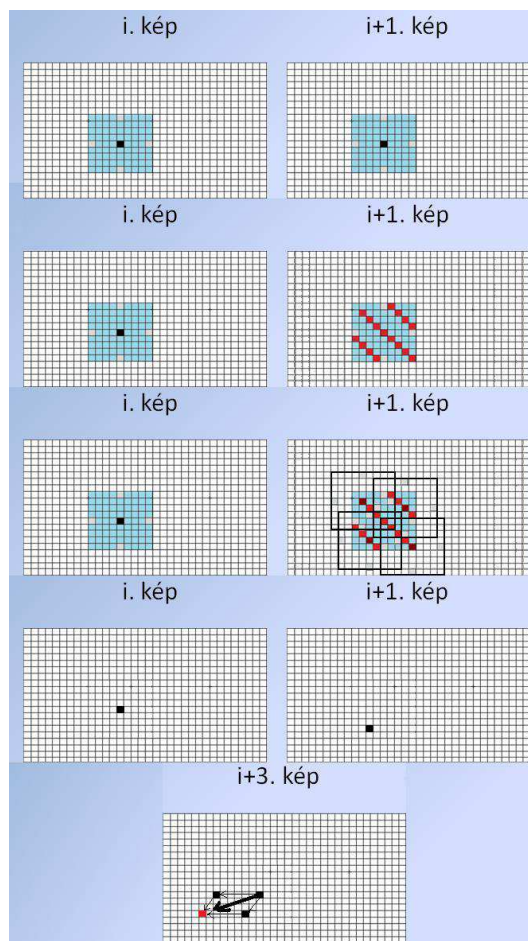
Összefoglalásként tehát vesszük az i indexet, ami a legelső képtől az utolsó-1-edik képig tart. Vesszük az i -edik képet. Létrehozunk különböző változókat, melyek indexek lesznek az aktuális képre, azon belül indexet deklarálunk egy adott pontlistára, és újabbat hozunk létre azon belül egy aktuális pontra. Végigmegyünk az i -edik képen az összes pontlistán, azokon belül is az összes ponton, és mindegyik pontnak meghatározzuk a helyét az $i + 1$ -edik képen a fent említett módon. A program ezután az összes olyan pixelt elsötétíti, amely nem vett részt a mozgás kivitelezésében, vagyis azokat, melyek nincsenek benne egy adott pontlista két szomszédos pontja által határolt szakaszban.

Az új pozíciót, koordinátákat kapott ponton gyakorlatilag egy geometriai transzformációt hajtottunk végre. A régi helyzetéhez képest eltoltuk új pozíciójába, tehát tekinthetjük ezt $\vec{a}$ vektorral való eltolásnak. Ennek a vektornak a hossza a régi és az új pont közötti szakasz hossza, és iránya az új pont iránya lesz. Feltételezzük, hogy a felhasználó egy olyan képsorozatot tallóz be, amin az eredeti videófelvétel sok egymást követő képkockája található, vagyis lehetőleg nem tartalmaz két egymással szomszédos képkockán nagy eltéréseket. Ekkor alkalmazható az az eljárás, aminek ez az elsődleges feltétele, lényege, hogy három szomszédos frame-en megfigyeljük ugyanannak a pontnak az elmozdulását, ami két vektor összegeként fog adódni. Ezzel az összegvektorral előre meghatározhatjuk, hogy körülbelül hol kell keresnünk a pont új helyét a 3. képen három szomszédos kép közül, biztosítva azt, hogy precízebben, „folyékonyabban” történjen a mozdulatsorok lekövetése.

Ezzel a megvalósítással létrejön egy mozgó, pontokból és vonalakból álló alakzat, amely – ha jó értékeket alkalmaztunk a sűrűség és a határ beállításához – teljes mértékben leköveti az eredeti képsorozat mozgó objektumát, és felismerhető lesz az ilyen módon átalakított képsorozaton az eredeti alakzat.

