



DIPLOMATERVEZÉSI FELADAT

Ureczky Bálint

szigorló mérnök-informatikus hallgató részére

Kiterjesztett valóság alkalmazása Android platformon

Napjainkban az okostelefonok számítási kapacitása elérte azt a szintet, hogy *realtime*, azaz valós időben vagyunk képesek képfeldolgozó algoritmusokat futtatni a telefon kamerája által készített előnézeti képen, valamint képes 3D-s objektumokat kirajzolni. Az *Augmented Reality* (AR, kiterjesztett valóság) alapötlete egy objektumnak és annak a pozíciójának a felismerése, aminek megfelelően 3D-s objektumokkal terjeszthetjük ki a valóságot a telefon képernyőjén keresztül tekintve.

A hallgató feladata egy olyan Android alapú kiterjesztett valóság alkalmazás tervezése és prototípusának megvalósítása, ami újszerű módon alkalmazza a valóság virtuális kiterjesztését.

A diplomatervnek a következőkre kell kiterjednie:

- Mutassa be a mai okostelefonok tulajdonságait, bennük rejlő képességeket, indokolja meg, hogy miért esett a választása az Android-ra.
- Ismertesse az AR alapötletét, működését és alkalmazási példáit.
- Ismertessen egy AR toolkitet és annak használatát, készítsen egyszerűbb példaprogramokat.
- Dolgozzon ki egy újszerű alkalmazást a valóság virtuális kibővítésére.
- Ismertesse a megvalósított rendszer felépítését, az alapvető algoritmusokat!
- Mutassa be a megvalósított rendszer működését és tesztelje használhatóságát!

Budapest, 2011. február

Dr. Forstner Bertalan
egyetemi docens
diplomaterv felelős

Dr. Vajk István
egyetemi tanár
tanszékvezető

HALLGATÓI NYILATKOZAT

Alulírott *Ureczky Bálint*, a Budapesti Műszaki és Gazdaságtudományi Egyetem szigorló hallgatója felelősségem tudatában kijelentem, hogy jelen „Kiterjesztett valóság alkalmazása Android platformon” diplomatervet meg nem engedett segítség nélkül, saját magam készítettem. Csak a megadott forrásokat (szakirodalom, eszközök, stb.) használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más nyomtatott, vagy elektronikus forrásból átvettettem, egyértelműen megjelöltetem, azok hivatalos adatait – az idézés forrását és az ott megjelölt szerző(ke)t – az irodalomjegyzékben megadtam.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik.

Budapest, 2015.05.15

Ureczky Bálint
hallgató



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

DIPLOMATERV

Kiterjesztett valóság alkalmazása Android platformon

Ureczky Bálint

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2015

Tartalomjegyzék

A dolgozat felépítése	1
Kivonat	3
Abstract	4
1. Bevezetés	5
1.1. Technológiai hármaspont	5
1.2. Kiterjesztett valóság	8
1.2.1. Valóság-virtualitás kontinuum	10
1.2.2. Látvány alapú AR	11
1.2.3. Orientáció alapuló AR	14
1.2.4. Pozíció és orientáció alapuló AR	15
1.2.5. Geometria alapú AR	16
1.3. Problémavezetés	17
2. Elemzés	19
2.1. Mobil-platformok	19
2.1.1. Android	21
2.1.2. iOS	24
2.1.3. Microsoft	24
2.1.4. További platformok	25
2.1.5. A választott platform	25
2.2. Fejlesztőeszközök	26
2.2.1. AR toolkitek	26
2.2.2. CV toolkitek	28
2.2.3. Grafikus megjelenítők	29
2.3. A választott módszerek	31
3. Égitest felismerés alapú kalibráció	32
3.1. Naprendszer modell	32
3.1.1. Égitest előrejelzés	34
3.1.2. Napi idő meghatározás	34
3.1.3. Északi irány meghatározása	34
3.1.4. Dátum meghatározása	35

3.1.5. Helymeghatározás	36
3.2. Eltérések	36
3.2.1. Mágneses eltérés	36
3.2.2. Gravitációs eltérések	38
3.2.3. Légköri refrakció	39
3.3. Hold	40
3.3.1. Holdfázis	40
3.4. Algoritmus	41
3.5. Égitest detekció	41
3.6. Távolság számítás	41
3.6.1. Sík távolság	42
3.6.2. Gömbi távolságképlet	42
3.6.3. Ellipszoid távolság	44
3.6.4. Implementáció	44
3.7. A program bemutatása	44
3.7.1. Főképernyő	44
3.7.2. AR nézet	45
3.7.3. Mérési eredmények nézet	46
3.7.4. Térkép nézet	46
3.8. Értékelés	46
3.9. Továbbfejlesztési lehetőségek	47
4. Ortofotó transzformáció	49
4.1. Problémafelvetés	49
4.2. Perspektív torzulás	53
4.3. Perspektív rektifikáció	56
4.4. Ortofotó transzformáció kiszámítása	59
4.4.1. Probléma megfogalmazása	61
4.4.2. Forgatás (rotáció)	63
4.4.3. Eltolás (transzláció)	66
4.4.4. Középpontos vetítés (projekció)	67
4.4.5. Az eredő síktranszformáció	70
4.5. Smart Zoom	72
4.6. Bemutatás	74
4.7. Alkalmazási lehetőségek	75
4.8. Továbbfejlesztési lehetőségek	75
5. Összefoglalás	77
Függelékek	79
A CD-melléklet tartalma	81

B A dolgozat elektronikus változata	83
B.1. PDF megjelenítő	83
B.2. Könyvjelzők	83
B.3. Hiperlinkek	84
C Felhasznált eszközök, programok	87
D Matematikai háttérísmerekek	89
D.1. Jobbkéz szabály	89
D.2. Lineáris transzformációk	89
D.2.1. Aktív transzformáció	89
D.2.2. Passzív transzformáció	90
D.2.3. Transzformáció más koordinátarendszerben felírva	91
D.3. Homogén koordináták	92
E Android hibák	93
E.1. RotationVector új paramétere	93
E.2. Remap 4x4	93
E.3. Fókusztávolság	93
E.4. XXXHDPI ikon	93
F Rövidítések	95
Ábrák jegyzéke	97
Táblázatok jegyzéke	101
Kódok jegyzéke	103
Irodalomjegyzék	105
Könyvek	105
Toolkitek	105
Applikációk	106
Információk	107
Statisztikák	108
Tárgymutató	109

A dolgozat felépítése

A dolgozat nyomtatott és elektronikus formában is elérhető. Utóbbi használatáról (pl. hogy az alábbi fejezetcímek hiperlinkekkel is működnek) részletesebben lásd a **B. függeléket**.

Mivel a szakszavak magyar fordítása esetenként kevésbé ismert és használt, ezért a dolgozatban igyekszem a fontosabb kifejezéseknek az angol megfelelőjét is feltüntetni zároljelben. A programok kódjában és elnevezésekben is az angol szóhasználatra törekszem.

A fejezetek ismertetése

A dolgozat **fedőlapja** előtti részben található a **feladatkiírás**, melyben a diplomatervként kiadott feladat részletezése és az elvárt tartalmi követelmények szerepelnek. Utána következő **hallgatói nyilatkozat**ban a munka egyéni megvalósítása kerül kinyilatkoztatásra és hogy minden módszer és forrásmegjelölés az előírásoknak megfelelően történt.

Ezután található a **tartalomjegyzék** a fejezetek listájáról, ami a dolgozat felépítését vázolja. Ezt egészíti ki a **jelen fejezetben** olvasható rövid leírások a fejezetekről. Ezután egy magyar nyelvű összefoglaló következik: a **kivonat** és ennek az angol nyelvű fordítása: az **abstract**.

- Az első **Bevezetés** c. fejezet a technika és mobil-készülékek fejlődéséről, majd az Augmented Reality megjelenéséről és fajtairól, végül pedig a megvalósítandó feladatról és a probléma felvetéséről szól.
- A 2. **Elemzés** c. fejezetben a mobil-platformok és különböző fejlesztőeszközök elemzése olvasható, majd megállapításra kerül a pontos fejlesztőkörnyezet, programozási nyelvek, felhasznált technológiák, amiknek a részleteit a C. függelék egészíti ki.
- A 3. **Égitest felismerés alapú kalibráció** c. fejezetben csupán a szenzorok közvetlen használatának egy újfajta alkalmazhatóságára láthatunk példát, miként használhatjuk a Nap/Hold helyzetét irány-, vagy akár hely-, időmeghatározásra készülékünk segítségével.
- A 4. **Ortofotó transzformáció** c. fejezetben a perspektíva rektifikálás módszerének és alkalmazásainak ismertetése következik, miként használható AR alkalmazásokban. A telefonos szkenner-alkalmazások elemzésével felvetődik a perspektíva automatikus, szenzor alapú rektifikálása, melynek matematikai levezetése is itt olvasható.
- Az 5. **Összefoglalás** c. fejezetben pedig az értékelés, eredmények és továbbfejlesztési lehetőségek olvashatóak.

A dolgozat végén található **Függelékek** szakaszban található:

- **CD-melléklet tartalma:** A CD-n található fájlok mappaszerkezete és leírása.
- **A dolgozat elektronikus változata:** A dolgozat CD-mellékleten megtalálható elektronikus változatában könyvjelzők, dokumentumon belüli és webes linkek segítik az olvasást. Ezek használata, beállítása olvasható ebben a mellékletben.
- **Matematikai háttérismerekek** Itt található néhány, a dolgozat matematikai vezetéseihez szükséges téTEL, fogalom ismertetése.
- **Hibajelentések** A dolgozat készítése során talált hibák, hiányosságok jegyzéke.
- **Felhasznált eszközök, programok:** A fejlesztés, dokumentálás és tesztelés során felhasznált eszközök, programok listája, pontos verziók megjelölésével.

Legutoljára pedig a különböző jegyzékek listája következik:

- **Rövidítések:** A dolgozatban előforduló rövidítések, mozaikszavak feloldása.
- **Ábrák jegyzéke:** A dolgozatban szereplő ábrák listája, oldalszámukkal együtt. Itt szerepel a képek készítésének módja, a felhasznált adatok hivatkozása, vagy magának a képfájlnak a forrása is.
- **Táblázatok jegyzéke:** A táblázatok listája, oldalszámukkal.
- **Kódok jegyzéke:** A dolgozatban előforduló kódrészletek listája, oldalszámukkal.
- **Irodalomjegyzék:** A hivatkozott források jegyzéke, kategóriákba rendezve.
- **Tárgymutató:** A fontosabb kifejezések, technológiák neveinek előfordulása a dolgozatban.

Kivonat

Napjainkban az okostelefonok számítási kapacitása elérte azt a szintet, hogy valós időben vagyunk képesek képfeldolgozó algoritmusokat futtatni a telefon kamerája által készített előnézeti képen, valamint 3D-s objektumokat is kirajzolni.

A *kiterjesztett valóság* (Augmented Reality, AR) alapötlete a környezet valamilyen szintű felismerése, aminek megfelelően 3D-s objektumokkal terjeszthetjük ki a valóságot a telefon képernyőjén keresztül tekintve.

Ez a felismerés többféle módon történhet. A telefon szenzorainak (GPS, gyorsulásmérő, giroszkóp, iránytű) segítségével meghatározhatjuk a pontos helyzetünket, nézeti irányunkat, de lehetőség van különböző objektumoknak a képi felismerésére is, amelyek nézetéből kiszámolható a relatív pozícióink és nézetünk.

Diplomatervem során először a mobiltelefonok fejlődésén keresztül felvázolom a kiterjesztett valóság téma körét és néhány jelenlegi alkalmazását. Ezt követően bemutatom az Androidban elérhető szenzorokat és kamerakezelési lehetőségeket. Ezután ismertetem a perspektív rektifikáció fogalmát, alkalmazási példáit, miként lehet használni AR programknál. Megvizsgálom a telefonos megvalósításokat – a szkenner alkalmazásokat és egy új módszer alkalmazását tűzöm ki célul, a perspektíva automatikus, szenzor alapú rektifikálását. Ismertetem az ehhez szükséges matematikai levezetést, majd bemutatom az implementáció során felmerült problémákat és a választott megoldásokat. Végül bemutatom az elkészült programot és eredményeket.

Kulcsszavak: *Kiterjesztett valóság, Android, képfeldolgozás, szenzorok, fényképezőgép, perspektív rektifikáció, madárnézet, trapéztorzítás, OpenCV, képfeldolgozás, kép illesztés*

Abstract

Nowadays, the computing power of smartphones have reached the level to execute image processing algorithms using the preview image of it's camera and draw 3D images in realtime.

The basic idea of the *Augmented Reality* (AR) is to extend the reality through the camera by identifying the surroundings. It can happen in several ways. With the sensors (GPS, accelerometer, gyroscope, magnetometer) we can identify our exact position, direction and view. We can also recognise objects via image processing which can lead us to calculate our relative position and direction to it.

During my thesis, I am introducing the topic of the Augmented Reality through the evolution of mobile devices and show some current applications. I introduce the camera and sensor API of Android and demonstrate my solution for handling the sensors easier for AR applications. I introduce the perspective rectification and its applications, how can it be useful for AR applications. I examine the mobile applications, the scanner applications and set a new method as an aim, how can we rectify the perspective view automatically using only the sensors. I present the mathematical derivation for this, show the difficulties of implementation and the choosen solutions. Finally, I present the complete application and its results.

Keywords: *Augmented reality, Android, sensors, camera, computational photography, perspective rectification, birdseye view, keystone effect, keystone correction, OpenCV, image processing, image stiching*

1. fejezet

Bevezetés

1.1. Technológiai hármaspont

Az elektronika rohamos fejlődésével, a miniatűrizálás egyre nagyobb sikereivel, ahogyan a tornateremnyi számítógépek szerepét doboznyi *PC-k*¹, majd hordozható *notebook-ok*² és *ultrabook-ok*³ vették át, úgy a kezdetben csak telefonálásra alkalmas telefonok számítási komplexitása és felhasználhatósági köre egyre csak bővült. A *feature-phone-ok*⁴ megjelenését követően végül a két technológiai vonal egy közös állapotot ért el, az *okostelefon-ok*⁵ és *táblagép-ek*⁶ világát (lásd 1.1. ábra). A számítási sebesség elérte azt a szintet, hogy komolyabb, akár valós idejű (real-time) számítások is elvégezhető rajtuk, nagyjából már csak a kijelző méretében és hordozhatóságukban különböznek ezek az eszközök egymástól.

Ezáltal, hogy a hordozható eszközök szegmensére ekkora számítási kapacitás került (illetve a vezeték nélküli hálózatok fejlődésével még nagyobb távoli kapacitások érhetőek el Interneten keresztül) egészen új lehetőségek nyíltak meg, főként az éppen ezért elterjedt szenzorok (pl. a kamera, *gyorsulásmérő*, giroszkóp, iránytű, GPS) beépítésével.

Ez a technológiai hármaspont tette lehetővé a következő fejezeben bemutatott *kiterjesztett valóság* térhódítását: A rengeteg beépítésre kerülő alkatrész egyáltalán a lehetőség megteremtését, a technológiai fejlődés a számítási kapacitás növekedésével a valósidejű, életszerű használatot, a miniatűrizáció pedig a hordozhatóságot teremtette meg. A technológia ugyanis létezett már korábban is, de a számítógép webkameráján keresztül ez inkább még csak érdekesség volt, mint igazán hasznos újítás.

¹ **PC:** Personal Computer, azaz személyi (asztali) számítógép.

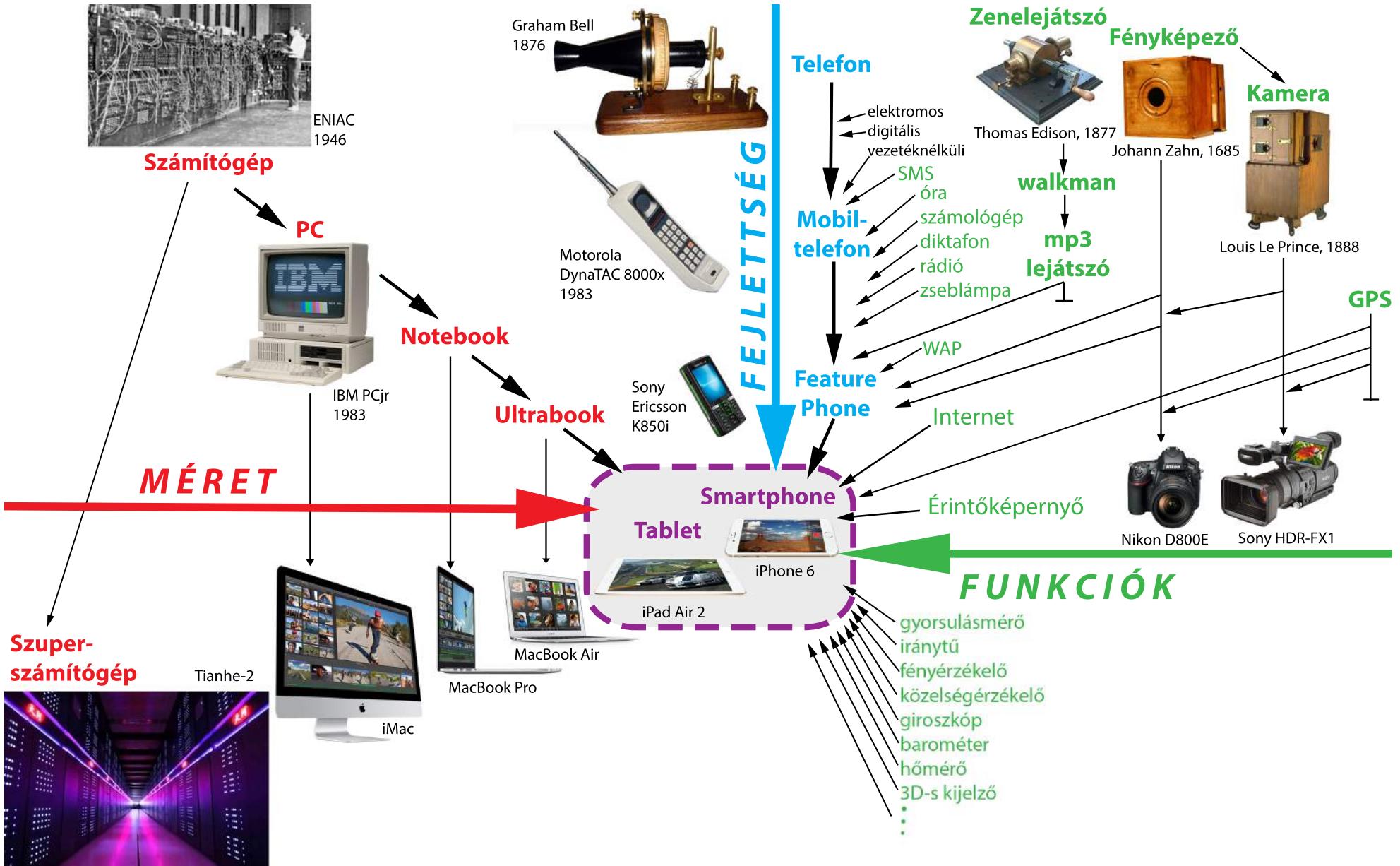
² **notebook**, vagy **laptop**: Kisméretű, hordozható személyi számítógép. Teljes értékű PC, lényegi különbség a hordozhatóságban rejlik. Angol jelentése szerint jegyzetfüzet (notesz), ami a kis méretre és a szövegbevitelre utal, a tulajdonos a hóna alá csapva, vagy hord-táskában viszi magával.

³ **ultrabook:** az Intel által definiált és levédett elnevezés, így ezt a kifejezés nem is használható más processzorral rendelkező gépek esetében. Mi viszont most általában a kis méretű, könnyű súlyú és vékony kivitelű számítógépekre gondolunk (SSD meghajtóval, optikai meghajtó nélkül), mint amilyen például az Apple Macbook Air is. Sajnos nem igazán létezik rá pontos kifejezés. A *subnotebook* kategóriába ugyanis minden notebooknál kisebb méretű számítógép beletartozik, a *táblagépek* és *PDA-k* is. Ide tartozik még a *netbook* elnevezés is, de az eredetileg csak netezésre alkalmas – nem teljes értékű – számítógépet jelent.

⁴ **feature-phone:** olyan középkategóriás mobiltelefon, ami az egyszerű telefonáláson kívül más funkcióra is képes (pl. fényképezés, böngészés), de még nem *okostelefon*.

⁵ **okostelefony** (smartphone): Nincs pontos definíciója, általában az érintőkijelző, szenzorok és korszerű operációs rendszer különbözteti meg a *feature-phone*-nak nevezett középkategóriás telefonuktól.

⁶ **táblagép:** Olyan érintőkijelzős PC, mely lényegében csak egy kijelzőből áll.



1.1. ábra. Az okostelefonok és táblagépek világának kialakulása a számítógépek miniatúrizálódása és a mobiltelefonok fejlődése révén létrejött határterület megjelenéseként.

A **kék színű nyíl** mutatja a **technológiai fejlődés** irányát (fentről lefelé), ahogyan az első elektronikus számítógépből mára hasonló (vagy még nagyobb) méretekben már szuperszámítógépeket építenek, az első távbeszélőből kifejlődött a mobiltelefon, az első fotokémiai elven működő fényképezőből pedig a digitális tükrökreflexes DSLR fényképező.

A **piros színű nyíl** mutatja a **miniatúrizáció** irányát (balról középre), ahogyan a számítógépek egyre kisebbek és kisebbek lehetnek azonos számítási kapacitás mellett: az eleinte szoba nagyságú számítógépekből otthoni használatú PC lett, majd hordozható notebook, végül még kisebb és még kompaktabb, ami mégis körülbelül ugyanazt a szolgálatot képes teljesíteni.

A **zöld színű nyíl** mutatja a telefonhoz **hozzáadott funkciók** sokaságát (jobbról középre). Ahogy a telefon digitálissá vált, úgy épültek be az extra szolgáltatások, mint a névjegyzék, óra, naptár, számológép, majd hardver elemek, mint diktafon, rádió, zenelejátszó, zseblámpa, fényképező, kamera, érintőképernyő és különféle egyéb szenzorok. Először csak a fényképezőkben is használt gyorsulásmérő (az orientáció meghatározására) és GPS, majd fényérzékelő, távolságérzékelő, iránytű, giroszkóp, de akár barométer, vagy hőmérő is található egyes készülékekben. Néhányuk már 3 dimenziós megjelenítésre alkalmas kijelzővel rendelkeznek.

Az ábrán látható elemek:

SZÁMÍTÓGÉPEK:

ENIAC: (Electronic Numerical Integrator And Computer) 1946.

Az első programozható, elektronikus, digitális „számító”-gép. 30 tonna súlyú, másodpercenként kb. 5.000 összeadás, 357 szorzás, vagy 38 osztásra volt képes. [39]

IBM PCjr: (PC Junior) 1983.

Az első otthoni és iskolai használatra szánt számítógép.

3,71kg (+ monitor + billentyűzet),

4,77 MHz, kb. 240.000 művelet másodpercenként. [34, 38]

Tianhe-2: Korunk legnagyobb teljesítményű (33,86 PFLOPS) szuperszámítógépe 2013 június óta. [44]

Apple iMac: Korunk egyik legjobb otthoni felhasználásra szánt számítógépe

Apple Macbook Pro: Korunk egyik legjobb notebookja

Apple Macbook Air: Korunk egyik legjobb ultra-hordozható notebookja

Apple iPad Air 2: Korunk egyik legjobb táblagépe

TELEFONOK:

Graham Bell (1876): Első távbeszélő készülék feltalálása. [41]

Motorola DynaTAC 8000x (1983): Az első kereskedelmi forgalomban megjelent mobiltelefon. 0,8kg súlyú.

Működő mobiltelefon már 1973-ban volt. [35]

Sony Ericsson K850i: Tegnapunk egyik feature-phone-ja.

Apple iPhone 6: Korunk egyik legjobb okostelefonja.

FUNKCIÓK:

Thomas Edison (1877): fonográf, az első hangrögzítő és visszajátszó szerkezet. [37]

Johann Zahn (1685): Az első hordozható méretű fényképező szerkezet elkészítése. [43]

Louis Le Prince (1888): cinematográf, az első kamera és visszajátszó szerkezet. [40]

Nikon D800E: Korunk egyik nagyon jó DSLR fényképezőgépe.

Sony HDR-FX1: Korunk egyik nagyon jó videókamerája.

1.2. Kiterjesztett valóság

A *kiterjesztett*, vagy *kibővített valóság*, angolul *augmented reality* (röviden AR) napjaink digitális kiegészítése a világnak. Ennek során a készülékben elhelyezett szenzorok (GPS, gyorsulásmérő, giroszkóp, iránytű, kamera) felelősek a tényleges, vagy relatív pozíció és orientáció meghatározásért, aminek segítségével az eszköz kamerája által látott kép a manapság elterjedt nagy kijelzőkön keresztül tekintve digitális tartalmakkal, objektumokkal kiegészítve egy kibővített valóságot mutat.

Olyan, mintha egy mesebeli varázsszemüveget vennénk fel, amivel különleges dolgokat láthatunk. Átláthatunk a falakon, ahogyan egy röntgengéppel vagy ultrahanggal a test mélyébe láthatunk. Éjjellátó készülékkel még akkor is látunk, amikor pusztta szemmel már korom sötét lenne (1.2a. ábra). Infrakamerával láthatjuk a környezetünk hőterképét (1.2b. ábra), melyre a mi szemünk nem alkalmas, viszont számos kígyófaj rendelkezik erre alkalmas érzékelőgödrökkel.

De a képzelőerő gazdagságára a science fiction és varázslóvilágokból vehetjük a legjobb példákat. A *Harry Potter* című regénysorozatban is gyakran olvasható, hogy valaki olyat lát, amit más nem. Például a lószerű *thesztrál*-okat csak olyanok láthatják, akik már láttak valakit meghalni. *Edevis tükrébe* (1.2d. ábra) tekintve pedig egyfajta képzeletbeli világba csöppennek, ahol saját magát mindenki úgy látja, ahogyan álmaiban szeretné, azt látja, amire vágyik (az ábrán Harry Potter például elhunyt szüleit látja a vallai fölött). Harry híres köpönyege pedig épphogy láthatatlanná teszi viselőjét. Viszont *Alastor Mordon* mágikus szeme minden átlát 1.2c – még a láthatatlanná tévő köpönyegen is. A *tekergők* térképe az iskola alaprajza felett pedig egy olyan extra rétegét mutat, amin valós időben követhető nyomon, hogy ki hol tartózkodik éppen. A *Doctor Who* c. sorozatban a *TARDIS* minden létező nyelvet képes lefordítani és ha a közelében tartózkodunk automatikusan megértyük az összes nyelvet, gond nélkül el tudjuk olvasni az idegen nyelvű szövegeket.

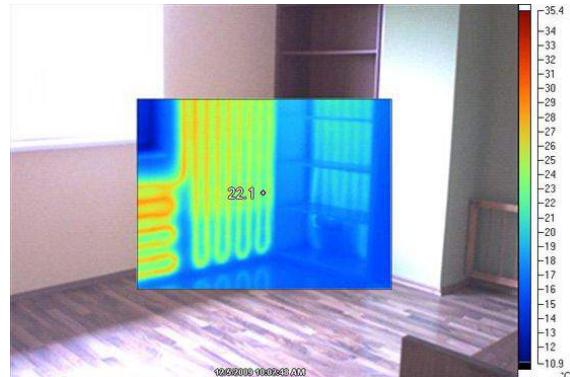
Nagyon érdekes, otthon is kipróbálható jelenség, hogy egy olcsóbb⁷ fényképezőgép, vagy telefon kameráján keresztül megfigyelve a távirányító elejét világítani látjuk miközben nyomogatjuk a gombokat (1.2e. ábra). A távirányító ugyanis infravörös sugárzással továbbítja a jeleket, ami valójában ugyanolyan elektromágneses sugárzás, mint a fény, csak szemünk már nem érzékeli, a fényképezőgépek szenzora viszont igen⁷. Bankjegyeken is megfigyelhetünk csak UV-fényben látható vízjeleket (1.2f. ábra).

Az AR lényegében erről szól: láthatóvá tenni a láthatatlant, megnyitni a rejtegeket, kiegészíteni vagy éppen megváltoztatni azokat. Egy képzeletbeli „álomvilág”, amiben olyan ötleteket valósíthatunk meg, azt láthatjuk, amit szeretnénk. Nem csoda, hogy számos alkalmazás született már rá, hiszen rengeteg lehetőség rejlik a módszerben. Segítségével megtapasztalhatjuk a szuperhősök és sci-fik világába illő képességeket. A *TARDIS*-hoz hasonlóan a *Word Lens* alkalmazással (1.2h. ábra) anyanyelvünkön olvashatjuk a szövegeket, a *Vuforia* occulsion managementjével (1.2g. ábra) röntgenlátás szerű hatás érhető el, a *Layar* alkalmazás (1.11. ábra) pedig napjaink *Tekergők* térképe. Ha a vetített hologramok ideje még nem is jött el, telefonunkkal hasonló élményben lehet részünk.

⁷Komolyabb fényképezőgépek lencséjére szándékosan raknak infraszűrőt, a valósághelyesebb felvételért.



(a) Éjjellátó készülék



(b) Hőkamerán megfigyelt falfűtés



(c) Alastor Mordon mágikus szemével mindenek átlát



(d) Edevis tükrében mindenki azt látja, amire vágyik



(e) A távirányító infraadójának fénye kamerán keresztül nézve láthatóvá válik



(f) A magyar bankjegyeken csak UV-fényben látható vízjelek



(g) A Vuforia occlusion management-tel áttetszővé tehető egy felismert kép

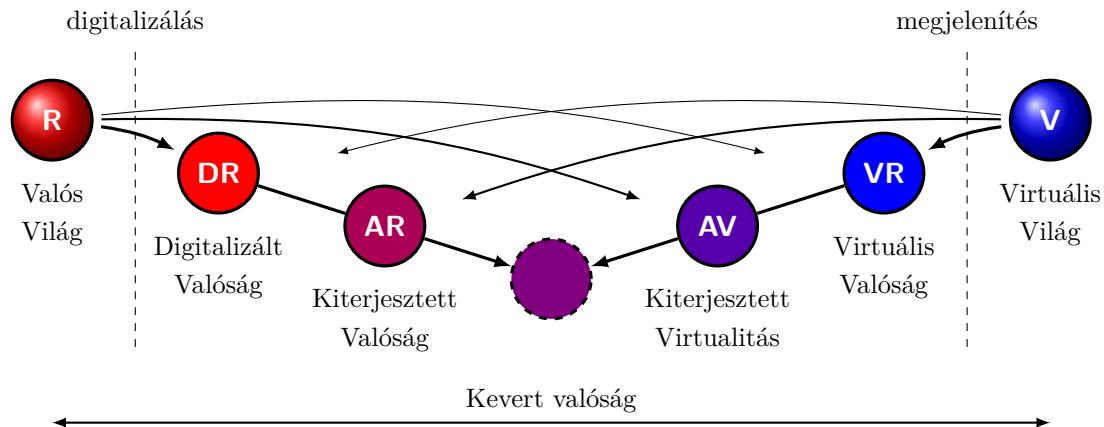


(h) A Word Lens alkalmazás valós időben fordít képen felismert szöveget és kicseréli a kameraképen

1.2. ábra. Látni a láthatatlant – Kiterjesztett valóság a csúcstechnológiában, science fiction és varázslat világában, hétköznapokban és mobil alkalmazásokban

1.2.1. Valóság-virtualitás kontinuum

A valóságot és virtualitást különböző mértékekben vegyíthatjuk egymással (1.3. ábra), melyet általánosan *kevert valóságnak* (Mixed Reality, MR) hívunk. Az egyik véglet a valóság pusztta digitalizálása (például fényképek, videók formájában). A másik véglet a virtuális tartalom pusztta megjelenítése, CGI⁸ képek, animációs filmek formájában. De minden két világot bővíthetjük (kiterjeszthetjük) a másikból vett elemekkel.



1.3. ábra. A valóság-virtualitás kontinuum

Ha például a virtuális világba belevisszük az irányításunkat, már úgynevezett *virtuális valóságnak* (Virtual Reality, VR) éljük meg, ilyenek a számítógépes játékok. Az irányításnak egy még valósághűbb formája a fejre szerelt kijelzők (Head Mounted Display, HMD) használata, melyek a fejmozgás követésével fejünk orientációját viszi a virtuális világba, ennek megfelelően mutatja a látott képet. Ennek a technológiának legismertebb úttörője az *Oculus Rift* által készített prototípus, melynek végső megjelenése 2016 elejére várható.

A virtuális tér kiegészíthető még való világból vett objektumokkal is, ez a *kiterjesztett virtualitás* (Augmented Virtuality, AV). Így készült például az *Avatar* c. film, melyben a karakterek mozgásai, arcvonásai, mimikái valódi színészektől származtak. Hasonlóképpen több játékban használhatjuk saját arcképünket karakterünk avatarjaként, vagy erről szól a *Mátrix* c. film is, ahol virtuális környezetben élt az emberek tudata.

Ha a másik irányból indulunk el, akkor a való világot terjeszthetjük ki digitális tartalommal, ez tulajdonképpen a *kiterjesztett valóság* (Augmented Reality, AR). A filmezés során ismert *greenbox* technikánál pl. a zöld háttéret kicsérlik egy digitális jelenettel.

A kiterjesztett valóságot kategorizálhatjuk tehát a kiterjesztés mértéke szerint. Van, hogy csak egyszerű információval egészítjük ki, vagy háttérrel, vagy épp 3D-s tartalmakkal.

Fontos viszont még az illeszkedés mértéke, mennyire illik bele a környezetébe, mennyire konzisztens. Ennek eléréséhez minél több információra van szükség a való világból. Így tehát aszerint is kategorizálhatunk, hogy milyen információkat használunk fel a világról. Ez lehet például a látott kamerakép, a készülék orientációja, pozíciója, vagy akár mélységtérkép, illetve ezek tetszőleges kombinációja. Ezeket a lehetőségeket mutatják be a következő alfejezetek.

⁸CGI: Computer Generated Imagery, számítógép által létrehozott kép

1.2.2. Látvány alapú AR

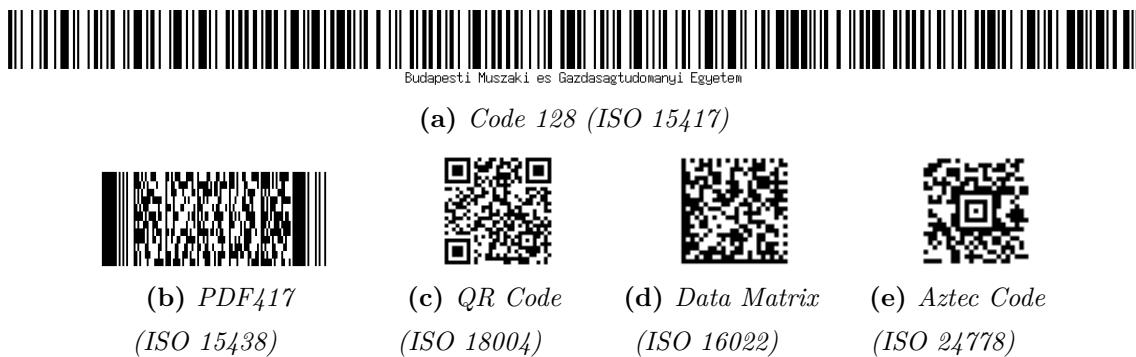
A látvány alapú (vision based) AR csak a látott kameraképet elemzi. A valóság kiterjesztésének mértéke alapján további 2 főbb típusba sorolható az szerint, hogy az azonosított célpontot követi-e, vagy sem.

Felismerés alapú AR

Van, hogy egy képrészletet csupán felismerni szeretnénk és ehhez kapcsolódó információkat szerezni.

Ez a kép gyakran egy *vonalkód*, vagy annak a speciálisan képi felismerésre továbbfejlesztett 2 dimenziós változata, egy általában fekete-fehér mátrix, ún. *pontkód*. Sokkal nagyobb információsűrűségű, így kevesebb helyet igényel (1.4. ábra), ezért nagyon elterjedt napjainkban, elsősorban a *QR-kód*⁹ szabvány (1.4c. ábra), de találkozhatunk még DataMatrix (1.4d. ábra), vagy Aztec (1.4e. ábra) kódolással is.

Általában plakátkon, újságokban találkozunk velük és leggyakrabban URL-t kódolnak vele, mert így kiváltja a hosszú begépelés szükségességét. De sok más típus szabványosított benne, mint a telefonszám, névjegy, WiFi hozzáférés, vagy akár csak egyszerű szöveg. A BKK jegyein és bérletein is ezt használják azonosítás és nyomonkövetés céljából. A MÁV-nál pedig az Interneten vásárolt, otthon nyomtatható jegy is QR-kódban tárolja az adatokat, amit a kalauz kamerás telefonnal kezel le.



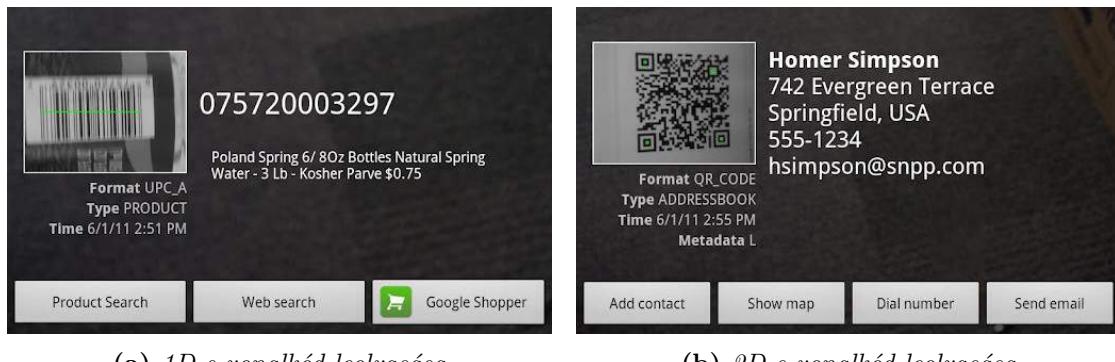
1.4. ábra. A „Budapesti Műszaki és Gazdasagtudományi Egyetem” kódolása különböző szabványú vonalkódokban.

Újságok gyakran készítenek saját alkalmazást képeik felismeréséhez, hogy kiváltsák a kevésbé esztétikus megjelenésű QR kódokat.

De vannak általános képeket, szöveget felismerő alkalmazások is (1.6. ábra). Ezek a jóval bonyolultabb felismerések általában már Internetes erőforrásra támaszkodnak, főleg a nagy számú keresési lista miatt, melyet nem lehet a készüléken tárolni.

Ez az AR típus tehát egy vonalkódot (1.5. ábra), vagy más képrészletet (1.6. ábra) csak felismer (akár Interneten keresztül), dekódol, majd megjeleníti a hozzá tartozó információt, vagy további interakciót nyújt, de nem igazodik a kép pozíójához.

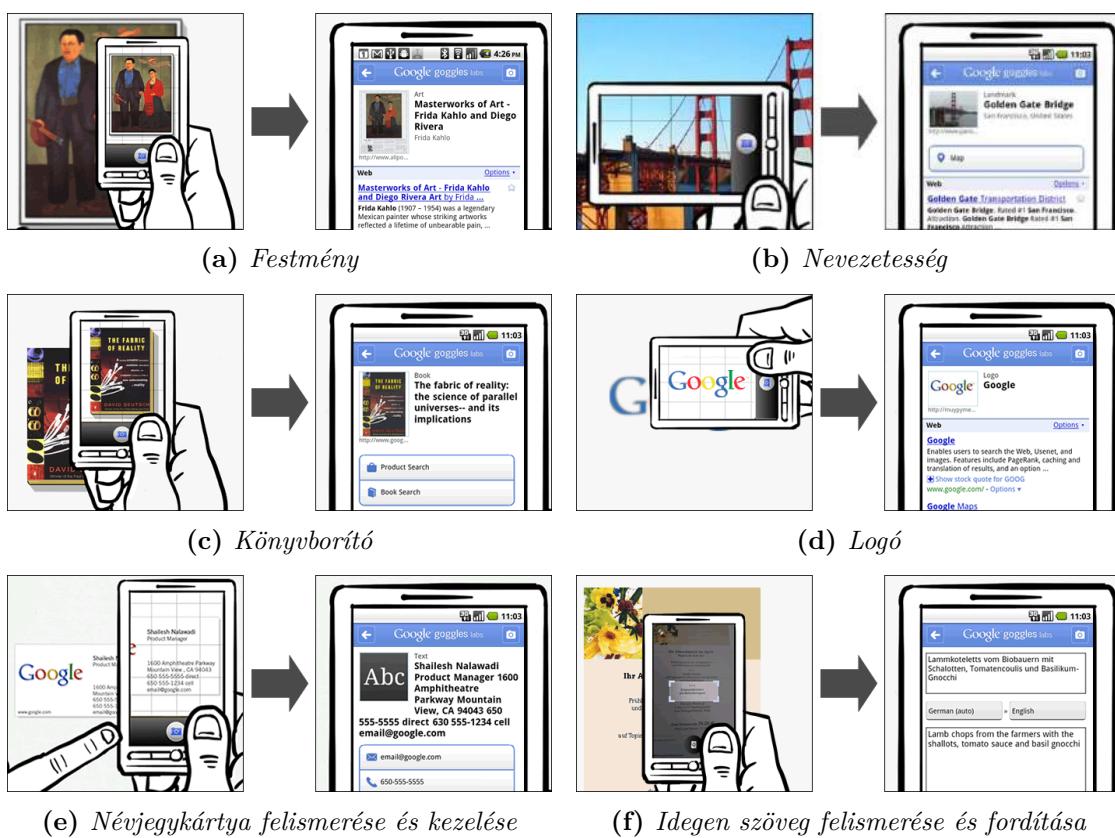
⁹ **QR-kód:** Quick Response, azaz gyors válasz kifejezésből származik, mert gyorsan visszafejthető, ami a felhasználónak is szempont, gyors interakció valósítható meg vele. A japán *Denso-Wave* cég fejlesztette ki 1994-ben. Előnye a skálázhatóságból ered, akár 2,953KB adat is eltárolható benne, 360°-ban bárhogyan leolvashatjuk és 7-30% hiba helyreállítására képes.[42]



1.5. ábra. *Zxing: Barcode Scanner*

A legismertebb, több platformra elérhető, nyílt forráskódú vonalkód leolvasó, ami nagyon sok egy (UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Code 128, ITF, RSS-14, RSS-Expanded) és több két dimenziós (QR Code, Data Matrix, Aztec(beta), PDF 417 (beta)) formátumot is felismer. A különböző típusú, szabványosított adatokhoz számos további interakciót is felajánl.

Google Play: [33], Toolkit: [20]



1.6. ábra. *Google: Goggles*

Ez az alkalmazás a Google vizuális keresője, alapjában véve egy többfunkciós on-line képfelismerő alkalmazás. Képes vonalkódokat, könyvbörítőt, festményeket, turisztikai látványosságokat, cégeket, emblémákat felismerni és járulékos információkat keresni hozzá interneten. Gyakorlatilag bármilyen vizuális adatot felismer, ami az adatbázisában szerepel és a lehetőségeket igyekeznek tovább bővíteni, például növényeket felismerni a leveleik alapján. Ezen kívül felismeri a névjegykártyákon található adatokat, amit a telefonba importálhatunk, képes külföldi szövegeket felismerni és lefordítani, valamint a Sudoku-feladványokat is felismeri és megoldja.

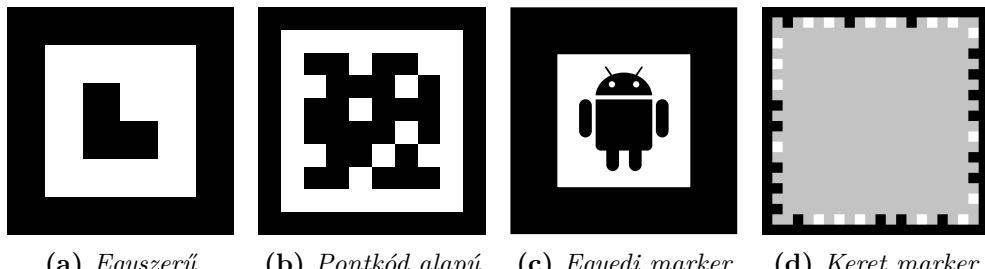
Google Play: [26]

Célpont követő AR

Az AR egyik legnépszerűbb típusa bizonyos célpontok (angolul *target*, vagy *tracker*) felismerésével és azok követésével képes a célpont körül térrészlet kiterjesztésére. Az ismert célpont kamerán keresztül látott képének változásaiból (méretváltozás, torzulás) kiszámítható az alakzat helyzete, ami így virtuálisan kiegészíthető. Gyakran használják újságokban, mert akkor a mintázatot már nem kell kinyomtatnia a felhasználónak és a mérete is biztosan megfelelő, így akár a valódi (1:1 méretarányban) is megjeleníthető a tartalom.