Affin transzformáció hajtódik végre abban az esetben, ha egy alakzat három dimenziós térben fordul el, így ennek a síkra való vetületének arányai torzulnak. A felvételen térben történő mozgás hajtódhat végre, amit két dimenziós síkban látunk, és a mért objektum oldalai torzulhatnak. Ekkor abban az esetben, ha ezt a mozgást programunkkal lekövetjük, az így létrejött alakzat arányai is hasonló mértékben változnak meg.

Mivel a lekövetés a programmal vázlatosan hajtódik végre, csontvázszerű kinézetet adva a lekövetett objektumnak, ezért erről a vázlatról viszonylag kevés információ nyerhető az eredetileg lekövetett objektumról, egyedül a vizsgált dolog típusa azo-



11. ábra
Az algoritmus nagyvonalaiban

nosítható belőle ténylegesen.

A határ és sűrűség értékeket célszerű a következőképpen beállítani: Ha olyan képsorozatot adunk programunknak, melyben akármelyik szomszédos egymást követő képkockán előfordul az, hogy a vizsgálat tárgya viszonylag nagy mértékben mozdul el, akkor a jó lekövetés érdekében ajánlott nagyobb értéket adni a határnak, ezzel együtt a sűrűséget is érdemes nagyobbra venni, hisz nagy területen, – ha nagyon alacsony a sűrűség, – nagymértékben csökkenhet a lekövetés pontossága. Ezzel együtt a végrehajtási idő is nőhet, egyenes arányban a határ és a sűrűség értékek nagyságával.

Másik lehetőségként, ha olyan képsorozatot használunk fel, melyben minden egyes képkocka és a rákövetkező frame között a vizsgált objektum elég kis mértékben tér el, ajánlott alacsonyabb határt beállítani, ezzel együtt olyan sűrűségértéket alkal-

mazni, mellyel a lehetséges kiválasztott pontok száma lefedi nagyjából a terület felét. (Tehát arányosan kevesebb, de megfelelő mennyiségű pontot a régi pontnak lehetséges új pozícióiként megválasztani.) Ezzel a végrehajtási időt jelentősen le tudjuk rövidíteni. Végül azt a konklúziót vonhatjuk le, hogy mindenképpen érdemes minél több képpel végezni a lekövetést, mint kevesebb, de nagy különbségeket tartalmazó képpel.

4.4. A program elkészítéséhez használt programozási ismeretek

1. Objektum-orientált programozás

- mezők
- metódusok
- típuskompatibilitási ismeretek
- típuskényszerítés
- kivételkezelés
- generic típusok alkalmazása(lista, hash-tábla)
- GDI+ ismeretek
- alapvető C# metódusok ismerete