A leggyakoribb célpont az ún. *marker*, azaz jelölő, ami egy egyszerűen felismerhető, kontrasztos, általában fekete-fehér kép. Szükséges, hogy ne legyen forgásszimmetrikus, máskülönben megkülönböztethetetlenek lennének az elforgatott állapotok.

Leggyakrabban csupán egy pontmátrixot használnak a jó felismerhetősége miatt, de nagyon egyszerű szerkezetűt, ami kevés pontból áll (1.7a-1.7b. ábra), mert csak a limitált lehetőségek (általában 1-10) közötti megkülönböztetéshez szükséges azonosítót kell kódolnia, a hozzá kapcsolódó információkat már a program tartalmazza. Bár lehet redundáns, hogy részleges takarás esetén is azonosítható legyen. A négyzetes keret elősegíti a képen megtalálást. Kicsit idegen kinézete miatt igyekeznek egyedi mintákat használni (1.7c. ábra), vagy egy *keret marker* (*frame marker*) belséjébe már tetszőleges kép is nyomtatható.



1.7. ábra. Követésre alkalmas markerek

A kreatívabb, egyedibb felhasználásokhoz végül is bármilyen kontrasztos vagy részlet gazdag kép alkalmas lehet a követéshez, így például az IKEA alkalmazása saját katalógusának fedőlapját alkalmazza kép-célpontként (*image target*) (ld. 1.8. ábra).



1.8. ábra. Az IKEA új AR alkalmazása a katalógus fedőlapját kép-célpontként használva valós méretben jeleníti meg a bútorokat lakásunkban. Google Play: [21]

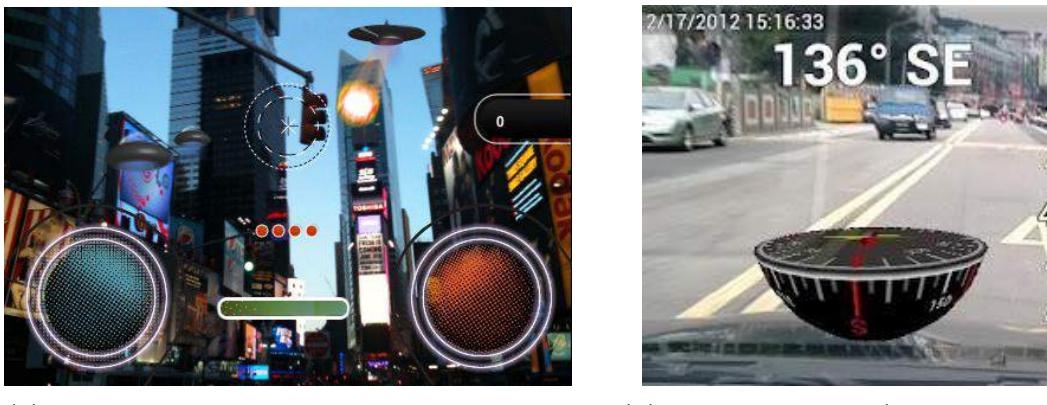
1.2.3. Orientáció alapuló AR

A készülék gyorsulásmérője és iránytűje segítségével meghatározható a telefon orientációja, mely változásának folyamatosságát a giroszkóp segítheti.

Így lehetőség adódik például a horizont kirajzolására, vagy 3 dimenziós iránytű elhelyezésére a térben (1.9b. ábra). Különböző lövöldözős játékok is születtek, amelyek a látott „térből” játszódnak (1.9a. ábra).

Ez a típus ugyan valósághűen, az iránynak megfelelően jeleníti meg a képen a kiterjesztést, de a kameraképet csak háttérként használja fel, így a tartalom nem illeszkedik szorosan a környezetéhez.

A panoráma készítő alkalmazások (1.10. ábra) már a látott képet is felhasználják, de csak a legvégén az illesztéshez, mert ez egy lassabb művelet.



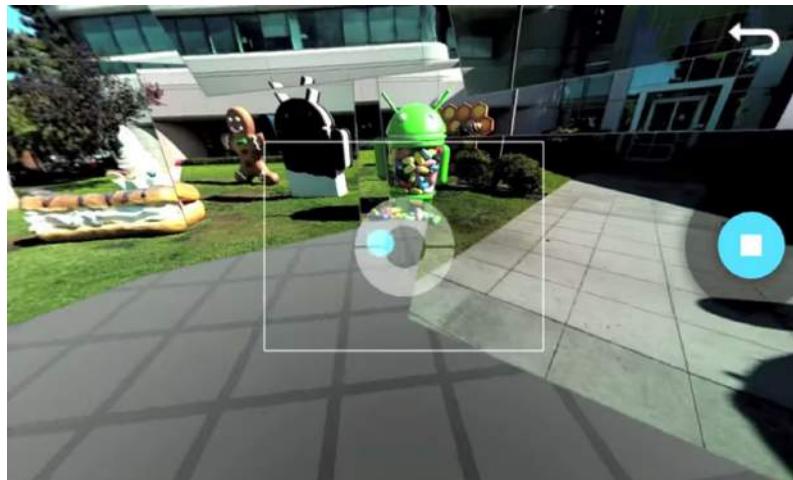
(a) *AR Invaders*

A telefonnal célozva vadászhatunk virtuálisan megjelenő UFO-kra. Google Play: [32]

(b) *3D Compass+ (AR Compass)*

A képernyőn megjelenő virtuális iránytű gömbkompassz módjára mutatja az északi irányt, nem kell vízszintesen tartani a kijelzőt. Google Play: [?]

1.9. ábra. Orientáció alapú alkalmazások



1.10. ábra. *Google: PhotoSphere*

Android 4.2-től elérhető gyári kameralkalmazásban lévő gömbpanoráma készítési lehetőség. Google Play: [25]

1.2.4. Pozíció és orientáció alapuló AR

A készülék orientációjának felismeréséhez ha hozzávesszük a GPS szolgáltatta pozíció információt, akkor már pontosan rendelkezésünkre áll, hogy a térben hol és merre nézünk.

A technológia felhasználásával reformizálható a navigáció, a kameraképre rajzolhatjuk ki, mi merre található. A *valóság-böngészőkkel* (1.11. ábra) a környezetünkben lévő POI¹⁰ jellegű objektumokat láthatjuk a kameraképen, melyik merre van, milyen messze. Az egyes elemekhez plusz információk, linkek tartozhatnak. Vannak még például régi fényképeket megjelenítő alkalmazások is, de minél közelebbi az objektum, annál kevésbé pontos a megjelenítés helye, mert a GPS nem ad elég pontos pontos helymeghatározást. Így ez a típus a minél távolabbi pontokat mutatja helyesen. Az égbolt megfigyelése jó példa erre, ilyen alkalmazásokkal a csillagok, Nap, Hold, vagy műholdak helyzetét nézhetjük meg kameraképen, vagy épp anélkül (1.12. ábra).



1.11. ábra. Layar Reality Browser

Az első és legismertebb valóság-böngésző, melyben különböző – akár saját magunk készített – rétegeket (AR layers) jeleníthetünk meg a képernyón, mint például hotelek, bárok, éttermek, ATM automaták, kórházak, tömegközlekedési megállók listáját. Google Play: [29] Toolkit: [14]



(a) Égbolt megtekintése

(b) Égitest keresése

1.12. ábra. Google: Sky Map

A Google által fejlesztett, majd 2012-ben nyílt forráskódúvá tett égbolt vizsgáló alkalmazás. A telefon orientációja, jelenlegi GPS koordinátánk és az idő (vagy átállíthatók más értékekre) alapján vetítíti a képernyőre a látott égboltot. Keresni is lehet egy adott égitestet, ekkor egy nyíllal vezet a megfelelő irányba. Google Play: [23], Forráskód: [22]

¹⁰POI: Point of Interest, azaz hasznos hely, érdekes pont.

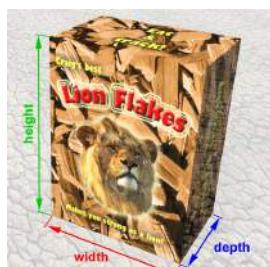
1.2.5. Geometria alapú AR

Az eddig ismertetett AR módszerekkel ugyan megjeleníthetők a kiterjesztett tartalmak, de csak egy rátételezett rétegként, ami még nem teszi képessé a más objektumokkal való interakcióra. A realisztikusság növeléséhez tovább kell finomítani a valóság és kiterjesztett tartalom kapcsolatát, amihez nem csak a digitális tartalom modellje, de a környezet geometriája is szükséges, hogy a takarások, átfedések megfelelőek lehessenek.

Az orientáció (és pozíció) alapú AR önmagában nem számol a képi információval, így a világot csak egy háttérként, egy merőleges síkként feltételezte, vagy a (gömb)panoráma kép készítő alkalmazások henger-, illetve gömbfelületként. Ráadásul ha így a szenzor adatokhoz illeszti a megjelenítést is, a képhez képest nem pontos az illeszkedése.

A látvány alapú AR-ek ezzel ellentétben pont a képhez illesztik a tartalmat, így ahhoz szorosabban illeszkedik és az ismert célpontok esetén azok síkját is fel tudja mérni (ismert méret esetén még a távolságot is).

Ha a célpontokról további geometriai előismeretünk van, akkor már megvalósítható egy szorosabb integráció. Ilyen lehet egy többszörös célpont (pl. téglalap oldalai), vagy henger célpont (1.13. ábra).



(a) Többszörös célpont, téglalap

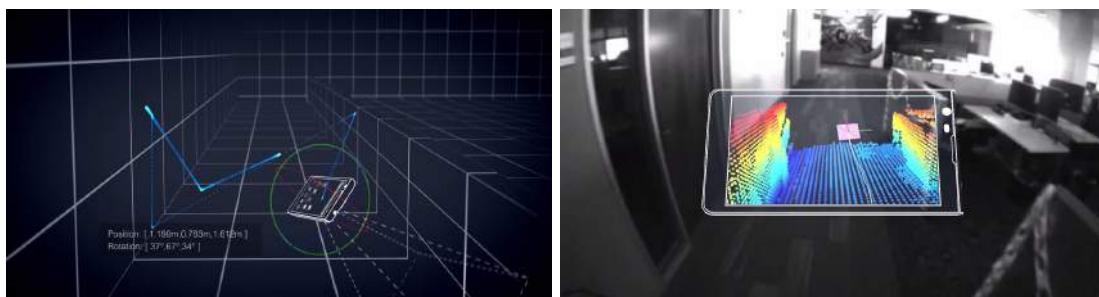


(b) Henger célpont



1.13. ábra. 3 dimenziós célpontok kiterjesztése takarás figyelembevételével.

Általános megoldást viszont csak a környezet geometriájának tényleges feltérképezése adhat. Ennek első lépcsőfoka a látott kép pontjaihoz tartozó távolságok (mélységtérkép) megléte, amiből egy 3D-s képet kapunk. Ha ezt több helyről is adott, akköl már tényleges 3D geometriákról is beszélhetünk, ezzel foglalkozik most a Google a kutatás alatt álló *Tango* nevezetű projektjében, ahol mélység-, és két optikai kamerával rendelkező telefonnal térképezik fel a környezet 3D-s modelljét valós időben (1.14. ábra). Már előrendelhető a Developer Kit.



1.14. ábra. Google: Project Tango [8]
Valós idejű 3D modellezés kétkamerás és mélysékgamerás telefonnal.

1.3. Problémafelvetés

Diplomatervem során új fajta megközelítéseket keresek kiterjesztett valósághoz kapcsolódóan, melyhez az orientáció alapú AR-t választottam kiindulásként.

A csak orientáció alapú AR-ek, mint az égbolt figyelő alkalmazások nem használják fel a látott képet. Megvizsgálom, hogy a látvány kiterjesztése helyett, annak felismerésével miként kalibrálható az egyik paraméter, azaz például a világító egitestek felismeréséből mennyire pontosan határozható meg földrajzi helyzetünk, orientációt, vagy a pontos idő.

A csak látvány alapú AR-ek ezzel szemben csak a relatív helyzetet mérik fel, nem szorítkoznak az orientációra. Megvizsgálom, hogy a sík célpont helyzetének ismeretében vagy annak feltételezésével miként javítható a látvány alapú AR-ek működése az orientáció felhasználásával.

Ezután a panorámakép készítő alkalmazások ilyen szintű degradációját is megvizsgálom, azaz, hogy a henger- és gömbpanoráma analógiájára hogy nézne ki egy „síkpanoráma” készítő alkalmazás, ami egy nagyméretű (nem merőleges helyzetű) sík egy részletének digitalizálását tenné lehetővé.

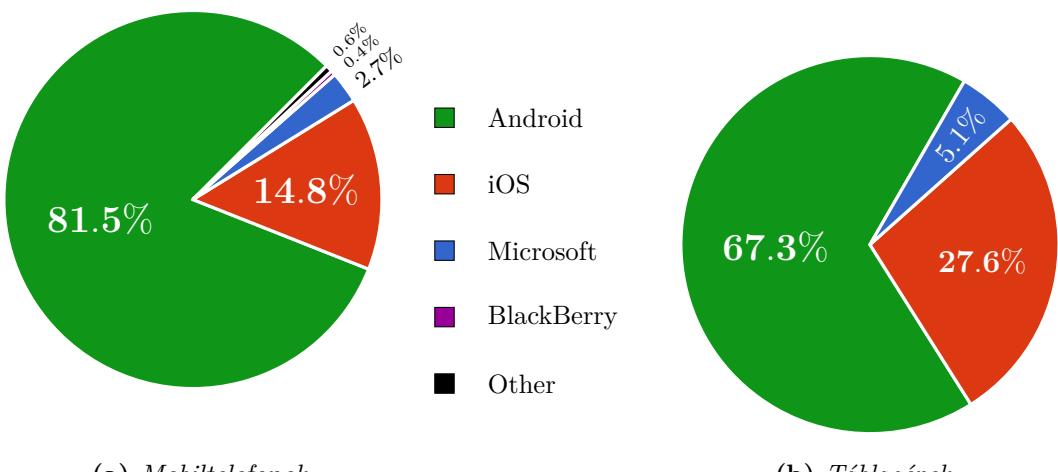
2. fejezet

Elemzés

Egy elkészítendő prototípus fejlesztése során célszerű először mérlegelni, hogy milyen *platform(ok)*ra¹ készüljön el, ugyanis ez is gátat szabhat – vagy éppen lehetőséget nyújthat – bizonyos funkciók használatára. A platform(ok) kiválasztásán túl a támogatott eszközöket pontosabban is definiálni kell az OS² verzió, kijelzőfelbontás szempontjából is, aminek legfőbb eleme az okostelefonok és táblagépek megkülönböztetése. Ezek a döntések szorosan összefüggnek a használni kívánt toolkit³(ek) megválasztásával is, ezért a következőkben ezen szempontok elemzése következik.

2.1. Mobil-platformok

A platform kiválasztása során az egyik fő szempont a jelenlegi trendeknek a vizsgálata (2.1. ábra). Hacsak nincs kifejezett igény egy adott platform választására, akkor érdemes a legelterjedtebbekre fejleszteni – a minél nagyobb piaci lefedettség érdekében.



2.1. ábra. Mobiltelefonok és táblagépek operációs rendszereinek világpiaci megoszlása a 2014-es évben eladtott készülékek viszonylatában (IDC, 2015).

¹ **platform:** olyan hardver- és/vagy a szoftverkörnyezet, amely meghatározza, hogy az adott készüléken milyen szoftverek futtathatóak.

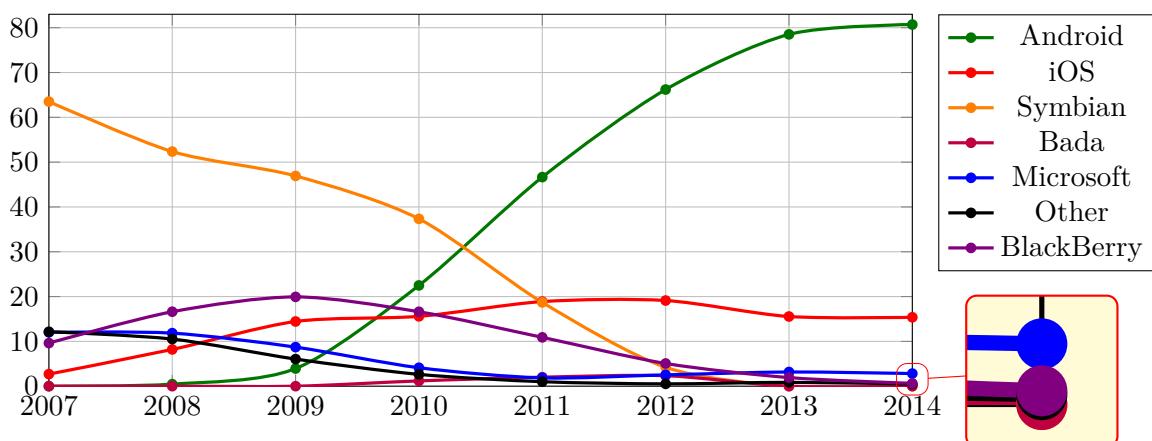
² **OS:** Operating System, azaz operációs rendszer. A hardverek és szoftverek kezelését, irányítását végző elsődleges szoftver a készüléken.

³ **toolkit:** segédkörnyezet (pl. osztálykönyvtárak), amely megkönnyíti bizonyos funkciók megvalósítását.

A 2.1. ábra az okostelefonok és táblagépek platform szerinti jelenlegi megoszlását mutatja. Látható, hogy az okostelefonok szegmensében (2.1a. ábra) az Android uralja a piacot majdnem 82%-kal, a táblagépek szegmensében (2.1b. ábra) viszont csak az utóbbi időben vette át vezető szerepét, sokáig az iOS volt a piacvezető. Már önmagában ennek a két platformnak a figyelembevételével az egész piac több mint 96%-a lefedhető, így ezek a legfontosabbak, ezekre történik manapság leggyakrabban a fejlesztés.

A telefonok, ill. táblagépek (továbbiakban csak *mobilkészülékek*) megoszlásását (platform, vagy akár készülékgyártó szerint) az újonnan eladott telefonok viszonylatában van lehetőség mérni. Arról ugyanis sajnos nincs információ, hogy a régebben vásárolt telefonok közül melyik van épp használatban, így figyelni kell arra, hogy ez egy kicsit eltér a pillanatnyi használtság megoszlásától. Ezzel szemben például a platformon *belüli* megoszlásról (pl. képernyőméret, OS verzió (2.3. ábra)) már viszonylag aktuális kép kapható az *alkalmazásboltokra*⁴ való csatlakozások figyelésével.

Fontos kérdés tehát, hogy mikorra tervezünk pontosan. A jelenlegi eladások jól tükrözik a közeljövőt, de ha kifejezetten az aktuális piaci igényre szeretnénk felkészülni, akkor az elmúlt évek adatait is figyelembe venni (2.2. ábra), ugyanis amikor például a Symbian-os készülékeket gyakorlatilag már alig értékesítették, az emberek nagy részénél még minden ilyen készülék volt.



2.2. ábra. Mobilplatformok százalékos világpiaci eloszlása 2007 óta az újonnan eladott telefonok viszonylatában. (Forrás: Gartner, 2007-2015)

A diplomaterv során olyan prototípust kívánok készíteni, ami széles körben alkalmazható lehete az aktuális mobileszközökön. Éppen ezért csak a 3 legelterjedtebb platformot elemzem részletesebben: az Androidot, iOS-t és Windows Phone-t, de érdemes lehetne egyéb platformokat is elemezni a jövőbeli tendencia változásokra felkészülve.

⁴ Az okostelefonok operációs rendszerei többé már nem komplett, fix rendszerek, hanem kisebb alkalmazásokkal (app-okkal) testreszabható keretrendszer. Ezt a felhasználó igénye szerinti folyamatos testreszabhatóságot a mobilinternet elterjedése segítette elő. minden letöltés, vagy frissítés az *alkalmazásbolton* keresztül zajlódik – és egyre inkább szükségessé is teszik ezt, ugyanis így tudják garantálni a letöltött szoftverek megbízhatóságát. Ez a gyakori csatlakozás lehetőséget nyújt arra, hogy a telefonok használatáról viszonylag pontos, naprakész statisztika készülhessen.

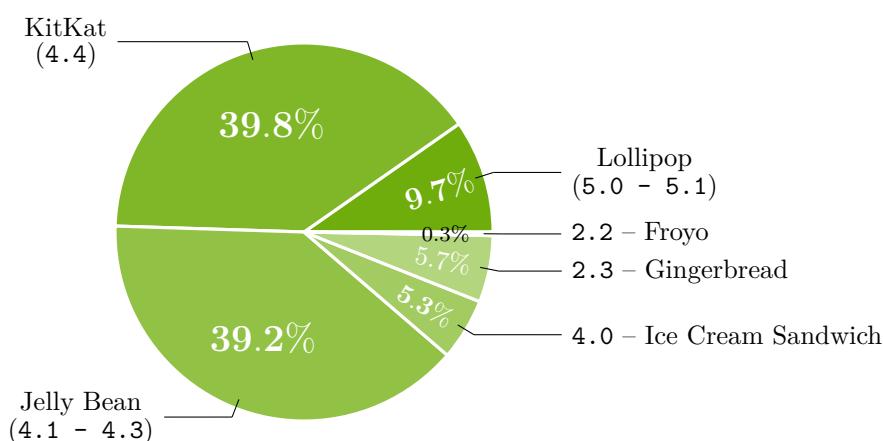
2.1.1. Android

Az Android napjaink legelterjedtebb mobil operációs rendszere (ld. 2.1. ábra).

Kezdetben a 2003-ben megalakult *Android Inc.* fejlesztette, majd 2005-ben vette meg a *Google*, aki a felvásárolt cégről nevezte el a platformot. Először titokban fejlesztette a rendszert, főként a webes technológiához (pl. Gmail, Google Maps) szeretett volna még egy platformon lefedést biztosítani. Később a 2007 novemberében megalakított OHA⁵ konzorcium folytatta az ekkori béta verzió fejlesztését, de a legjelentősebb szerepet továbbra is a Google képviseli. Sikeréhez nagy mértékben hozzájárult, hogy a konzorciumhoz csatlakozott mobilkészülék gyártók részt vehettek a fejlesztésben, ingyen hozzájutottak a forráskódhoz, az olcsóbb készülékeket pedig szívesebben vásárolták az emberek.

2007-ben jelent meg a publikus *SDK*⁶, majd 2008 szeptemberében jelent meg az 1.0-ás verzió a legelső Androidos készüléken, a HTC Dream (T-mobile G1)-en, melyre 2009 februárban adták ki az 1.1-es verziót.

A 2009-ben kiadott kezdeti 1.x (*Cupcake*, *Donut*) verziókat követően a 2.x (*Eclair*, *Froyo*, *Gingerbread*) verziók már jelentősen elterjedtek, a 2.3-as *Gingerbread* verzió főszereplésével, mellyel nagyon sok készüléket gyártottak (amíg nem forrt ki a méltó utód), így sokáig nagyon fontos volt ennek a verziónak a támogatása, máig megmaradt egy 6% arányban (2.3. ábra). A 2011 februárban megjelent 3.x-es (*Honeycomb*) verzió ugyanis csak táblagépekre készült, mely új perspektívát nyitott, de nagyon nehézkes volt a két felbontás együttes támogatása és ironikus módon a *fragment*-ek (képernyörészek) bevezetése okozta az egyik legnagyobb szerepet az Android-ra jellemző ún. *fragmentáltság*ban (több verzió, típus együttes kezelése). Végül 2011 októberben egyesült ismét a mobiltelefonok verziójával a megjelent 4.x (*Ice Cream Sandwich*, *Jelly Bean*, *KitKat*) verzióban, mely verzió minimumként szerencsére már 94%-ban feltételezhető. Legutóbb pedig 2014 novemberében jelent meg az 5.x (*Lollipop*) verzió, mely a felhasználói élményt értelmezi újra letisztultságával, gördülékenységével, optimalizáltságával és már 10%-os elterjedtségnek örvend.



2.3. ábra. Androidos készülékek verzió szerinti eloszlása (2015 május)

⁵ **OHA:** Open Handset Alliance. Egy kezdetben 34, ma már 84 tagból álló üzleti szövetség, melynek célja, hogy elősegítse a mobiltelefonok fejlődését nyílt szabványok létrehozásával. A konzorcium tagjai telefonkészülék gyártók, félvezető gyártók, hálózat operátorok, szoftveres cégek és egyéb vállalatok. [36]

⁶ **SDK:** Software Development Kit, azaz szoftver fejlesztő készlet

Érdekesség, hogy a főbb verziókat a Google mindenkorábban következő betűjével kezdődő süteményről nevezi el. Egy adott verziót elérhető feature-öket viszont az *API⁷ szint* (vagy natív kódnál NDK verzió) határozza meg, így a gyakran felváltva használt verziószámok, verziónevek, API szintek, NDK verziók közötti kavalkádot a 2.1. táblázat segít átláthatóbbá tenni.

A rendszer Linux kernelre épül, amihez a teljesítmény kritikusabb, főbb komponensek C++-ban készültek. A keretrendszer nagyobbik része Java nyelven íródott, ezt a keretrendszer felhasználói programokat írni. Az alkalmazásokat egy saját fejlesztésű *Dalvik VM⁸*, *DVM* virtuális gép futtatja a Java VM helyett, így az alkalmazások nem teljesen kompatibilisek a Java SE-vel és Java ME-vel sem (főleg a felhasználói felület terén), de így is előnyös, hogy felhasználhatók Java-s programrészek.

Közben pedig az egyre nagyobb tárhelyeknek köszönhetően – immár a sebességre optimalizálva a 4.4-es verziótól először opcionálisan, majd az 5.0-ás verziótól már teljeskörűen le lett cserélve a *JIT⁹* fordítású DVM, *AOT¹⁰* fordítású *ART¹¹*-ra.

Az 1.5-ös verziótól kezdve rendelkezésre áll az *NDK¹²*, melynek használatával a virtuális gépet megkerülve natív, azaz C, C++ kódok is futtathatóak *JNI¹³*-on keresztül történő kommunikáció lehetőségével. Az NDK részeként egyre több komponensnek készül el a natív implementációja is. Bizonyos típusú kódrészletek így akár többször gyorsabban is futhatnak. Ez AR alkalmazásoknál is fontos lehet, hiszen a válaszidő és képfrissítési frekvencia határozza meg az alkalmazás életszerűségét.

További előnye a platformnak, hogy nyílt, az *AOSP¹⁴* keretében bárki számára elérhető [10] a forráskód, letölthető és a fejlesztéshez is hozzá lehet járulni (ahogy én is tettem az E. függelékben felsorolt hibák észrevétele esetén). A platform nagyfokú szabadságot nyújt szinte minden komponenshez, a hivatalos weboldalon [11] pedig nagyon részletes dokumentáció érhető el, példaprogramokkal együtt.

Egyik legnagyobb hátránya a *fragmentáció* néven ismert problémája. Mivel nagyon sokféle eszközt támogat, nehéz általános, minden készüléken megfelelően futó alkalmazást írni. Az újabb Android verziókban ugyan már egységesen kezelhetőek bizonyos problémák (pl. az eltérő képernyőméretekhez skálázódó *Fragment API*), de a készülékek rövid támogatottsága miatt sok készülék futtat régi verziót. Ennek mérséklésére létrehoztak egy *Support Library-t*, amibe a 17, 13, 8, 7, vagy 4-es API verzióig visszamenően támogatják néhány fontos osztály alternatív implementációját. Erre nagy szükség volt például a 3.x-es verzióról említett *Fragment API* megjelenésekor, hiszen hiába volt a 4.0 verziótól már egységes a megjelenítés, ha régebbi verzióra ezt a kódot nem lehetett lefordítani. Szerencsére ma már könnyen feltételezhető akár 4.x verzió is, annak 94%-os elterjedtsége miatt (2.3. ábra).

⁷ **API:** Application Programming Interface, azaz alkalmazás fejlesztési interfész.

⁸ **VM:** Virtual Machine, virtuális gép, melynek segítségével architektúra független kód futtatható

⁹ **JIT:** Just-In-Time. Futásidejű fordítási mód a tárhely optimalizálására.

¹⁰ **AOT:** Ahead-Of-Time. Előre fordítás (telepítéskor) a tárhely rovására.

¹¹ **ART:** Android RunTime.

¹² **NDK:** Native Development Kit, natív fejlesztői környezet

¹³ **JNI:** Java Native Interface

¹⁴ **AOSP:** Android Open Source Project, Android nyílt forráskodú projekt

Megjelenés	Verzió	API szint	NDK verzió	Kódnev		
2008.09.23	1.0	API 1	-	-		
2009.02.09	1.1	API 2				
2009.04.30	1.5	API 3	NDK 1	Cupcake		
2009.09.15	1.6	API 4	NDK 2	Donut		
2009.10.26	2.0	API 5		Eclair		
2009.12.03	2.0.1	API 6				
2010.01.12	2.1	API 7	NDK 3			
2010.05.20	2.2	API 8	NDK 4	Froyo		
2011.01.18	2.2.1					
2011.01.22	2.2.2					
2011.11.21	2.2.3					
2010.12.06	2.3	API 9	NDK 5	Gingerbread		
2010.12	2.3.1					
2010.01	2.3.2					
2011.02.09	2.3.3					
2011.04.28	2.3.4					
2011.07.25	2.3.5					
2011.09.02	2.3.6					
2011.09.21	2.3.7	API 10	NDK 6	Honeycomb		
2011.02.22	3.0					
2011.05.10	3.1					
2011.07.15	3.2					
2011.08.30	3.2.1					
2011.09.20	3.2.2					
2011.10	3.2.3					
2011.12	3.2.4					
2012.01	3.2.5					
2012.02	3.2.6					
2011.10.19	4.0	API 14	NDK 7	Ice Cream Sandwich		
2011.10.21	4.0.1					
2011.11.28	4.0.2					
2011.12.16	4.0.3					
2012.03.29	4.0.4	API 15	NDK 8	Jelly Bean		
2012.07.09	4.1					
2012.07.23	4.1.1					
2012.10.09	4.1.2					
2012.11.13	4.2					
2012.11.27	4.2.1	API 16	NDK 9	KitKat		
2013.02.11	4.2.2					
2013.07.24	4.3					
2013.10.03	4.3.1	API 17				
2013.10.31	4.4					
2013.12.05	4.4.1					
2013.12.09	4.4.2					
2014.05.23	4.4.3					
2014.06.19	4.4.4	API 18	NDK 10	Lollipop		
2014.06.25	4.4W					
2014.09.06	4.4W.1					
2014.10.21	4.4W.2					
2014.11.12	5.0					
2014.12.02	5.0.1	API 19	NDK 11	Marshmallow		
2014.12.19	5.0.2					
2015.03.09	5.1					
2015.04.21	5.1.1	API 20	NDK 12	Nougat		

2.1. táblázat. Android verziók, API szintek, NDK verziók és kódnevek közötti összerendelés

2.1.2. iOS

Az iOS (korábban csak iPhone OS, OS X iPhone-ként emlegetett operációs rendszer) az Apple hordozható multimédiás készülékekre kifejlesztett operációs rendszere. 2007 júniusában jelent meg az 1.0-ás verziója az *iPhone* bemutatásával, majd az októberben megjelent *iPod Touch* is ezt futtatta. Eredetileg zárt rendszernek terveztek, végül 2008 márciusában mégis adtak ki SDK-t. A 2.0-ás verziótól jelent meg a saját alkalmazásboltja, a nagysikerű *App Store*. Az évente megjelenő verziók közül a 3.0-ás 2009 júniusában jelent meg a világsikert arató *iPad* együtt, azóta már a 8.3-as verziótól tartanak.

A cég szemlélete a minőség és megbízhatóság, amit gyakran megszorításokkal, korlátozásokkal ér el, viszont így a legtöbb alkalmazás először erre a platformra készül el, főleg, ha fizetős, vagy ha a robosztus működés más szempontokat megelőz.

Az iOS az asztali gépeken futó Mac OS X-hez hasonlóan Unix-alapokra épül, de egymással nem kompatibilisek. Az alkalmazásokat Objective-C-ben kell írni, ami egy C nyelvet kiegészítő objektum-orientált nyelv, de a C++-szal nem kompatibilis. A lefordított kód natív módon fut, ami csúcsminőségű hardverekkel párosul (ugyanis iOS csak az iPhone készülékeken futhat).

A cég filozófiájából kifolyólag a rendszer nagyon sok szempontból zárt, például csak a publikus API használható fejlesztésre. A termékek jóval drágábbak, mint vetélytársaié, így csak egy szűkebb réteget céloz meg, valószínűleg sosem uralja majd a piac nagyobbik felét, ellenben jó ideig biztos pozíciója van a prémium kategóriában. Fölénye leginkább a táblagépek szegmensén volt érezhető sokáig, hiszen az *iPad* volt az elsőként megjelent táblagép, mára viszont már utolérte az Android (ld. 2.1b. ábra).

Összességében a rendszer gyors, megbízható, robosztus és a fejlesztéshez is nagyon gazdag anyag található, érdemes fejleszteni rá. Csupán szimpatiából és egyéb önérvédekekből nem ezt a platformot választottam elsődlegesnek, ugyanis a fejlesztéshez nem csak iPhone, de egy Macintosh számítógép is kellene, mert csak arra érhető el az Objective-C nyelvet fordító *XCode* fordítókörnyezet. Természetesen a későbbiekben bemutatott módszerek átültethetők erre a rendszerre is.

2.1.3. Microsoft

Talán a Microsoft tette a legemlékezetesebb áttérést a számítógépek és okostelefonok közt a sokak által ismert PND-ken futó **Windows CE**¹⁵ operációs rendszerével.

Nagyon sokáig kitartott az erre épülő **Windows Mobile** operációs rendszerével, sajnos nem ismerte fel az Apple igazát: egy okostelefonnak nem a lekicsinyített számítógép a legalkalmasabb. Újra kellett volna reformálnia az egész koncepciót, kezdve a felhasználói felület újjal való vezérlésének megfelelően. Kevés fejlesztési lehetőséget nyújtott, nehéz volt alkalmazásokat letölteni rá, bár akkor még nem is volt olyan elterjedt a mobilnet.

A 2010 februárjában megjelent **Windows Phone 7** már az új ideológiáknak megfelelően, csodaszép, újragondolt felülettel (Metro UI) jelent meg. A megszokott technológiáinak segítségével .NET-ben fejleszthető, sok fejlesztő tudott volna gyorsan áttérni a fejlesztésre.

¹⁵Windows CE – Windows Embedded Compact

Sajnos a Microsoft a legjobb példa arra, hogy ma már nem csak a platform jellemzői fontosak, sokkal fontosabb a támogatottság, a rájuk írt alkalmazások határozzák meg sikérét. A 2012-ben később kiadott **Windows Phone 8** (és 2014-es 8.1) is hiába modern, szép, gyors, könnyen fejleszthető, már nehezen tud versenytársai közelébe kerülni. Leginkább a nagyobb táblagépek piacán tudhat érvényesülni **Windows 8** (és 8.1) operációs rendszerével, melyet sokan választhatnak laptop helyett. A 2015-re várható **Windows 10** viszont egységesíteni próbálja a telefon, táblagép, számítógép és Xbox platformokra való fejlesztést, ami még egy komoly lökést jelenthet.

Egy alkalmazást sok esetben csak az Androidos és iOS-es verzió elkészültét követően implementálnak erre a platformra is. Ez viszont már annyi fejlesztést igényelne, hogy ilyen esetben inkább valami cross-platform megoldást készítenek.

2.1.4. További platformok

Érdemes még megemlíteni a *Nokia* által fejlesztett **Symbian** operációs rendszert hosszantartó sikerének köszönhetően (2.2 ábra), vagy az amerikában népszerűbb **BlackBerry** operációs rendszert. Gyakran felbukkan még a **Bada**, illetve a **Tizen** operációs rendszer is. Nagy érdeklődéssel várják a **Firefox OS** és az **Ubuntu Edge** operációs rendszerek megjelenését is, ám ezeket már nem vizsgáltam.

2.1.5. A választott platform

Diplomatervem során fontos szempont volt egy jól dokumentált, elterjedt platform ki-választása, hiszen nagyrészt általánosan alkalmazható módszereket vizsgáltam, így most nem egy új platform kipróbálása, megismerése, vagy népszerűsítése volt a cél, hanem az kapott eredmények hatékony lefejlesztése.

A legelterjedtebb Android platformot választottam, mely döntéshez nagymértékben hozzájárultak személyes preferenciám is: telefonkészülékem operációs rendszere, valamint ezen a platformon rendelkezlek tapasztalattal.

De ezen kívül minden szempontból megfelelőnek bizonyult, kellően támogatott, sok – a következő fejezetben bemutatott – toolkit érhető el rá és megvan a lehetőség NDK-val, vagy párhuzamos programozással való optimalizálásokra is.

Mivel munkám során újszerű alkalmazásokat vizsgáltam, ezért magas szintű felhasználások helyett az API-k legalapvetőbb felhasználási módjait vizsgáltam meg. Ennek során fontos volt, hogy a platform nem csak sok, kiváló minőségű dokumentációval rendelkezik, de a nyílt forráskód még könnyebbé tette az utánjárást, értelmezést. Az E. függelékben több példa mutatja, hogy 1-1 ilyen utánjárás során a forráskódból értettem meg a hiba okát, melyre így még javítást is tudtam javasolni. Ez az open-source projektek hatalmas előnye.

2.2. Fejlesztőeszközök

Fejlesztések során célszerű megfontolni, milyen alapokkal is dolgozzunk. Van, hogy elegendőek csupán a platform nyújtotta lehetőségek is, de összetettebb alkalmazásoknál sok fejlesztési idő mehet rá olyan programkódokra, amit már más programok során is átgondoltak, lefejlesztettek és még kiforrottabb is. A nagyon gyakori, hasonló felhasználási módokra kész keretrendszereket, könyvtárakat (framework, toolkit, library) is találhatunk.

A kiterjesztett valóság egy összetett alkalmazási mód, ahol a kamera, szenzorok, képfelismerés, képmegjelenítés mindegyike szerepet játszik. Léteznek külön szenzor-, képfeldolgozási- és grafikus toolkitek is, illetve speciálisan AR alkalmazásokhoz is fejlesztenek magasabb szintű, kombinált keretrendszereket.

Ezek közül ismertetnék néhányat és mérlegelem melyike(ke)t használjam fejlesztésem során.

2.2.1. AR toolkitek

Ma már több sikeres AR alkalmazás kínál fejlesztői keretrendszert is egyben, melyben megoldásukat általánosítva nyújtanak testre szabható megoldást. Ilyen a POI és képfelismerés jellegű AR alkalmazások készítésére megoldást nyújtó *Wikitude* [9] és *Layar* [14] is. Ezek azonban csak limitált képességű, speciálisan a célfeladatokra alkalmasak, így ezeket nem vizsgáltam. A két legnépszerűbb kiforrott AR toolkit a *Vuforia* és az *ARToolkit*, melyeket röviden ismertetnék.

ARToolkit

Weboldal: [13], [5], *Letöltés*: [4], *Dokumentáció*: [6]

Az **ARToolkit** (Augmented Reality Tool Kit) az egyik legrégebbi múltra visszatekintő és legismertebb AR toolkit, melyet több, mint 600.000-szer töltöttek le. Fejlesztését *Hirokazu Kato* kezdte, először 1999-ben mutatta be a *Siggraph* kiállításon, majd a *HitLab* (Human Interface Technology Laboratory) adta ki 2003-ban, ma az *ARToolWorks* fejleszti. Kettős licenszelésű, nem kereskedelmi célra ingyenesen használható (GNU GPL licensz alatt), kereskedelmi célra viszont engedélyköteles.

Egy szabadon felhasználható, publikus C++ nyelven írt gépi látás alapú multiplatform könyvtár, valós idejű AR alkalmazásokhoz. Leginkább 3×3 -as és 4×4 -es barcode-ok felismerésre alkalmas, melyeket a [7] oldalon generálhatunk. 3D-s megjelenítésre OpenGL-t használ. Van benne kamera kalibrálás is a különböző készülékek használatához. Sajnos a legutolsó kiadott open-source verziója a 2007 májusi 2.72.1-es verzió.

Azota zártan fejlesztik tovább, **ARToolkit Professional** néven, mely már Androidon is elérhető. 2013 szeptemberében adták ki az 5.0-ás verziót.

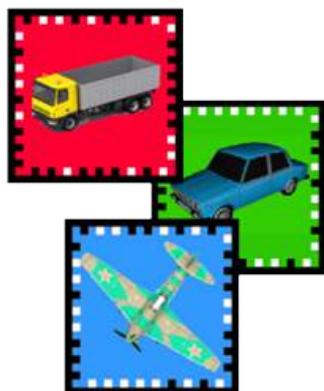
Az *ARToolKit* könyvtárat Androidra is portolták [3], **AndAR** (Android Augmented Reality) néven. Segítségével tisztán Java oldalról is használható a toolkit. A forrás nyílt, de már ez sem frissült 2010 október óta. Közben viszont az *ARToolWorks* is kiadta az *ARToolKit Professional* mobilokra szánt (*for Android* és *for iOS*) változatát.

Vuforia

Weboldal: [19]

A Qualcomm Augmented Reality megoldása (**QCAR**, ill. 1.5-ös verzió óta **Vuforia**) az egyik legkorszerűbb, legrobusztusabb keretrendszer. Aktívan fejlesztik, legutóbbi 4.2.3-as verziója 2015 májusában jelent meg. Több, mint 100.000 regisztrált felhasználója van és több, mint 10.000 alkalmazás készült már vele. Natív C++ kódban íródott, versenytársainál jóval gyorsabb és számtalan kisegítő lehetőséget építettek bele, mint a keret célpont (2.4a), kép alapú célpont (2.4c), többszörös célpont (2.4b), henger-, kúp- alakú célpont, virtuális gombok (2.4d), átlátszóság/takarás kezelés (occlusion management) (1.2g), szövegfelismerés, felhő alapú felismerés, hozzáférés a háttérképhez. Több példakód és tutorial található az oldalon. Szinte az összes – legalább 2.3-as Android verziójú, ARMv7 processzorral rendelkező készüléket támogatja. Tartozik hozzá plugin a népszerű cross-platform *Unity* játék motorhoz is.

Gyorsasága, korszerűsége, kisegítő lehetőségei és nagyfokú támogatottsága miatt mindenkorban a jelenlegi legjobb toolkitnek tűnik.



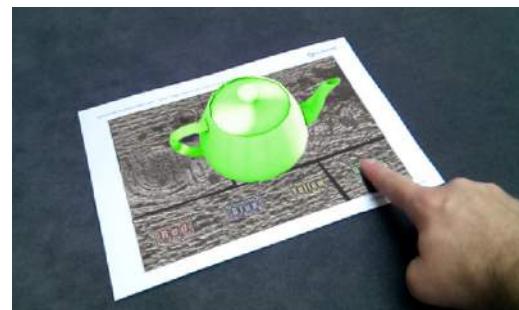
(a) Keret célpontok



(b) Többszörös célpont



(c) Kép célpont



(d) Virtuális gombok

2.4. ábra. Vuforia feature-ök

2.2.2. CV toolkitek

A kiterjesztett valóság egy alacsonyabb szintű építőköve a *gépi látás* (Computer Vision, röviden CV), így ha a szokványos markerektől eltérően saját magunknak kell implementálni valamilyen képfelismerést, képfeldolgozást, akkor érdemes valamilyen CV könyvtárat felhasználni.

OpenCV

Az OpenCV (**O**pen Source **C**omputer **V**sion Library) a legismertebb, nyílt forráskódú, CV könyvtár [16]. Optimalizált C/C++ nyelven íródott, multiplatform és teljesen ingyen használható (BSD licensz alatt áll). Fejlesztését 1999-ben az Intel kezdte, mára több, mint 2500, nagyrészt képfeldolgozással kapcsolatos függvény található benne. Több, mint 7 milliószor töltötték le és több, mint 47.000 fős közösséget számol. Fejlesztése ma is aktívan zajlik, nagyjából 3 havonta adnak ki új verziót, legutóbb 2015 márciusban adták ki a 2.4.11-es verziót, de közben már a 3-as verzió is készül párhuzamosan, melyből áprilisban érkezett a 3.0-RC1.