2. Algoritmizálás

- minimumkiválasztás elve
- lineáris keresés

3. Matematika

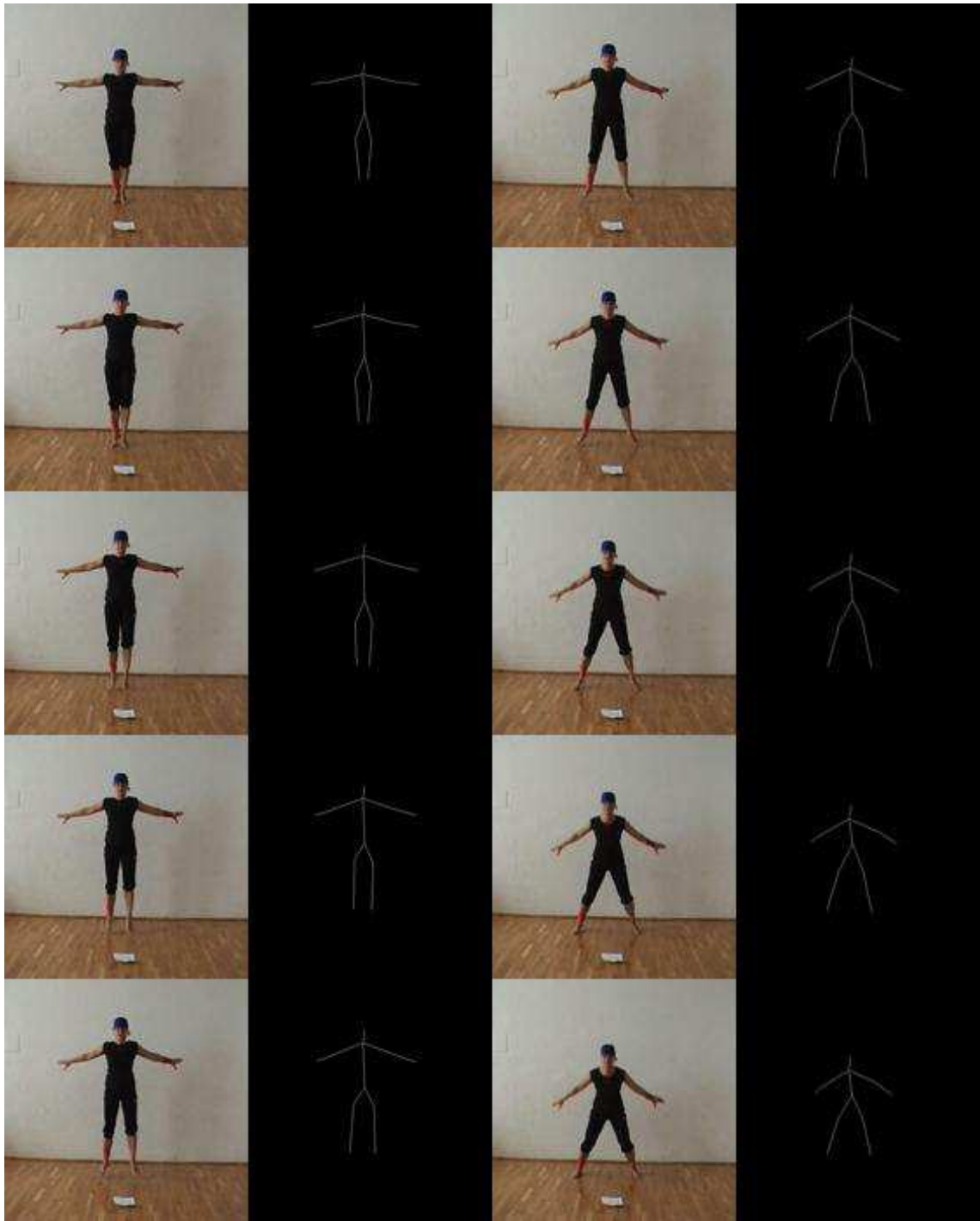
- informatikai logika
- számítási közép számítás
- vektorműveletek