Android platformra is portolták [17], így Java oldalról is sok funkciója használható, de NDK-ból persze közvetlen is hívhatóak a C++-os osztályok. Tartozik hozzá 4 példaalkalmazás és 3 tutorial. Hasznos, hogy vannak benne képfeldolgozó, képtranszformációs függvények, illetve kamera kalibráció, panoráma kép illesztő is.

A 2.4.2-es verziótól egy dinamikus könyvtár koncepciója használandó Androidon. Ennek lényege, hogy a közel 5MB-os könyvtár statikus fordításkor jelentős mértékben megnövelné az általában kb fél MB-os program méretét, így inkább külön alkalmazásba szedték az OpenCV Managert [30], amit csak egyszer kell telepítenie a felhasználónak.

FastCV

A *FastCV* a *Qualcomm* által fejlesztett, mobilra optimalizált ingyenesen használható CV könyvtár [18]. Az 1.0-ás verziót elsősorban Androidra terveztek, a jelenlegi 1.4-es verzió már a Windows Phone 8-at is támogatja és iOS-re is tervezik a támogatást. ARM processzorokra lett optimalizálva, de a *Qualcomm* saját *Snapdragon* processzoraihoz hardveres gyorsítást is használ.

Az implementált algoritmusok főbb típusai:

- Matematikai és vektor műveletek
- Képfeldolgozás
- Kép transzformációk
- Objektum felismerés
- 3D rekonstrukció
- Mozgás és objektumkövetés

Zxing

Barcode-ok felismerésére alkalmas a nyílt forráskódú, ingyenesen használható **Zxing** (Zebra Crossing) könyvtár [20], melyről már az 1.2.2. fejezetben esett szó.

2.2.3. Grafikus megjelenítők

Ha összetettebb grafikai megjelenítésre van szükségünk, érdemes hardveres gyorsítást használni, mert az erre specializált GPU-n párhuzamosítva, hatékonyabban számolhatóak ki a képrenderelési feladatok, így nagyobb lehet a képfürissítési ráta, vagy több CPU idő marad egyéb számításokra. AR feladatoknál mindkettő fontos szempont.

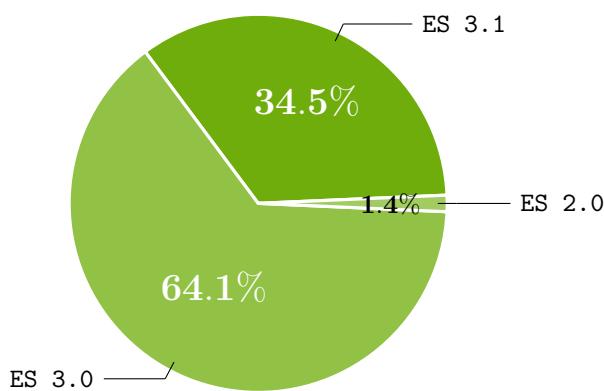
Androidon a *Khronos Group* konzorcium által készített OpenGL ES¹⁶ API-ján keresztül használhatunk grafikus rendereléseket. Ez az asztali OpenGL¹⁷ API egyszerűsített változata beágyazott rendszerekhez, mint a konzolok, okostelefonok, táblagépek. Több nyelven, több platformon is elérhető. Androidon Java és NDK oldalról is tudjuk használni.

A 2003-ban specifikált OpenGL **ES 1.0** az OpenGL 1.3 API alapján készült, annak jelentős egyszerűsítésével. Androidon a kezdeti 1.0-ás verziótól elérhető.

Az OpenGL **ES 1.1** már az OpenGL 1.5 alapján készült, Androidon 1.6 (API 4) verziótól használható.

A 2007-ben specifikált OpenGL **ES 2.0** az asztali OpenGL 2.0-ra épült és nem kompatibilis a korábbi OpenGL ES 1.x-szel, a fixfunkciós fotószalag helyét átvette a programozható futószalag. Androidon a 2.2-es (API 8) verziótól érhető el.

A 2.0-ás verzió legnagyobb hátránya, hogy a szakszerűtlenül definiált pufferformátum specifikációja alapján az egyes gyártók esetenként eltérő pufferformátumot implementáltak a hardverbe, amire a programozóknak külön oda kell figyelnie. A 2012-ben bemutatott OpenGL **ES 3.0**-ás verziójában ezt már javították, ráadásul felülről kompatibilis a 2.0-ás verzióval. Androidon a 4.3-as (API 18) verziótól érhető el. A legújabb, 2014 márciusában kiadott OpenGL **ES 3.1** az Android 5.0-ás verziójától támogatott.



2.5. ábra. *Androidos készülékek OpenGL ES verzió szerinti eloszlása (2015 május)*

¹⁶OpenGL **ES**: OpenGL for Embedded Systems

¹⁷**OpenGL**: Open Graphics Library

Az Android verziók önmagában csak az interfészt adják az OpenGL ES felé, a készülék gyártóján múlik, hogy végül melyik verziót implementálja, de szükségszerűen a korábbi verziókat is. A 2.5 ábrán látszik, hogy a 2.0-ás verziót szinte minden készülék támogatja, így emiatt nem kell kisebb verziók mellett dönteni, csak ha a könnyebben érhető programozási nyelvet szeretnénk használni, mert különben általánosságban is gyorsabb a 2.0-ás és még inkább a 3.0-ás verzió és jóval több funkcióval rendelkeznek.

Általában véve elég nehéz az OpenGL nyelvezete, ezért gyakran célszerűbb egy grafikus keretrendszer használni, ilyen például a *min3d*, vagy a *Rajawali*.

min3d

Az egyik legegyszerűbben használható OpenGL ES 1.0-t használó open source keretrendszer a 2010-ben létrehozott **min3d** [2]. Android 1.5 verzióval kompatibilis. 24 egyszerűen megérthető példa tartozik hozzá. Képes .jpg, .obj és .3ds file-okat betölteni. 2011 óta már nem fejlesztik.

Rajawali

Szintén egy egyszerűen használható keretrendszer az OpenGL ES 2.0/3.0 API-kat használó szintén open source **Rajawali** is [15]. Android 4.0+ (API 14)-re támogatott. Tartozik hozzá 31 tutorial és készült hozzá Vuforia integráció is.

2.3. A választott módszerek

Mobilplatformok közül tehát elterjedtségének, személyes preferenciáknak és nyíltságának köszönhetően az **Android**ot választottam. Mivel a módszer eredményes kipróbálása és továbbfejlesztése fontosabb szempontnak tűnt, mint a sok féle Android verzió és készülékkel való kompatibilitásra törekvés, ezért – a mára már kellően, 94%-os arányban feltételezhető – **4.0 verziót** tűztem ki minimum követelményként. Erre a kamerakép OpenGL-ben történő felhasználása miatt is volt szükség, mely gyorsabb, mint más módszerek. Ezen kívül nem törekszem különösebben a kis képernyőjű telefonokra és nagyobb táblagépekre, melyekhez külön UI-t kellene tervezni. **Közepes méretű** (4-5 colos) **okostelefonokhoz** tervezem a UI-t, elsősorban **Nexus 5-re** (ennek ellenére várhatóan a többin is megfelelően használható lesz).

Az **AR toolkitek** tekintetében viszont nem lesz szükségem nagyon magas szintű képességekre, a szenzorok és kamera kezeléséhez csak az **Android API**-ját használom fel.

Képfeldolgozási feladatokhoz **OpenCV** használatát vizsgálom meg. Van benne ugyanis perspektív torzítás (warpPerspective) és panorámaképek készítéséhez hasznos függvények, ami a *FastCV*-ben viszont nincs. További előnye az *OpenCV*-nek, hogy más platformokon is használható (bár a *FastCV*-hez is tervezik az iOS támogatást).

Grafikai renderelésre pedig az **OpenGL ES**-t, mégpedig a 2.0-ás változatát választottam. Ez ugyanis szinte az összes készüléken elérhető és szükség esetén bővíthető 3.0-ás funkciókkal. Szerencsére egyelőre csak pár egyszerűbb objektumot tervezek kirajzolni, textúrázott síkokat és vonalakat. Ha általános tartalmakkal egészíteném ki a látott képet, használnék valamilyen toolkit-et, például a *Rajawali*-t, aminek elég egyszerű a használata és OpenGL ES 2.0/3.0-t használ.

Alapvetően az Android **Java**-s interfészét használom, esetleg teljesítmény kritikus részről lehetne **NDK**-val megvalósítani. A fejlesztés során **Windows**-os környezetben, **Eclipse**ben programozok a hozzá kiadott *ADT*¹⁸-vel. Közben az *Android Studio* ugyan meghaladta az Eclipse-hez kiadott ADT pluginban rejlik lehetőségeket (egyedül a natív kód debuggolásában van csak lemaradása), valamint már ez lett a hivatalosan támogatott fejlesztőkörnyezet és már csak ezt fejlesztik. Viszont sok kis példa-projekten dolgozom, a közös részeket külön library-be kiemelve, ezért pillanatnyilag jobban alkalmazható az Eclipse-es *Package Explorer*, ahol egyszerre nyitva lehet tetszőlegesen sok projekt és egyből értesülök arról, hogy egy library-ben történt változtatás mely projektek buildjét „törte el”, amit így gyorsan módosítani tudok.

A tervezés során **UML** diagrammokat használok, diplomatervemet **LATEX** dokumentumszerkesztővel készítem. A részletesebb tesztkörnyezet, használt programok listája a **CD melléklet** c. függelékben olvasható.

¹⁸ **ADT:** Android Development Tools

3. fejezet

Égitest felismerés alapú kalibráció

Az emberiség űsidők óta foglalkozik a Nap, csillagok és más égitestek mozgásának vizsgálatával, előrejelzésekkel és ezekhez kapcsolódóan a tájékozódással, időméréssel.

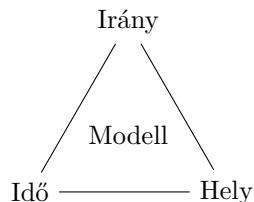
Az iránytű első formája csak a XIII. században került Európába (mely ekkor még egy vízzel telt edényben úszó szalmatutaj volt, melynek szalmaszálai közé szöttek a mágneses fémszálakat). Egészen addig a csillagok állása alapján tudtak tájékozódni.

A Nap (és árnyékának) mozgását megfigyelve abban periodikusságot találtak napi és éves szinten. Ezen mérések képezték a Nap helyzetének előrejelzésének alapjait, illetve készítettek napórát, mely a vetületi árnyék alapján mutatta az időt. A tengerészek pedig egy mechanikus szerkezettel, a *szextánssal* mérték a Nap és csillagok állását és ebből számolták vissza a földrajzi pozíciójukat, meglehetősen pontosan.

Ebben a fejezetben megvizsgálom, hol tart a mai technika, mennyire pontosak a telefonokba épített szenzorok, milyen pontossággal használhatóak. Érdekes kérdés, hogy vajon mennyire lennénk képesek kiváltani a digitális iránytűt, kvarcórát vagy a GPS-es helymeghatározást, ha valamelyik nem lenne. De nem csupán érdekesség, valóban szükség lehet némelyik kalibrációjára bizonyos esetekben.

3.1. Naprendszer modell

A naprendszer mozgásának megismerésének, előrejelzésének kulcsa, hogy meg tudjuk határozni bármely időben a teljes állapotát. Ezek után ha ismerjük a Földön lévő pozíiónkat, meg tudjuk mondani, hogy látjuk a környezetet – merre van a Nap, a Hold (ill. a csillagok). Ha tehát az idő és a pozícióink meghatározza a környezetet és más-más paraméterekhez más-más környezetet rendel (azaz *injektív* a leképezés), akkor valamelyik paraméter ki is található a többi ismeretében (ld. 3.1. ábra), azaz létezik *inverz* hozzárendelés.



3.1. ábra. Bármely 2 egyértelműen meghatározza a harmadikat.

Több alkalmazás mutatja a Nap, vagy más égitestek, műholdak pillanatnyi helyzetét, mint az 1.12. ábrán bemutatott *Sky Map*. Jelenlegi GPS koordinátánk és pillanatnyi idő alapján megnézhetjük, hogy az égen milyen irányban található egy-egy égitest.

De a dolog megfordításaként, ha ismert az általános modell és ismerjük az egyik paramétert(hely vagy idő), valamint az eredményt (pl. hogy merre van a Nap), kitalálható a másik, ismeretlen paraméter értéke. Ezt a 3 felhasználási módot a 3.2 táblázat foglalja össze, melynél a + jelöli az ismert, ? pedig az ismeretlen paramétert.

Hely	Idő	Irány	Példák
+	+	?	Égitest előrejelzés - maga a modell, pl. <i>Sky Map</i>
+	?	+	Időmeghatározás, pl. gnómon, napóra
?	+	+	Pozícionálás: földrajzi hely, pl. tengerészek navigációja

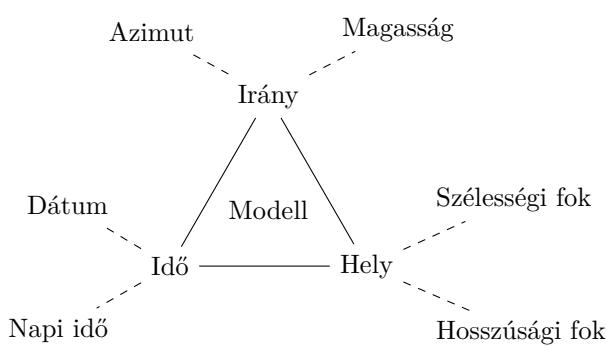
3.2. ábra. A 3 alapprobléma és alkalmazási példái

Ez persze így még nem teljesen állja meg a helyét, ugyanis az idő túl változatos paraméter ahhoz, hogy a leképezés injektív legyen. Általában van egy Föld forgásából adódó kb. napi periódus és a Nap körüli keringésből adódó éves periódus, de ez attól függ a Holdat, Napot vagy más égitestet veszünk figyelembe. Ezért ha az időt akarnánk megtudni a többi paraméterből, több megoldás is adódna. Mindenképpen célszerű az időre inkább a naptári dátum és napi idő együtteseként tekinteni. Igazából még a naptári dátumot is tovább kellene bontani évre és dátumra.

A elemzés céljából az irányt is érdemes kettébontani az északi irányhoz képesti elfordulás (melyet *azimutnak* (angolul *azimuth*) nevezünk) és az emelkedési szögre, amit *magasságnak* (angolul *elevation*) hívunk.

A helyet, mint földrajzi pozíciót pedig *szélességi* és *hosszúsági fokra* bonthatjuk (*latitude* és *longitude*). Így a 3.3. ábrán látható felosztást kapjuk.

Ebben a felosztásban elemezve már pontosabb képet kapunk és újabb alkalmazási területeket találhatunk (ld. 3.4. ábra), nézzük sorra ezeket a lehetőségeket.



3.3. ábra. Bármielyik rész-paraméter is kitalálható a többiből.

Hely		Idő		Irány		Példák
Szélességi fok	Hosszúsági fok	Dátum	Napi idő	Magasság	Azimut	
+	+	+	+	?	?	Égitest előrejelzés - maga a modell, pl. <i>Sky Map</i>
+	+	+	?	+	+	Napi idő meghatározás, pl. gnómon, napóra
+	+	+	+	+	?	Északi irány meghatározása
+	+	?	+	+	+	Dátum meghatározása, pl. analemma, csillagképek
?	?	+	+	+	+	Helymeghatározás: földrajzi hely, pl. tengerészek navigációja

3.4. ábra. Részletesebb elemzés részparaméterek bontásában

3.1.1. Égitest előrejelzés

Ez gyakorlatilag maga a modell, ezzel foglalkozik az asztrológia, avagy csillagászat. Számos egyenletet találhatunk, melyek a Hold, Föld, és más Naprendszerbeli égitest pályáját írják le az idő függvényében. Ha pedig ismerjük a földrajzi helyünket, akkor átszámíthatjuk a Földi megfigyelésre (azimut, magasság).

3.1.2. Napi idő meghatározás

A Nap helyzetéből meghatározhattuk nagyjából az időt. Teljesen evidens, hogyha egyáltalán látjuk a Napot, akkor nappal van, ha nem éjszaka, szóval már ebből el tudjuk dönteni hogy reggel 6 és este 6 óra között van, vagy ezen kívül. Ha pedig pontosabban megfigyeljük - épp most kel fel a Nap, épp a legmagasabban jár, épp nyugat fele halad, stb, akkor a megfigyelés pontosságától függően akár pár óra pontossággal is megmondhatjuk csupán szemrevételezéssel, mennyi az idő. A vetületi árnyék pontosabb leolvasásával használták lényegében a napórát is. Viszont ez csak az úgynevezett *helyi polgári idő (local terrestrial time)*, pár kilométerrel arrébb már más időt mérnénk. A folyamatos időállítás miatt a Földön zónaidőt használunk. Ahhoz hogy pontosabban meg tudjuk határozni az időt, szükségünk van arra, hogy tudjuk, hol is vagyunk és mi a dátum. Ugyanis az egyes szélességi fokon más-más a Nap emelkedési szöge, ráadásul a Nap pályája változik attól függően hogy az év melyik szakában vagyunk. De a pontos dátum és hely ismeretében a Nap pozíciójából kiszámolható a pontos idő.

3.1.3. Északi irány meghatározása

Az északi irányt egész jól meghatározhattuk a Nap állásából a következő egyszerűsített gondolatmenettel: A Nap nagyjából keleten kel 6 óra fele, nyugaton nyugszik délután 6 fele és delelőskor dél fele található, a közben lévő időpontokban pedig lineárisan interpoláljuk az azimut-ját. A Nap azimutja így 180° -ot tesz meg, eközben 12 óra telik el, ami egy karórán

lévő kismutató esetében 360° körbefordulást jelent, ami pont kétszerese a nap sebességének. Ez a gondolat vezérli a következő, egyszerű iránymeghatározást karóra segítségével:

Fordulunk az óra 12-es számlapjával a Nap fele, majd a kismutató és a 12-es számlap közötti felezővonal megadja az északi irányt.

Ez a modell azonban elégé egyszerűsített, számos ponton pontosítható. Egyszerűt a Nap csak az északi féltekén delel dél irányban, a déli féltekén pont észak fele találjuk. Ráadásul ez sem pontos megfogalmazás, mert a Föld tengelyferdesége miatt a Nap az év során $\pm 23,5^\circ$ -ot vándorol a Rák- és Baktérítő között, szóval igazából az a kérdés, hogy a Nap évszaknak megfelelő Földre eső vetületéhez képest vagyunk-e északra vagy délre. Ezen kívül figyelembe kell venni, hogy nyári időszámítás esetén a helyi idő 1 órával eltér, tehát ilyenkor 1-kor delel a nap.

De pontos számításokkal meghatározható az északi irány.

3.1.4. Dátum meghatározása

Ha a Nap azonos időben vett pozícióit megfigyeljük az év folyamán, észrevehetjük, hogy nap, mint nap eltér egy picit. Ha a fényképeket egymásra exponáljuk (ld. 3.5), érdekes 8-as alakzatot figyelhetünk meg, az ún. *analemmát*. Ennek a hosszanti tengelye a Föld tengelyferdesége miatt alakul ki, hiszen az év során más-más szögben érik a napsugarak a Földet, így más más magasságban is látható. A kereszt irányú tengelye pedig a Föld ellipszis alakú pályán történő keringése miatt alakul ki, hiszen nem állandó a szögsebesség, napközelben gyorsabban halad a Föld, mint naptávolban.



3.5. ábra. *Analemma. A Nap látszólagos vándorlása az év során.*

Ez a jelenség mutatja, hogy ha ismert a pozícióink, idő és a Nap helyzete, akkor a dátum is kitalálható (leszámítva a tavaszi és őszi napéjegyenlőség megkülönböztetését, mert minden esetben a 8-as közepénél van a Nap).

Egy hasonló megfigyelhető jelenség a csillagképek helyzetének vizsgálata. A Föld napon-ta körbefordul „egyszer” a tengelye körül, ezt úgy tapasztaljuk, hogy az égbolt „egyszer” körbefordul a Sarkcsillag körül. De mivel az 1 napot nem a körbeforduláshoz mérjük, hanem 2 delelés között, ezért valójában naponta kicsit többet forog a Föld, mert 1 év alatt a delelek irányát nézve is körbefordul egyszer. Így a $365,25$ napos év alatt valójában $366,25$ -ször fordul meg a Föld a tengelye körül, ezért 1 nap alatt sem egyszer, hanem $1 + 1/365,25$ -ször fordul körbe. Ez azt jelenti, hogy ugyanabban az időben megfigyelve a

csillagképeket, minden nap kb 1 fokot elfodulnak a Sarkcsillag körül. Ráadásul a Föld tengelyferdesége miatt az év során a Sarkcsillagot ugyan minden közel északi irányban találjuk, de más-más magasságban, ezért az égboltból is kicsit más részletet láthatunk, ahogy a Sarkcsillaghoz képest a Föld horizontja más-más irányba dülönél.

3.1.5. Helymeghatározás

Ha adott időben a Föld egyes pontjain máshol látjuk az égitesteket, akkor ezek megmérésével kitalálható a pozícióink, mely módszerekre nagy szükség volt a tengeri navigációk során, melyre akár már az iránytű megjelenése előtt, csupán a magasság mérésével is lehetőség volt, nappal jellemzően a Nap segítségével, éjszaka pedig a csillagokéval.

Előzetes számítás

Tekintve, hogy a Föld egyenlítői kerülete kb. $40.000\ km$, a teljes kör pedig 360° , megbecsülhető, hogy egy-egy irány leolvasásával milyen pontossággal határozható meg a helyzetünk:

- 10° : $1000\ km \rightarrow$ kontinens és annak főbb részeinek elkülöntése, pl.
- 5° : $500\ km \rightarrow$ nagyobb ország
- 1° : $100\ km \rightarrow$ kisebb ország
- 0.1° : $10\ km \rightarrow$ megye

3.2. Eltérek

A modellek által számolt értékek számos módon eltérhetnek a valós mérésektől, például az iránytű nem pontosan észak fele mutat, vagy a Föld légköre megtöri a fény útját, ezért máshol látjuk az égitesteket. Ilyen eltéréseket mutatnak be a következő alfejezetek.

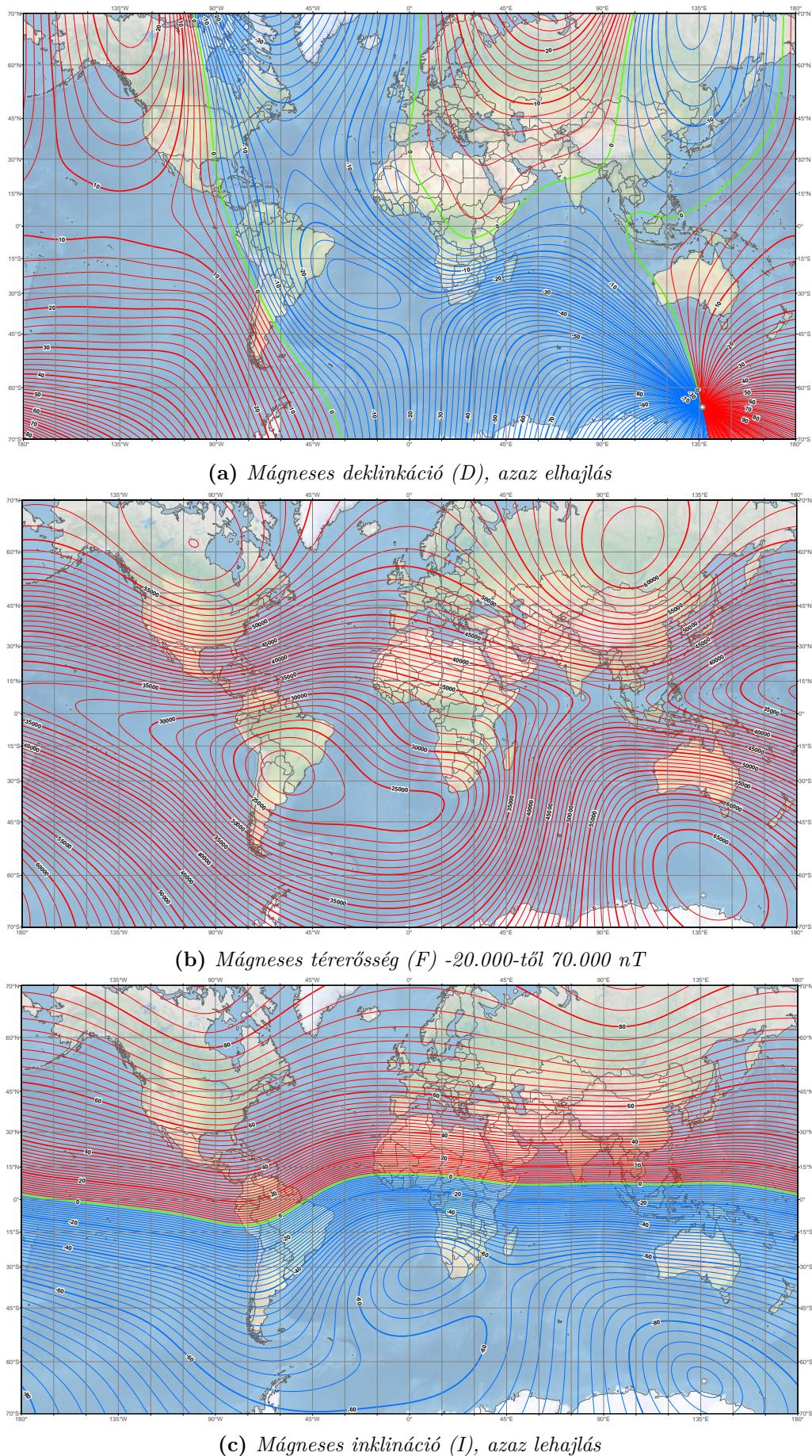
3.2.1. Mágneses eltérés

A mágneses eltérés, vagy *deklináció* (magnetic declination) abból adódik, hogy a Föld mágneses pólusa nem egyezik meg a földrajzi (forgástengely) pólusával, így a Föld pontjain más-más irányba korrigálni kell a mért irányt, ld. 3.6a. ábra. Hazánkban ez az érték $D = 4^\circ 23'$.

Ráadásul az évek során a mágneses pólus is vándorol a Földben lévő mágneses magma örvénylése miatt, ezért különböző pontosságú modellek is használnak. Az egyik ilyen a WMM (World Magnetic Model), mellyel 5 évre előre modellezik a mágneses értékek változását és 5 évente frissítik a modellt. Jelenleg a 2015-ös modell a legfrissebb, ld. ?? ábra.

Egy másik eltérés, hogy a mágneses erővonalak nem párhuzamosan követi a Föld felületét, így egy mágneses lehajlás, inklináció (magnetic inclination) is megfigyelhető. Ez hazánkban például $I = 64^\circ 7'$.

Egy harmadik eltérés, hogy a mágneses térerősség sem állandó a Föld pontjain, Magyarországon ez $F = 48,6\mu T$.



3.6. ábra. WMM 2015.0 mágneses modell Mercator vetületén

Az Android API-ban szerencsére beépítve találunk egy `android.hardware.GeomagneticField` nevű osztályt, melyet arra használhatunk, hogy lekérdezzük egy adott idő és helyhez tartozó értékeket. Mivel 5 évente frissül a WMM modell, ezért az Android 2.2-es verziója a kezdeti 2005-ös helyett már a 2010-es modellt használja. Ezek azonban csak 5 évre számolva pontosak, így az általános felhasználás végett, letöltöttem minden két változatot, valamint frissítettem az értékeket az új 2015-ös modell alapján is, melyeknek készítettem egy közös interfészt és így jelenleg 2005-2020-ig ad pontos eredményt.

3.2.2. Gravitációs eltérések

A mobilkészülékekkel az emelkedési szög mérése a gyorsulásmérő segítségével történik, így a gravitációs mező pontatlanságai kihatással lehetnek a mért értékekre.

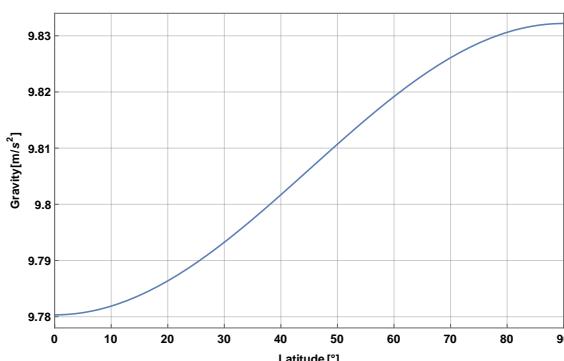
Egyrészt a Föld forgásellipszoid alakja miatt az érintősík szerinti merőleges nem egyezik meg a Föld közepe fele mutató tengellyel, ráadásul a Föld szerkezete sem homogén. Ezek szerencsére nagyon kis eltérések és bőven hibahatáron belüli eltéréseket jelentenek.

A Föld forgó mozgásából adódóan az egyes szélességi körökön más-más nagyságú a forgástengellyel merőleges irányú, kifele mutató centripetalis gyorsulás értéke. Ez hatással van a mért gravitációs gyorsulás irányára és nagyságára.

A nemzetközi gravitációs formula (International Gravity Formula, IGF) az egyes szélességi körökhöz adja meg az átlagos gyorsulás nagyságát (ld. fig:celebr:gravity:igf. ábra):

$$g_0 = 9,7803267714 \cdot \frac{(1 + 0,00193185138639 \sin^2 x)}{\sqrt{1 - 0,00669437999013 \cdot \sin^2 x}} \frac{m}{s^2} \quad (3.1)$$

Ez az érték $9,78033 m/s^2$ és $9,83219 m/s^2$ között mozog. Ez az ingadozás egy 0,5%-os ingadozást jelent, aminél nagyobb a gyorsulásmérő hibája, pedig különben akár a szélességi fok meghatározására is alkalmas lehetne.



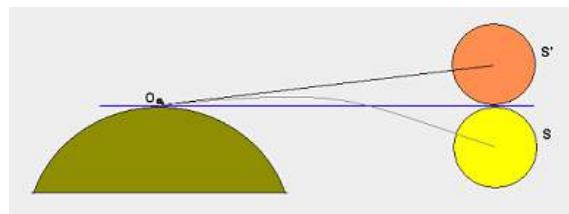
3.7. ábra. WGS84 ellipszoid gravitációs formula. A gravitációs gyorsulás értéke a szélességi fok függvényében.

A Föld szerkezetéből adódó helyi anomália még ennél is kisebb, max $0,0006 m/s^2$.

Az árapály hatások is okoznak gravitációs eltéréseket, ezek viszont még kisebb, maximum $0.000003 m/s^2$ nagyságúak.

3.2.3. Lékgöri refrakció

Mivel a világűrből érkező fények a föld lékgörébe lépve egyre sűrűbb közegbe lépnek, a folyamatos fénytörés a Föld felé hajlítja a fény útját, így az égitesteket a Föld felszínéről figyelve magasabb emelkedési szög alatt látjuk, mely jelenség neve lékgöri fénytörés, refrakció (atmospheric refraction), lásd 3.8. ábra.



3.8. ábra. *Lékgöri refrakció szemléltetése*

Magas emelkedési szögeknél ez az eltérés még elhanyagolható esetünkben, viszont 10° alatt az eltérés meghaladja a 0.1° -ot, a horizontnál pedig a 0.5° -ot, ami a teljes Nap, vagy Hold szögátmérője. Ez azt jelenti, hogy amikor napfelkeltekor úgy látjuk, hogy a Nap már teljes egészében felkelt, valójában még csak azután fogja átlépni a horizontot.

Ráadásul a Nap és a Hold a csillagokkal ellentétben nem pontszerűnek látszódnak, kiterjedtségükönél fogva más lesz a refrakció mértéke a tetején és az alján, így még össze is lapulnak, méghozzá nem szimmetrikusan, fentről lefele haladva lesz egyre nagyobb a torzítás mértéke, mint egy pattanó guminabda (ld. 3.9. ábra).



3.9. ábra. *Lékgöri refrakció hatása a holdon.*

1982-ben Bennett viszonylag egyszerű és mégis pontos ($\pm 0,001^\circ$) képletet írt fel, mely a fokban megadott látszólagos (apparent) h_a emelkedési szögre adja meg az R refrakció nagyságát fokban.

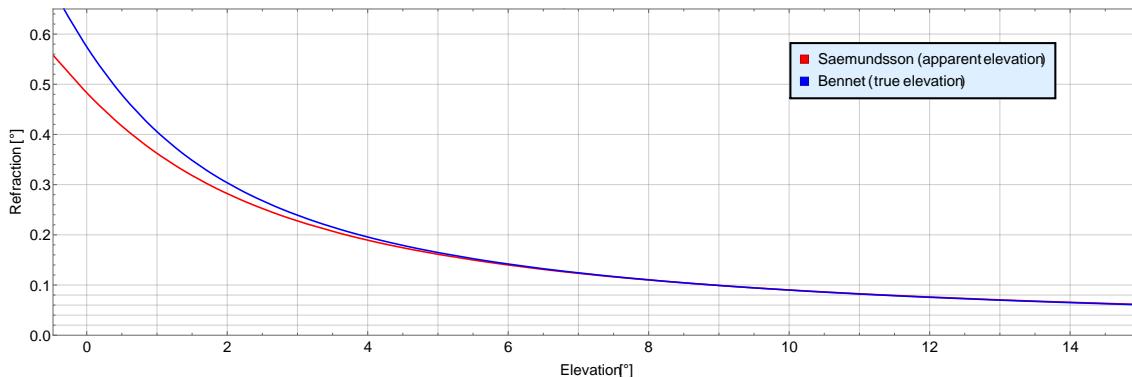
$$R = \operatorname{ctg} \left(h_a + \frac{7,31}{h_a + 4,4} \right) / 60 \quad (3.2)$$

Ennyivel van valójában alacsonyabban a látott égitest, azaz $h = h_a - R$.

A probléma inverze, hogy a valódi h emelkedési szögű égitestet mennyivel látjuk lejjebb, erre Sæmundsson adott egy hasonló alakú képletet, mely egy égitest fokban mért valódi h emelkedési szögére adja meg a refrakció nagyságát fokban:

$$R = 1,02 \cdot \operatorname{ctg} \left(h + \frac{10,3}{h + 5,11} \right) / 60 \quad (3.3)$$

A két függvényt a 3.10 grafikonon láthatjuk.



3.10. ábra. Légbeli refrakció értéke az emelkedési szög ismeretében.

Mindkét képlet 10°C , 101.0 kPa nyomásra lett kiszámolva, más értékek esetén a szükséges kompenzációs szorzó:

$$\frac{P}{101} \cdot \frac{283}{273+T} \quad (3.4)$$

3.3. Hold

A Nap és a Hold szögátmérője körülbelül azonos (gondoljunk pl. a napfogyatkozásra) – fél fok. A napé pontosabban 0.52° azaz $31'$ (szögperek), a Holdé 0.54° , azaz $32'$). Viszont a Hold sokkal könnyebben detektálható, mert fényudvaraikat is figyelembe véve sokkal kisebb, mint a Nap és kontrasztosabb a körvonalá. Ráadásul általában nem is egyszerre találhatóak az égen, így mindenkorban érdemes a Napon kívül a Hold modelljét is használni a mérések során.

3.3.1. Holdfázis

Mivel Nappal szemben a Holdnak különböző fázisai ismertek (ld. 3.11. ábra) a 29,53059 napos periódusában, ezek kiszámítása segítheti a detekció pontosabb illeszkedését, vagy a dátum meghatározását.



3.11. ábra. Holdfázisok

3.4. Algoritmus

A megvalósított program a modellek alapján folytonosan finomodó felbontásban keresi meg a legjobban illeszkedő hiányzó paramétert. Explicit egyenlet, megoldás nem igazán alkalmazható a különböző eltérések miatt. Már eleve az égitestek ellipszis pályán történő mozgása és a Föld gömb helyett ellipszoidal való modellezése nagyon bonyolulttá teszi az asztronómiai számításokat. De ehhez még hozzájön a mágneses eltérés, légköri refrakció is.

3.5. Égitest detekció

A legegyszerűbb (jelenleg általam is használt megoldás) az égitestek bemérésére, hogy a kamerát az égitest felé fordítjuk, minél pontosabban. Ennél azonban robosztusabb megoldást kaphatunk, ha az égitesteket, például a Napot, vagy a Holdat, mint fényforrásokat detektálunk a kameraképen és az irányeltést is kiszámoljuk. A Nap és a Hold szögátmérője körülbelül azonos, fél fok, azaz 30 szögperc. Mivel ismert a kamera látószöge is, ezért kiszámolható a várható méret, mely segítheti a fals pozitív találatok kiszűrését.

3.6. Távolság számítás

A tesztelések során a kapott pozíciót (fokrajzi koordinátákat) összehasonlítottam az ismert, Budapesti koordinátámmal, az eredmény szemléletesebb értelmezéséhez pedig távolságot képeztem belőlük. A Föld felszínén azonban csak kis távolságokra működik jól az euklidészi síkgeometriában alkalmazott távolságképlet, így megvizsgáltam a gömbi és ellipszoid távolságokat is.

3.6.1. Sík távolság

Kis távolságok esetén, ahol a földfelület még jól közelíti a síket, bátran alkalmazhatjuk az euklidészi távolságot: vesszük a két koordinátatengely mentén számított távolságokat és a derékszögű háromszögekre ismert Pithagorasz-tétellel ezek négyzetösszegének gyöke adja a távolságot:

$$d = \sqrt{\Delta x^2 + \Delta y^2} \quad (3.5)$$

Azonban a koordináták a fokhálózatban ismertek, így vegyük először ezen szögek radiánban vett különbségét:

$$\Delta\phi = |\phi_1 - \phi_2|, \quad (3.6)$$

$$\Delta\lambda = |\lambda_1 - \lambda_2| \quad (3.7)$$

majd mint középponti szögekből, a Föld R sugarának ismeretében adódik a körívek hossza:

$$\Delta x = r \cdot \Delta\lambda, \quad (3.8)$$

$$\Delta y = R \cdot \Delta\phi, \quad (3.9)$$

A 3.8. egyenletben szándékasan szerepelt kis r , hiszen a szélességi körök a gömbön nem főkörök (a hosszúsági körökkel ellentétben), így az R csak az egyenlítő közelében lenne jó közelítés. A sarkok fele haladva a megfelelő körszeletek sugara egyre kisebb és persze különbözik ϕ_1 és ϕ_2 esetén is, így számolunk például az átlagos (mean) szélességi fokkal:

$$\phi_m = \frac{\phi_1 + \phi_2}{2}, \quad (3.10)$$

$$r = \cos \phi_m \cdot R \quad (3.11)$$

Összefoglalva tehát a **sík távolságképlet**:

$$d_{plane} = R \cdot \sqrt{\Delta\phi + (\cos \phi_m \cdot \Delta\lambda)^2}, \quad (3.12)$$

ahol R a Föld sugara (≈ 6371 km), $\Delta\phi$ és $\Delta\lambda$ pedig a szélességi és hosszúsági fokok különbsége, ϕ_m pedig az átlagos szélességi fok, mind radiánban mérve.

3.6.2. Gömbi távolságképlet

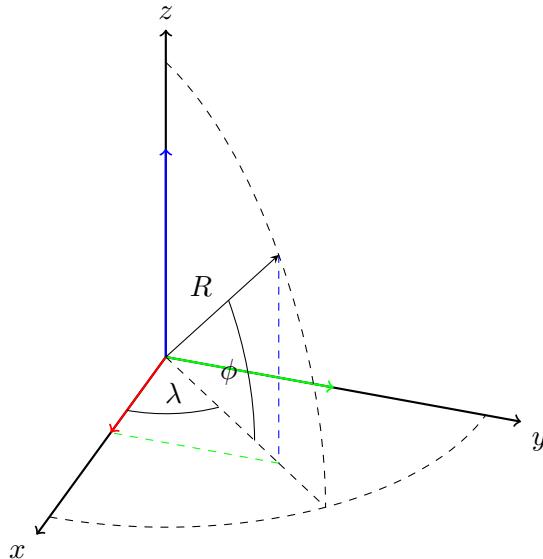
Nagyobb távolságokra, vagy egyenlítőtől távolabbi helyeken biztosabb és pontosabb eredményt kaphatunk, ha kiszámoljuk pontosan a gömbi távolságot, azaz a két ponton átmenő főkör két pont közé eső körívének d hosszát, melyhez a δ középponti szög tartozik.

Írjuk át a két földrajzi koordinátát a 3.12. ábrán látható vetületek segítségével Descartes

koordinátákra:

$$\mathbf{v}_1 = (R \cdot \cos \phi_1 \cdot \cos \lambda_1, R \cdot \cos \phi_1 \cdot \sin \lambda_1, R \cdot \sin \phi_1) \quad (3.13)$$

$$\mathbf{v}_2 = (R \cdot \cos \phi_2 \cdot \cos \lambda_2, R \cdot \cos \phi_2 \cdot \sin \lambda_2, R \cdot \sin \phi_2), \quad (3.14)$$



3.12. ábra. Gömbi és Descartes koordináták megfeleltetése

majd vegyük a két vektor skalár szorzatát (megfelelő koordináták szorzatának összege):

$$\begin{aligned} \mathbf{v}_1 \cdot \mathbf{v}_2 &= \left(R^2 \cdot \cos \phi_1 \cdot \cos \phi_2 \cdot \cos \lambda_1 \cdot \cos \lambda_2 + R^2 \cdot \cos \phi_1 \cdot \cos \phi_2 \cdot \sin \lambda_1 \cdot \sin \lambda_2 + R^2 \cdot \sin \phi_1 \cdot \sin \phi_2 \right) \\ &= R^2 \cdot \left(\cos \phi_1 \cdot \cos \phi_2 \cdot \underbrace{(\cos \lambda_1 \cdot \cos \lambda_2 + \sin \lambda_1 \cdot \sin \lambda_2)}_{\cos(\lambda_1 - \lambda_2) \text{ (addíciós tétel)}} + \sin \phi_1 \cdot \sin \phi_2 \right). \end{aligned} \quad (3.15)$$

A skalár szorzatról ugyanis tudjuk, hogy a vektorok hosszának (esetünkben R) és közrezárt szögük koszinuszának szorzata:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = R^2 \cdot \cos \delta, \quad (3.16)$$

így a 3.15. és 3.16. egyenletek jobb oldalait egyenlővé téve, R^2 -el egyszerűsítve:

$$\cos \delta = \cos \phi_1 \cdot \cos \phi_2 \cdot \cos (\lambda_1 - \lambda_2) + \sin \phi_1 \cdot \sin \phi_2, \quad (3.17)$$

adódik, melynek két oldalának ha vesszük az \arccos függvényét, megkapjuk δ -t, melyből adódik az ívhossz ($d = R \cdot \delta$). Így végül a **gömbi távolság formula**:

$$d_{sphere} = R \cdot \arccos (\sin \phi_1 \cdot \sin \phi_2 + \cos \phi_1 \cdot \cos \phi_2 \cdot \cos \Delta \lambda)$$

(3.18)

Gyakran használt még a **Haversine formula** is, ami kis (kilométer közeli) távolságokra is alkalmazható, de ma már dupla lebegőpontos számításokkal nem jelentős a fenti képlet hibája kis távolságokra sem.

3.6.3. Ellipszoid távolság

A Föld azonban nem is pontosan gömb alakú, jobban közelíthető egy lapult forgásellipszoiddal (pl. a legelterjedtebben használt *WGS84* modell által leírt paraméterekkel).

Az ellipszoidra azonban nem ismert explicit pontos képlet, a legismertebb módszer a **Vincenty formula** egy iteratív közelítést használ.

Összehasonlító tesztjeim során sokszor nem adott választ a módszer, mert ismert probléma, hogy átellenes közeli pontokra nem konvergál az algoritmus. Ezt úgy oldottam meg, hogy ezen esetekben felvettem egy köztes pontot (szélességi és hosszúsági fokok átlagát, a kisebbik szög szögfelezőjeként) és újra lefuttattam a keletkezett 2 szakaszra, majd összeadtam a hosszokat. Ezt a javítást a `hu.ureczky.utils.astro.distancecalculator.DC_Ellipsoid.tryMyHack()` függvény tartalmazza, ez hívódik meg az üres visszatérés helyett. Lehet, hogy így nem a minimális távolságot adja vissza, de legalább közel jó eredményt ad. Általában persze nagyon elronthatná a számítást egy ilyen köztes pont beszúrása, viszont az átellenes pontoknak a szélességi koordinátájuk pont ellenkező. Ha a gömbi modellt nézzük, ott ezen köztes (egyenlítőre eső) pont épp a rajtuk áthaladó főkörre esne. Tehát ha figyelembe vesszük, hogy a gömbi távolság nem sokban tér el az ellipszoidon mértől, akkor jelentős hibát nem okozhat.