4.5. Felhasználhatóság

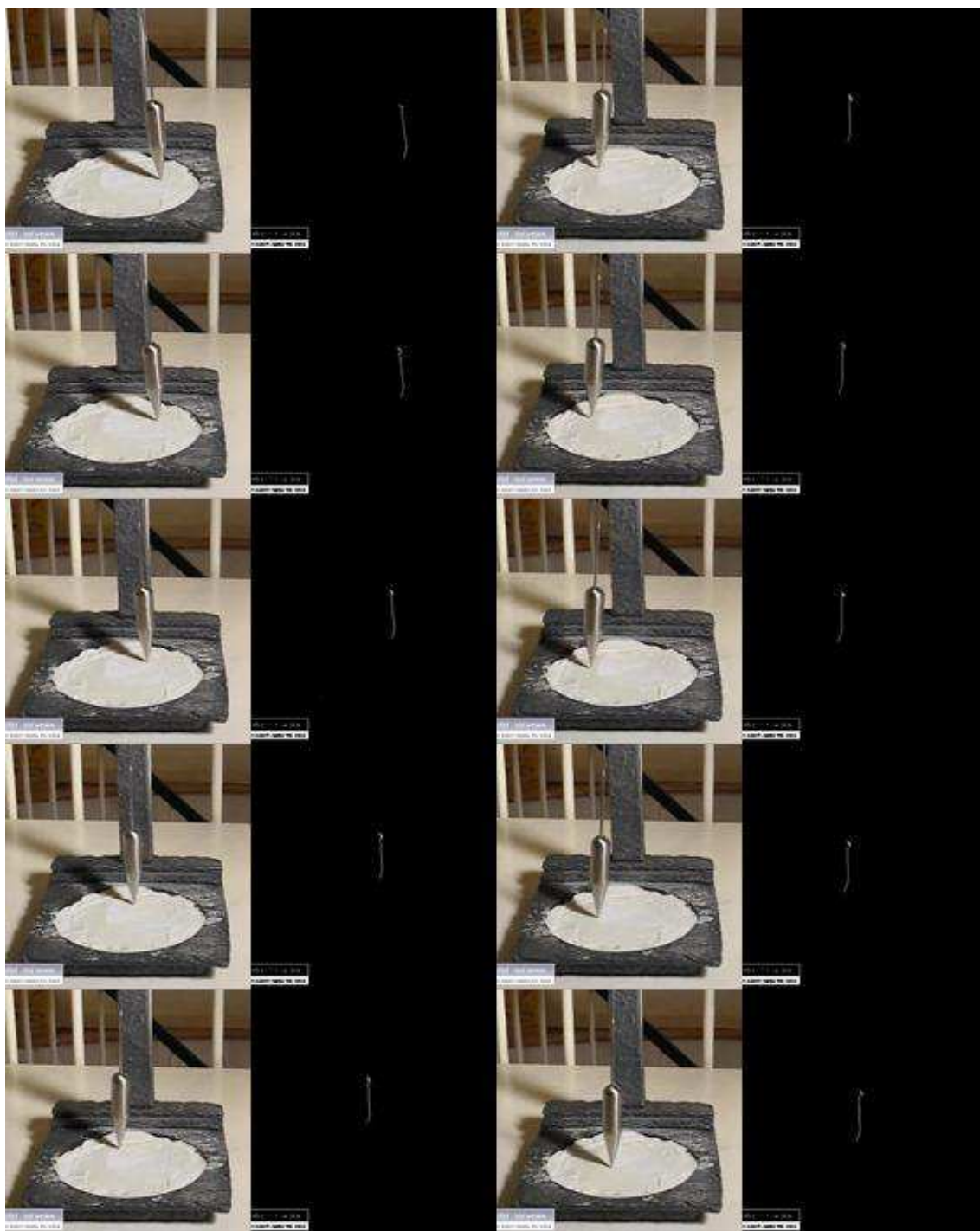
A programot oktatási célból ajánlott felhasználni, ugyanis ennek segítségével a hallgatók jobban megérthetik mind az *objektum-orientált programozást*, mind a Windows Forms lehetőségeit, és megismerhetik vele a GDI+ (Windows Graphics Device Interface) fontosabb metódusait egyaránt. Megtapasztalhatják, hogy egy mintázatot lekövető programhoz milyen mértékben szükségesek matematikai és informatikai ismeretek. Demonstrálhatjuk az informatikai eszközök egyféle népszerű felhasználását a tanulók számára. Érdekesebbé teheti számukra az informatikát és az ehhez nélkülözhetetlen matematikát. Ezenkívül a program felhasználható például arcmini-

ka rögzítéséhez, 2D-s oldalnézetes, úgynevezett *side-scrolling* játékok karaktereinek mozgásához, 2D-s scriptelt jelenetekhez, animációkhoz, mozgások elemzéséhez, helyességük ellenőrzéséhez, például hogy helyesen tartotta-e a karját a vizsgált személy egy adott pillanatban.