2013-ban azonban *Karney* publikált egy megoldást a problémára.

3.6.4. Implementáció

A fenti 4 módszerre (sík, gömbi, gömbi-Haversine, ellipszoid-Vincently) készítettem egy-egy osztályt a `hu.ureczky.utils.astro.distancecalculator` package-be, ahol a `DC_PLANE`, `DC_SPHERE`, `DC_HAVERSINE`, `DC_ELLIPSOID` implementációk egy közös absztrakt `DistanceCalculator` osztályból származnak és egy `create()` metódussal gyártható le a kívánt implementációs kalkulátor egy enum paraméter segítségével.

Mivel a programban ez a távolságszámítás nem ciklusban fut, csak a végeredmény ellenőrzésre használom, ezért a leg pontosabb ellipszoid-távolságot használom.

3.7. A program bemutatása

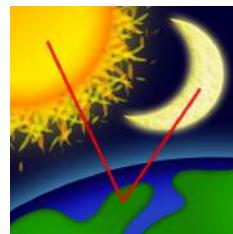
Az alkalmazás a . fejezetben ismertetett elérési útról telepíthető az előre fordított apk fájl készülékre másolásával és telepítésével (vagy a mellékelt kódból újrafordítással).

3.7.1. Főképernyő

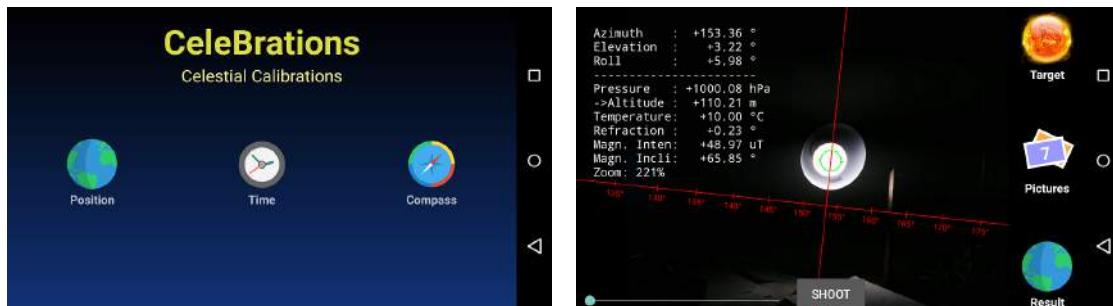
A 3.13. ábrán látható indítóikkal elindítva a programot, a 3.14a kezdőképernyő fogad minket, ahol 3 féle kalibrációt közül választhatunk:

- *Position* – helymeghatározás idő, irány (és más szenzorok) segítségével (égi navigáció funkció).
- *Time* – idő meghatározása GPS pozícióból és irány ismeretében (napóra funkció)
- *Compass* – irány meghatározása GPS pozícióból és az idő ismeretében (iránytű funkció),

melynek következtében a 3.14b. ábrán látható AR nézetre kerülünk.

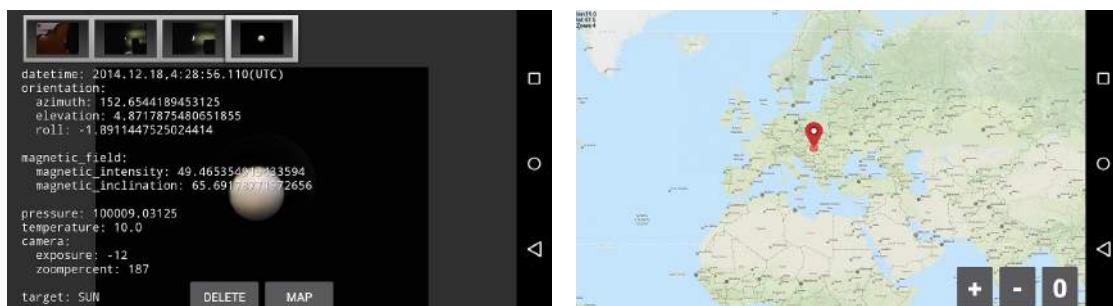


3.13. ábra. A CeleBrations indítóikonja



(a) Főképernyő

(b) AR nézet



(c) Mérési pontok

(d) Térkép nézet

3.14. ábra. Képernyőképek a CeleBrations alkalmazásról

3.7.2. AR nézet

Ebben a nézeben mérhetjük be az égitest pozícióját, melynek segítségével a választott értéket meghatározhatjuk.

A háttérben a kameraképet láthatjuk, ami feletti rétegen extra információk terjesztik ki a valóságot. Piros vonal mutatja a horizontot, melyen 5° -onként szerepelnek az azimut irányok is. Erre merőlegesen található egy másik piros vonal, mely az aktuális nézetünk azimutját szemlélteti. A pontos érték a bal felső sarokban található információs mezőről

olvasható le. Középen található egy 1° -os szögátmérőjű zöld kör, mely a kamera pontos nézeti irányát jelöli. Ez a kör segít az égitestet becélozni.

A kamerakép bal alsó szélén található egy csúszka, mellyel a zoom értékét állíthatjuk. A beállított %-os értéket szintén a bal felső sarokban található információs mezőben tekinthetjük meg. A kamerakép alján található egy *Shoot* feliratú fényképező gomb, mellyel a bemért égitestről készíthetünk fényképet, a szükséges metaadatokkal együtt.

A bal felső sarokban található információs mezőben található a már említett zoom érték és nézeti irány (azimut (*Azimuth*), emelkedési szög (*Elevation*), bedöntési szög (*Roll*)). Ezen kívül leolvashatjuk a pillanatnyi légnormális (*Pressure*) értékét, mely az ezt követő tengerszint feletti magasság közelítőleges meghatározására szolgál. Ezt követi a légköri hőmérséklet (*Temperature*), ha nincs a készülékben ilyen szenzor, akkor konstans 10° -ot mutat. Az előbbi két érték az emelkedési szöggel együtt a légköri elhajlás, refrakció (*Refraction*) kiszámításához szükséges. Horizont közelben ennyi fokkal van lejjebb valójában a látott égitest a személyhez képest. Ezután a mágneses tér két jellemzőjét találhatjuk, a mágneses térerősséget (*Magnetic intensity*) és mágneses lehajlást, inklinációt (*Magnetic inclination*), mely értékek szintén eltérnek a föld egyes pontjain, így segítik a helymeghatározást.

Jobb oldalt találunk még 3 gombot:

- *Target* – Célpont kiválasztása: Nap (alapértelmezett), vagy Hold.
- *Pictures* – A készített képek, mért értékek megtekintése (lásd. 3.7.3. alfejezet).
- *Result* – A kiszámolt eredmény térkép nézetben megtekintése (lásd. 3.7.4. alfejezet).

3.7.3. Mérési eredmények nézet

Ebben a nézetben áttekinthetjük a mérési pontokat. Felül található egy scrollozható filmszalagos nézet, melyben a készített képek miniatűrjét találjuk. Egyiket kiválasztva alul megjelenik nagyban a készített kép, valamint szintén scrollozható módon a kép készítésének szükséges metaadatai, idő, irány, szenzorértékek. Alul található még 2 gomb, a *Delete* gombbal kitörölhetjük a kiválasztott mérési pontot, a *Map* gombbal pedig megtekinthetjük térképnézetben a behatárolt pozíciót.

3.7.4. Térkép nézet

Ebben a nézetben a kiválasztott mérési ponthoz – vagy az AR nézetből választva az összes mérési ponthoz tartozó – legvalószínűbb tartózkodási helyünket tekinthetjük meg térkép nézetben. Három gomb található a jobb alsó sarokban, a + és - gombokkal a térkép zoom szintjét állíthatjuk, a 0 gombbal pedig középre igazíthatjuk a térképet a célponthoz viszonyítva.

3.8. Értékelés

Habár a szenzorok érzékenysége valóban figyelemre méltó a hordozható eszközökben, sajnos a pontosságuk több fokot is eltérhet. A felhasznált modellek, számítások alkalmasak

lennének akár 10 km-es pontosság elérésére is, ami még tovább finomítható lenne, sajnos a felhasznált szenzorok még nem rendelkeznek a szükséges pontossággal.

Az elmúlt évek hirtelen, rohamos fejlődése a szenzorok pontosságában lassulni látszik. Eleinte egyre több szenzor került a készülékekbe (1 tengelyű gyorsulásmérőt követte a 3 tengelyű gyorsulásmérő, iránytű, majd a giroszkóp), melyek egyre jobban kiegészítették egymást, főként *sensor fusion* megoldásokkal. Ezután a technológia a miniatűrizálásra ment rá, folyamatosan egyesítették a szenzorokat és egyre kisebbek is lettek, mára már egyben kapható 9 szabadságfokú IC akár 3x3x1 mm-es kivitelben. Ennek előnye ugyan, hogy chip-en belül, kisebb késleltetéssel, jobb sensor fusion megoldások születhetnek és a CPU-ról is leveszi a számítási terhet, ha a számítások nagy részét célhardver végzi. Ugyanakkor azáltal, hogy AR feladatokra alkalmassá váltak az eszközök, kevés a motiváció a még pontosabb szenzorok megalkotására, mely a dolgozatomban említett és hasonló újszerű alkalmazásokra lehetőséget adna. Ehelyett érthető módon fontosabb a miniatűrizálás és költségcsökkentés. Szerencsére a fejlődés nem állt meg és további mozgatórugót jelenthet például a virtuális valóságok, head mount display-ek, okoszemüvegek fejlesztése, hiszen ott előtérbe került az a probléma, hogy a nem elég pontos és nem elég gyors válaszidejű megoldások valószerűtlen élményt nyújtanak és rosszullétet is okozhatnak.

3.9. Továbbfejlesztési lehetőségek

- Tesztelések, szenzorértékek hibáinak pontosabb felderítése, kalibrációs módszerek.
- Képfelismeréssel (OpenCV, vagy RenderScript) nem kellene pontosan az objektumot célozni, így egyszerűbb lenne a használata és pontosabb lenne a bemért pozíció.
- Légköri refrakciós torzítás beépítése a programba. A képfelismerés előfeldolgozó egységeként egyszerűbb lehet a kördetektálás is horizont közeli helyzetben.
- Stabil fényképezés (stable shoot): időablakban figyelni a szenzorértékek ingadozását és stabil helyzetig késleltetni a fényképezést. Ezzel egyszerűbb megoldás a fényképezés gomb megnyomásának hatására történt bemozdulás, másrészt nem a felhasználón múlna, hogy mikor ítéli stabilnak a helyzetet.
- Az új (Android 5.0-ban bevezetett) Camera2 API-ra áttérni.
- A jelenlegi WMM modell Androidos implementációja helyett más, pontosabb implementáció használata. Gyorsabb C/C++-os implementáció használata NDK-ból.
- A régebbi 2005-ös és 2010-es WMM modellek helyett az 1900-ig kiterjesztett IGRF modell használata.
- A jelenlegi WMM modell helyett pontosabb modell (EMM, vagy még pontosabb HDGM modell használata)
- Holdfázis felhasználása képfelismerés pontosítására.
- Számítások gyorsítása. Párhuzamosítás több magra (szálak indításával, vagy Renderscript-tel).

3. FEJEZET. ÉGITEM FELISMERÉS ALAPÚ KALIBRÁCIÓ.9. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

- Play store-ba feltöltés, Github-on megosztás.
- Programkód stabilizálása: hibakezelések, kivételek kezelése, null pointerek ellenőrzése, szabad hely ellenőrzése, mountolva van-e az sd kártya, írható-e, állapotok kiemelése bundle-be, tesztelés, kódtisztítás.
- Tesztek írása, főleg a utility-khez, külön projektbe rakni.

4. fejezet

Ortofotó transzformáció

Ebben a fejezetben azt vizsgálom meg, hogy miként segíthetik az okostelefonokba épített szenzorok a sík objektumok digitalizálását. Bemutatom a felmerülő problémákat, jelenlegi megoldásokat és ezek hiányosságait. Egy javítási lehetőséggel kiszámolom a nem merőlegesen készített vízsintes képek helyreállításához szükséges transzformációt, amelyre alkalmazási példákat és továbbfejlesztési lehetőségeket mutatok.

4.1. Problémafelvetés

Szkenner hiányában (viszont az okostelefonok korában) gyakori eset, hogy telefonnal készítünk képet iratokról. Ez ugyan nem ad ugyanolyan letisztult, kontrasztos, olvasható képet, de egy kis utómunkával könnyen hasonlóra javítható a minősége. Eltekintve a fehéregyen-súly, kontraszt, képelesség beállításától, az egyik fő probléma, hogy általában nem sikerül teljesen merőleges szögből fényképezni, így nem elég csak a széleit levágni, még torzítani is kell a képet.

Itt nem csak az emberi pontatlanságról van szó, néha szándékasan kell máshonnan fényképezni. Kevés környezeti fény esetén (ld. 4.1a. ábra) például sötét lesz a kép és nem is biztos, hogy a kamera megtalálja a fókuszt.

Valamennyit ugyan világosíthatunk utólag is (ld. 4.1b. ábra), de fény hiányában kevesebb információ is rögzül a képről, így az zajos lesz, a színek pedig pontatlanok lehetnek. A másik probléma, hogy gyakran a kamera nem is találja el a fókuszt – ami főleg akkor zavaró, ha már csak világosítás során vesszük észre. Ilyenkor életlen lesz a kép, amin csak minimális mértékben tudunk szoftveresen javítani.

Ezért érdemes általában nagyobb megvilágítást használni, a megfelelő képminőség előréssének céljából. Ráadásul adja is magát a lehetőség, hogy amikor tudjuk, használjuk a telefonba épített segédfényt, vakut. Ez viszont egy másik problémát vet fel egy fényes felület esetében, mert ilyenkor a vaku visszacsillanása (ld. 4.1c. ábra) miatt elveszhet a kép egy részéről az ottani információ, de jobb esetben is zavaró lesz.

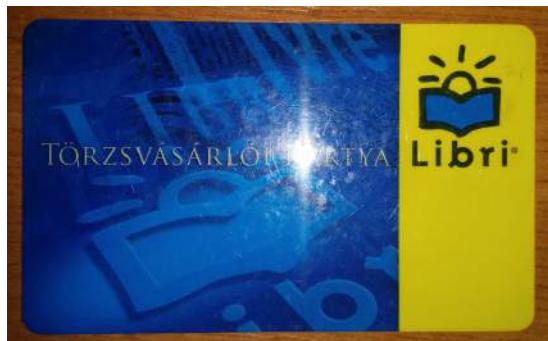
Gyakori eset például egy laminált dokumentum, fénykép, vagy plasztik kártya (névjegykártya, vagy valamely okmány) digitalizálása. Matt anyagok tekintetében is jó trükk lehet egy gyűrött, görbült lap esetén, ha szkenner hiányában magunk rakunk rá egy üveglapot, hogy kiegyenesedjen, vagy belehelyezzük egy genotherm-be.



(a) Alulvilágított helyiségben készített kép: sötét, nehéz a fókusz megállapítani



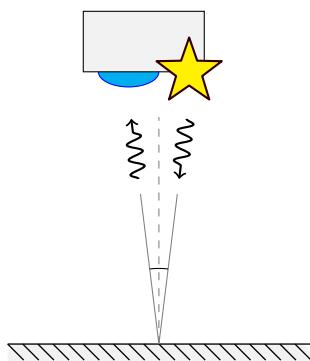
(b) Utólagos világosítást követően: zajos, esetenként életlen



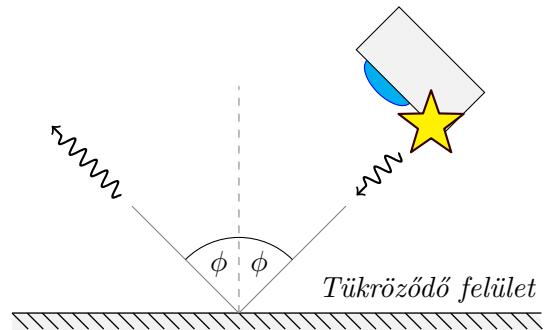
(c) Vakuval készítve: világos, nem zajos, fókuszált, de becsillan



(d) Oldalról készítve vakuval: világos, homogénebb, de torz



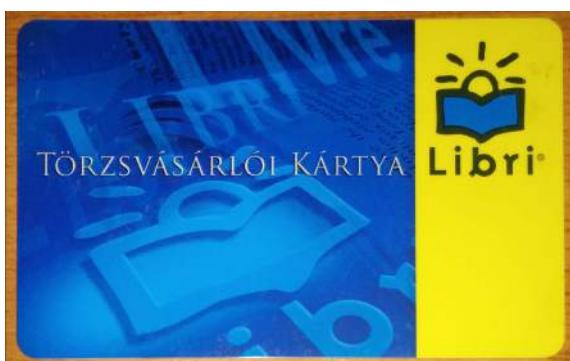
(e) Merőlegesen fényképezve a vaku fénye visszaverődik és megcsillan a képen



(f) Oldalról fényképezve a vaku fényének tükrözése nem jelenik meg a képen



(g) Perspektív rektifikálást követően: helytelen lehet a képarány



(h) Ortofotó transzformált kép: világos, fókuszált, helyes képarány

4.1. ábra. Fényes objektum digitalizálásának problémája segédfényes fényképezővel

Én személy szerint például a spirálozásnál használt tisztább, átlátszóbb, vastagabb fedőlap mögé szoktam rakni, mert az jobban leszorítja, kiegyenesíti. De fényes felület lehet egy asztallap is, vagy gyakran fényképeznek vitrin mögé – akár egy múzeumban, vagy épp a középiskolai tabló digitalizálásánál (ahol igencsak sajnálnánk, ha épp a mi tablóképünk helyén lenne a fénylő folt).

Ilyen esetekben tehát jó megoldás, ha nem merőlegesen fényképezünk, hanem egy másik szögből (ld. 4.1d. ábra), mert akkor a vaku fénye nem a kamerába fog visszatükröződni (ld. 4.1e., 4.1f. ábra), így nem okoz csillanást, mégis világos lesz a kép, egyenletesebb intenzitással (bár egy nagyobb felület esetén a távolabbi pontok esetleg kevésbé lesznek megvilágítottak).

Az ilyen „laposabb” szögekben készült kép perspektív torzulásnak esik áldozatul, mely helyreállítható (ld. 4.1h. ábra) és végeredményben világos, éles, olvasható képet kapunk csillogás nélkül.

De ha megfelelő is a világítás és nem a vaku csillan vissza, akkor is lehet, hogy visszatükröződik valami. Például ha fényképet akarunk készíteni egy tükörről (vagy bármilyen tükröződő felületről, ami lehet polírozott fémfelület, vagy víz is), de nem szeretnénk, hogy a tükrökünk látszódjon benne, viszont fontos, hogy szemből látszódjon a kép. A 4.2. ábrázorozaton erre láthatunk egy megoldást: ha kicsit más szögből fényképezünk és utólag transzformáljuk vissza a síkot, eltűnik a tükrökép. Egy kicsit ugyan valószerűtlen lesz a tükrözés, de ilyen kis mértékű szögeltérésnél még nem feltűnő elsőre.



(a) Szemből fényképezve (b) Oldalról fényképezve (c) Végül transzformálva

4.2. ábra. Tükörkép eltűntetése szemből fényképezés esetén

Az is elképzelhető még, hogy a fotózandó felületet eleve nem is merőlegesen szándékozzuk fotózni, csak épp a megfelelő beállításban valami más tükröződik (tehát nem a vaku, de például a nap, vagy csak a világos égbolt). Más beállítás pedig nem lenne alkalmas a célra, mert nehéz lenne például elolvasni a szöveget. Érdekes, de két tetszőleges kameraállás között megadható a transzformációs mátrix (sík alakzat esetén).

De nem csak a tükröződések miatt lehet hasznos kameranézetet változtatni. Lehet például, hogy azért nem tudunk merőlegesen fényképet készíteni, mert megközelíthetetlen, vagy szándékban nem akarjuk megközelíteni az ideális fényképezési pontot.

Egy óriási aszfaltrajz esetén például nem tudunk általában több méter magasra emelkedni a rajz közepénél. De még egy kisebb aszfaltrajz esetén – ahol megfelelő lenne a szemmagasság – is nehéz úgy képet készíteni, hogy ne legyen benne a lábunk. Ilyen esetekben is egy pár lépéssel odébbről tudunk csak kisebb szögben képet készíteni.

Légi fotózásnál (térképezésnél) sem pásztázzák végig az egész területet, hiszen ha az oldalról készített kép is helyreállítható, akkor kevesebb utat kell bejárnia a repülőnek, ami gazdaságosabb. Bizonyos esetekben pedig a repülő sem képes megközelíteni az ideális fényképezési helyszínt (például egy alagút, híd, vagy valamilyen fentről fedett terület esetén) és kénytelen lehet akár még a tűrhetőnél is kisebb szögben készíteni a képet.

Még inkább igaz lehet ez egy szűk alagútrendszerben fényképező régészrobotnál, ami egy nehézkesen megközelíthető, ősi írásról készít képet, aminek elolvasásához előnyösebb a merőleges szögben készített kép a helyes arányokkal. A BME - AUT¹ tanszéken is sok kutatási projekt zajlik kvadrokopterekkel, melyek túlnyomó része AR alkalmazásokhoz kapcsolódik. Ezeknél nagy segítséget jelentene, ha például egy marker, vagy más mintázat felismerését nem a közvetlenül látott képen, hanem már az orto-transzformált felülnézeti képen lehetne elvégezni.