4.6. Eredmények



12. ábra
Emberi mozgás lekövetése



13. ábra
Ingamozgás lekövetése

Irodalomjegyzék

- [1] ismeretlen szerző: <http://www.xsens.com/en/company--pages/company/human-mocap/>
- [2] David P. Noonan, Peter Mountney, Daniel S. Elson, Ara Darzi and Guang-Zhong Yang: *A Stereoscopic Fibroscope for Camera Motion and 3D Depth Recovery during Minimally Invasive Surgery* <http://www.doc.ic.ac.uk/~pmountne/publications/ICRA%202009.pdf>
- [3] ismeretlen szerző: <http://hu.wikipedia.org/wiki/H%C3%A1romsz%C3%B6gel%C3%A9s>
- [4] ismeretlen szerző: <http://en.wikipedia.org/wiki/Triangulation>
- [5] ismeretlen szerző: http://hu.wikipedia.org/wiki/RGB_sz%C3%ADnt%C3%A9rt%C3%A9k
- [6] ELTE Informatikai Kara: *Informatikai Algoritmusok I.*; ELTE Eötvös Kiadó, Budapest (2005)
- [7] ismeretlen szerző: http://en.wikipedia.org/wiki/Alpha_compositing
- [8] ismeretlen szerző: http://en.wikipedia.org/wiki/RGBA_color_space
- [9] ismeretlen szerző: http://en.wikipedia.org/wiki/CMYK_color_model
- [10] ismeretlen szerző: <http://mediapedia.hu/cmyk>
- [11] Tamás, Járdán - Sándor, Pomaházi: *Adatszerkezetek és algoritmusok*; könyv, EKF Líceum Kiadó, Eger (2010)
- [12] GEDA, G., VIDA, J., MURÁNYI Z., B. TÓTH SZ., *How to study the phenomena of nature in the future (Multimédia a Kísérleti mérések szolgálatában)*, NETWORK SHOP 2005, Szeged (2005)
- [13] GEDA, G., VIDA, J., *Observation of mechanical movements through virtual experiments*, 6th International Conference on Applied Informatics, Eger (2004)
- [14] GEDA, G., VIDA, J., *Digitális tudásbázis és fizikai mérőkísérlet*, Agria Media, Eger (2004)
- [15] GEDA, G., VIDA, J., *Virtuális kísérletek mechanikai mozgások vizsgálatára*, 47. Országos Középiskolai Fizikatanári Ankét és Eszközkiállítást, Miskolc (2004)

- [16] GEDA, G., VIDA, J., *Virtuális kísérletek mechanikai mozgások vizsgálatára*, XXVIII. Országos Általános Iskolai Fizikatanári Ankét és Eszközkiállítást, Karcag (2004)
- [17] GEDA, G., VIDA, J., *Digitális oktatási segédanyagok fejlesztése (IHM-ITP-11): Mechanikai mozgások vizsgálata interaktív módon, virtuális kísérletekkel*, Informatikai és Hírközlési Minisztérium (2003)
- [18] CSAPÓ, B., *A tantárgyakkal kapcsolatos attitűdök összefüggései*, MAGYAR PEDAGÓGIA **100/3** (2000) 343-366.
- [19] FERNENGEL, A., *A kémia tantárgy helyzete és fejlesztési feladatai*, Új pedagógiai szemle **52** (2002) 68-82.
- [20] HADHÁZY, T., SZABÓ, Á., *Általános iskolai tanulók véleménye a fizikaoktatásról*, Fizikai Szemle **46** (1996) 166.
- [21] JÓZSA K., LENCSEŠ GY., PAPP K., *Merre tovább iskolai természettudomány?*, Fizikai Szemle **46** (1996) 167.
- [22] SZALAY, B., SZEPEŠI, I., *A matematika- és természettudományoktatásról TIMSS 2007*, Új Pedagógiai Szemle **1** (2009) 3
- [23] VÁTI P., BÁNFI I., FELVÉGI E., KROLOPP J., RÓZSA C., SZALAY B., *A PISA 2000 vizsgálatról*, Új Pedagógiai Szemle 51/12 (2001)
- [24] *Nemzeti alaptanterv: 202/2007. (VII. 31.) rendelet a Nemzeti alaptanterv kiadásáról, bevezetéséről és alkalmazásáról szóló 243/2003. (XII. 17.) Korm. rendelet módosításáról* (2007)
- [25] DEQING SUN, STEFAN ROTH J.P. LEWIS, MICHAEL J. BLACK *Learning Optical Flow*