Csoportosítva tehát néhány alkalmazási lehetőség:

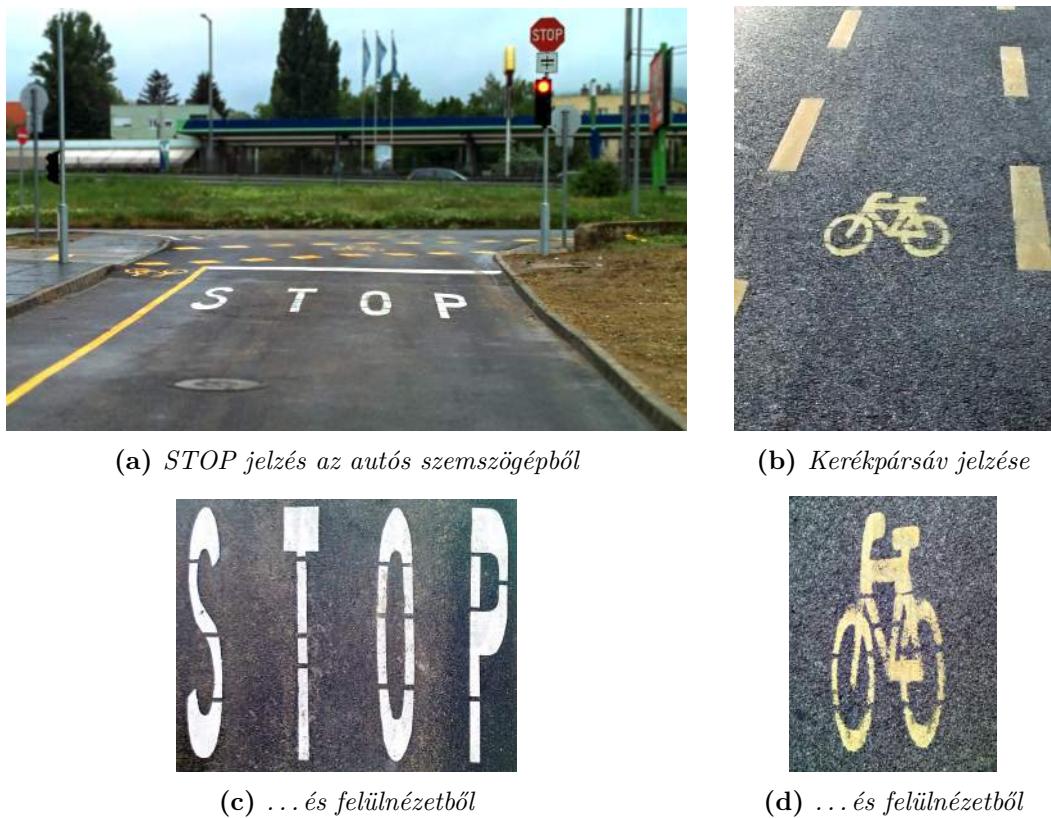
- Iratok, számlák, fényképek házilagos digitalizálása
 - Emberi pontatlanságból adódó szabálytalanság automatikus kiküszöbölése
 - Tükröződő felület vakuval történő megvilágításánál csillanás kiküszöbölése
- Fotó nézetek közötti változtatáshoz
 - Tükröződés miatt
 - Hozzáérhetetlenség miatt
- Kvadrokopteres, robotos alkalmazások
 - Ortofotózás (térképezés, digitalizálás)
 - Mintázatfelismerés elősegítése

¹AUT: Automatizálási és Alkalmazott Informatikai Tanszék

4.2. Perspektív torzulás

Az előző fejezetben említett perspektív torzulásra (perspective distortion) néhány esetben szándékosan odafigyelünk. Közlekedési jelzésekkel például felismerhetetlenek, vagy olvashatatlanok lennének az úttestre festett figyelmeztető jelzések, ill. feliratok a torzítás miatt, hiszen az autósok, biciklisek nem felülről, hanem laposabb szögből látják az úttestet. Éppen ezért megpróbálják úgy felfesteni a jelzéseket, hogy a torzítást követően is felismerhető/olvasható maradjon a jelzés (ld. 4.3. ábra).

Ez a torzulás annál jelentősebb, minél laposabb szögben látjuk. Így a különböző sebességgel használatba vett területeken (amíg a városban például csak pár tíz méterrel figyeljük előre a jelzéseket, addig az autópályán több száz méterrel), valamint eltérő méretű felfestések esetén (kisebb felfestést csak közelebbről látunk) más-más lehet a torzulás mértéke.



4.3. ábra. *Közlekedési jelzések szándékos elnyújtása*

Itt azonban csak egy egyszerűsített transzformációról van szó, minden össze lineárisan nyújtják meg függőleges irányban a jelést, ugyanis nincs egy kitüntetett távolság, sem magasság (lehet ugyanis süllyeszített autó, vagy emeletes busz), aminek megfelelően pontosabban lehetne beállítani az oldalirányú torzulást. Ennek hiányában a látott ábra/felirat egy kicsit „befele dől”, ahogyan a 4.3a. ábrán is látszik a „STOP” feliraton. De ha kellően távolról nézzük, és így elegendően kis szög alatt látjuk, akkor a középpontos vetítés megközelíti a merőleges vetést, ahol ez a modell megfelelő is lenne, ezért is olyan kicsi az eltérés és használható eredményesen. Ráadásul még talán előnyös is, hogy nem olyan életszerű – így nem keverhető össze egy akadályal, látszódik, hogy az út síkjában van.

Ennél látványosabbak az ún. *anamorfikus* (anamorphic) torzítású műalkotások, melyek csak egy bizonyos nézőpontból válnak felismerhetővé. Julian Beever – napjaink egyik legismertebb aszfaltművészé – például olyan aszfaltrajzokat készít, melyek egy adott nézőpontból térbeli jelenet hatását keltik (lásd például a 4.4. ábrán). Itt már látszik, hogy a perspektivikus torzulás nem csupán lineáris nyújtásból áll, a távolabbi pontok jobban elnyúlnak hátra, mint a közeliek és oldalirányban is szélesednek, hogy a látott képen megtartsák szélességüket.



(a) Megfelelő nézőpontból

(b) És egy másik nézőpontból

4.4. ábra. Julian Beever egyik legtorzultabb (kb. 15 m-esre nyúlt), anamorfikus aszfaltrajza.

Lényegében azt a transzformáltat kell az aszfaltra rajzolni, amit a projektor képe is elszenned, amikor nem merőleges falfelületre vetítjük. Ha a projektorral csak vízszintes, vagy függőleges irányban térünk el a merőlegestől, akkor egy négyzet képe ilyenkor egy trapéz lesz – ezért hívják a projektoroknál ezt a torzulást *trapéztorzulásnak* (angolul keystone-effect, vagy tombstone-effect).

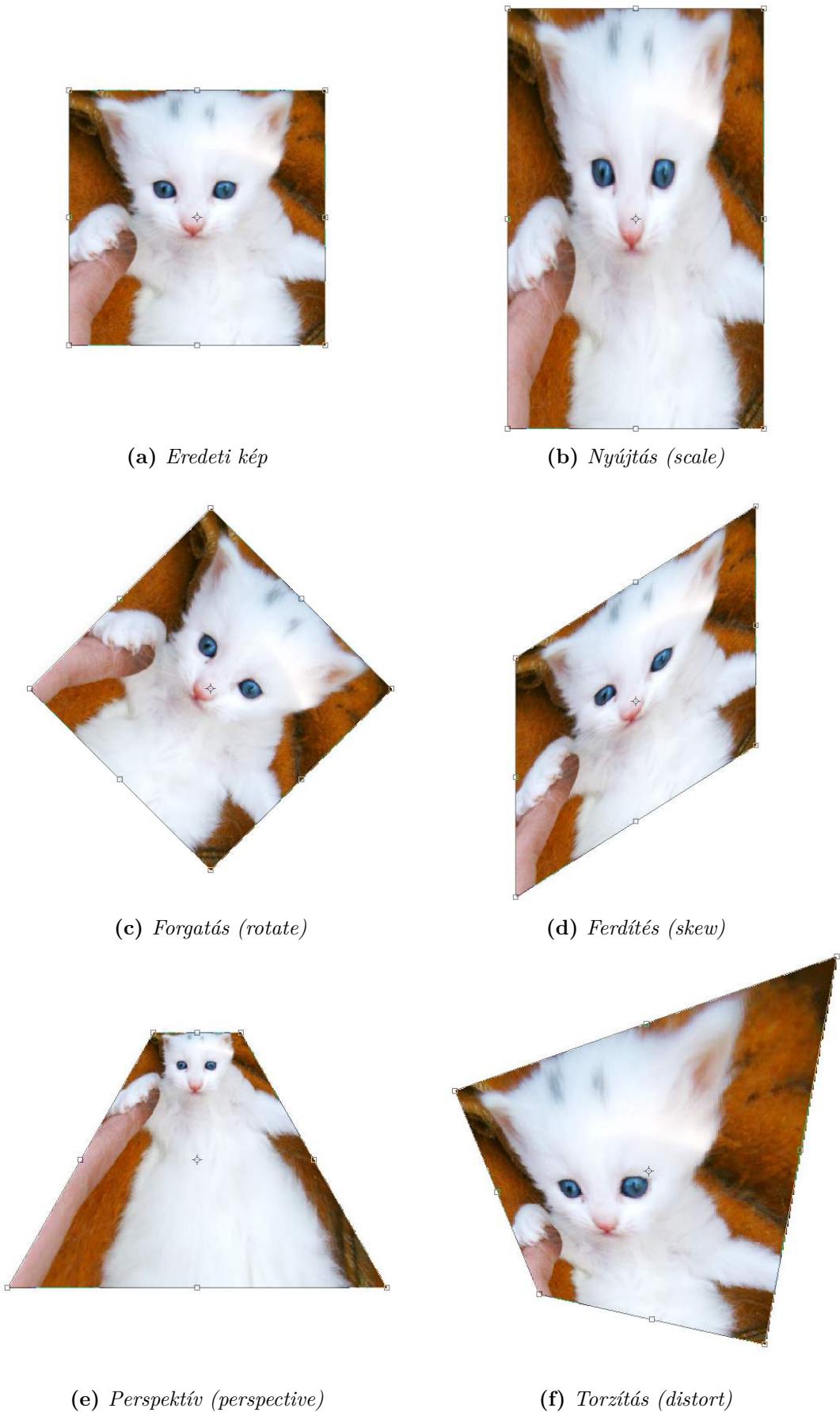
Perspektív torzulás lép fel abban az esetben is, amikor fényképezővel nem merőlegesen fényképezünk le egy síkot, például a magas épületeket általában „alulról” fényképezzük, ilyenkor az alsó (közelebbi fele) szélesebb lesz, a felső (távolabbi) pedig keskenyebb.

Ha az eltérés nem csak vízszintes vagy függőleges irányú, akkor már nem csak trapéz-, hanem általánosan perspektív torzulásról beszélünk. Összefoglalva a perspektív torzulásnak a következő speciális eseteit különböztethetjük meg:

Perspektív torzulás: Általános eset, amikor a nézeti, vagy vetítési irány nem merőleges a síkra. Egy négyzet képe egy általános négyzet lesz.

- **Trapéztorzulás:** Ha az eltérés csak függőleges vagy vízszintes irányú. Egy négyzet képe trapéz lesz. *Példák:* Amikor a projektor lentről, vagy fentről (tehát nem a vászon közepének magasságából) vetít.

- **Lineáris nyújtás:** Ortogonális (merőleges) vetületnél (amikor a középpontos vetítés a 0°-os látószöget közelíti). Egy négyzet képe téglalap lesz. *Példák:* közlekedési jelzések az úttesten.



(e) Perspektív (perspective)

(f) Torzítás (distort)

4.5. ábra. Képszerkesztő programokban található transzformációs eszközök

4.3. Perspektív rektifikáció

Az előző fejezetekben említett, perspektíva által transzformált képek eredeti formájára visszaalakítását, helyreállítását *perspektív rektifikációt* (perspective rectification) hívjuk, vagy trapéztorzítás esetén *trapéztorzítás-korrekciót* (angolul *tombstone correction*, *keystone correction*, vagy röviden csak *keystoning*). A képszerkesztő programok számos segítséget igyekeznek nyújtani ezek megoldására.

Egy képet (vagy kijelölt képrészletet) a 4.5. ábrán látható módon nyújthatunk (scale), forgathatunk (rotate), ferdíthetünk (skew), vagy állíthatjuk a perspektíóját (perspective). De tetszőleges csúcsát megfogva is torzíthatjuk (distort), amit gyakran szabadkézi torzításnak (free transform) neveznek. Ez ugyan általános torzításra ad lehetőséget (tehát az összes többi előállítható vele), viszont már kevésbé intuitív a használata. Elsőre ugyanis még nem látható, melyik sarokpont hova fog kerülni, a sorozatos mozgatásokkal több lépéssben érhető el, hogy a végeredmény az elvártnak megfelelő legyen. Egy ilyen beállítást mutat a 4.6b. ábra, ahol ugyan megkönnyítette a helyzetet, hogy egy téglalap a fő téma, de egy általános mintázatnál komoly fejtörést okozhat, hogy mikor melyik pontot merre mozgassuk, hogy életszerűbb legyen a kép.



(a) Általános helyzetű lap



(b) A kép szabadkézi torzítása az eredeti alakzat visszaállításának reményében



(c) Perspektív kivágás



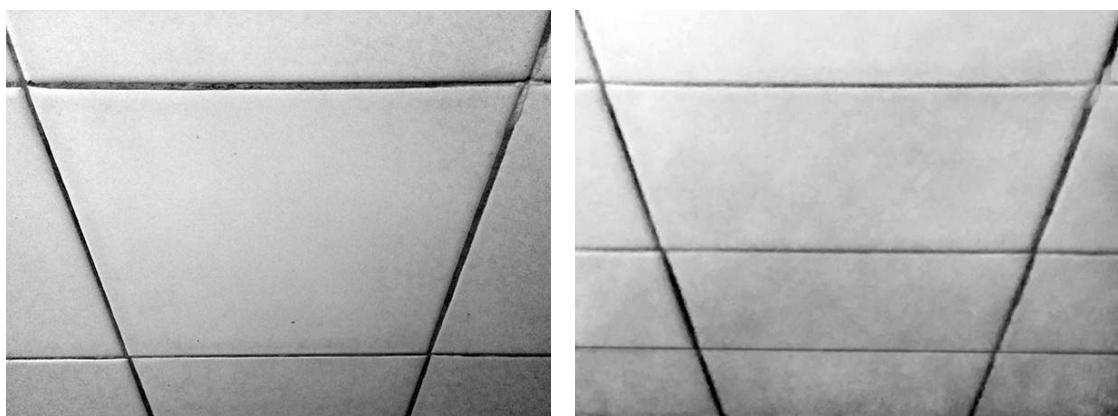
(d) Kivágott kép téglalappá transzformálva

4.6. ábra. Perspektív rektifikációs módszerek

Van azonban egy könnyebben használható eszköz is erre a nagyon gyakori esetre, amikor az eredeti mintázat pont egy téglalap (pl. iratok esetén), vagy sikerül olyan befoglalóját találni, ami téglalap. Ilyenkor a *perspektív kivágás* (perspective crop) nevű eszközzel transzformálhatjuk téglalappá a tetszőleges négyzet alakú kijelölést (ld. 4.6c., 4.6d. ábra). A módszer azonban önmagában még így sem determinisztikus, nem mindegy ugyanis a képarány sem.

Képarány

Négy tetszőleges pont kijelölése esetén még kevés az a feltételezés, hogy ezek téglalapot alkotnak, szükséges a képarány is. Legalábbis a fényképezés módjának és a képrészlet további ismereteinek hiányában ez még nem állapítható meg. A 4.7. ábrán látható kísérleten látszik, hogy egy 1×1 -es és egy 2×1 -es csempe ugyan annak a trapéznak látszik megfelelő beállítások mellett.



4.7. ábra. A kameraképen azonos (75° -os, azonos arányú) trapéznak látszó, de valójában 2 különböző arányú téglalap

Csupán egy képrészletből tehát nincs információink az eredeti képméret arányáról, nem mindegy ugyanis milyen látószögben készült a kép, illetve a kép melyik területén jelöljük ki a négyzetet.

A fényképekben elvileg van lehetőség a *metadata*-ban tárolni információkat a kép készítésének körülményeiről (mint a látószög, fókusztávolság, GPS pozíció), de a telefonos – beleértve a gyári – fényképező alkalmazások ezeket gyakran hiányosan töltik ki.

A Photoshop arány beállítása nélkül is megpróbálja kitalálni a képarányt, valószínűleg feltételez egy átlagos, kb. 50° -os látószöget, ami azonban csak feltételezés, így a képarány alkalmanként nem teljesen megfelelő (ld. 4.1g. ábra). Ráadásul a kép körbevágásának látószögekre vonatkozó hatását már nem követi, így ha több lépésben vágunk, igazítunk a kép kivágáson, egyre valószínűbb hogy eltorzul a képarány.

Így tehát manuálisan kell beírni a kívánt képarányt. Ez a leggyakoribb esetben, egy A4-es papírlap esetén ismert: $\sqrt{2}:1 \approx 1.41$, de más esetekben általában csak hozzávetőlegesen tudunk beállítani egy értéket, így az eredmény torz lehet. Ez néhány esetben nem túl feltűnő és talán nem baj, de néha fontos lenne az eredeti arányokat megtartani. Ha van a képen négyzet, kör, vagy ismert alakzat, akkor ezek segíthetik a helyes beállítás megtalálását, ezek hiányában viszont csak hozzávetőlegesen tudjuk beállítani. Viszont van, hogy nem is ismerjük az eredeti alakzatot (pl. nem is láttuk előben, vagy nem tudtuk megjegyezni), ilyenkor már tényleg csak hasra ütés szerűen tudunk tippelni. Még nehezebb a helyzet, ha a képen nincs is semmi kézzel fogható körvonal, alakzat, csupán egy mintázatot szeretnénk helyreállítani (például egy textúrázáshoz készített mintázat).

Mobilalkalmazások

A négyszög kijelölésű perspektív transzformálás elősegítésére viszont létrehoztak mobilalkalmazásokat, amelyeket például a Google Play-en [24] is találhatunk Android-os készülékünkönöz. Ezekkel már telefonon is elvégezhetjük a perspektíva rektifikálást, nem szükséges feltölteni a képet számítógépre és ott elvégezni a műveletet.

Néhány népszerű példa a teljesség igénye nélkül:

- **CamScanner** – A legtöbbet (10–50 millióan) letöltött alkalmazás [27].
- **Mobile Doc Scanner Lite** – 1-5 millió letöltés. [31].
- **Genius Scan** – 100-500.000 letöltés. [28].

A telefonos alkalmazásokon a négy csúcs kézzel való bejelölése kissé körülményesebben, pontatlanul végezhető el a kisebb képernyőn, mint egy nagy monitoron. Ennek mérséklésére viszont született megoldás, általában a csúcsot arrébb helyezik az ujjunktól, hogy ne legyen takarásban és fel is nagyítják a pont körüli képrészletet.

Ezek a programok minden élkereséssel keresik meg a dokumentum széleit, majd így alkamaznak rá perspektív rektifikálást. Ezt követően számos extra szolgáltatást nyújtanak még, mint a képjavítás (kontraszt, fehéregyensúly beállítás, fekete-fehér konvertáció), OCR² és a különböző gyors megosztási módok (Dropbox, Facebook, G-mail) integrálása.

²OCR: Optical Character Recognition, azaz optikai alapú karakterfelismerés

Probléma felvetés

Az említett és még sok más meglévő alkalmazást kipróbálva nem találtam olyat, ami a szükséges transzformációt képfelismerés nélkül, automatikusan állítaná elő a sík helyzetének feltételezésével, vagy bemérésével.

Pedig a telefonban lévő szenzorokkal mérhető lenne a készülék dőlése, iránya, így van potenciál ennek kihasználásában, mely sok területen alkalmazható lehetne. A fejezet hátra lévő részében ezt a lehetőséget vizsgálom meg – miként lehet csupán szenzorok segítségével perspektív rektifikációt végrehajtani.

4.4. Ortofotó transzformáció kiszámítása

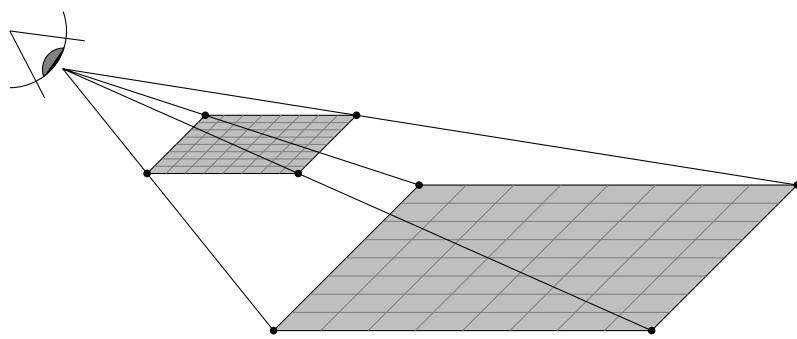
Elméletileg a pontos matematikai transzformáció meghatározásához szükséges lenne a tárgy távolságának, a kamera látószögének, fókusztávolságának és talán még egyéb paramétereknek is az ismerete.

A látószög és fókusztávolság egy telefon esetében szerencsére nagyjából állandó (ellenértében egy cserélhető objektíves, vagy optikai zoom-os géppel), így akár méréseket is lehetne végezni egy adott telefonnál, hogy kiderítsük. Szerencsére még erre sincs szükség, hiszen API-szinten is lekérdezhető. .

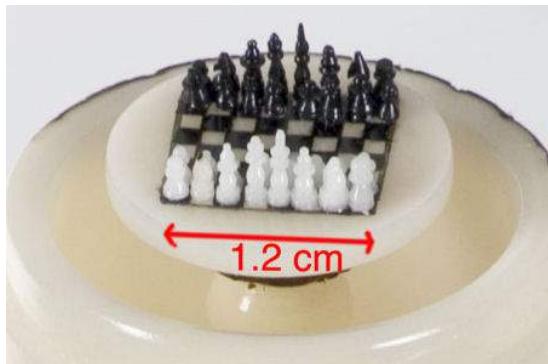
A tárgytávolsággal kapcsolatban viszont az az érdekes, hogy talán nem is szükséges. Ez elsőre merész gondolatnak tűnik, de végül is gyakran látunk képen olyan beállításokat makettekről, melyről azt hisszük, hogy életnagyságú.

Ha kameramodellként a középpontos vetítésre gondolunk, akkor az azonos látószög alatt látszó tárgyak megkülönböztethetetlennek tűnnek, függetlenül a távolságuktól és méretüktől. Ha egy tárgyat kétszer akkorára növelünk és kétszer olyan messze rakjuk, ugyan olyannak látjuk. Így ha egy síkon nézzük egy mintázatot (például a 4.8. ábrán látható sakktábla mintázatot), akkor elvileg mindegy milyen messze van (lehet az a világ legkisebb kézzel készített sakktáblája, vagy a legnagyobb, földbe szántott sakktábla).

Számoljuk ki tehát a transzformáció mátrix-szát és nézzük meg, hogy a tárgy távolsága mint paraméter tényleg kiesik-e a végső megoldásból.



(a) Azonos látószögek esetén ugyanaz a vetület



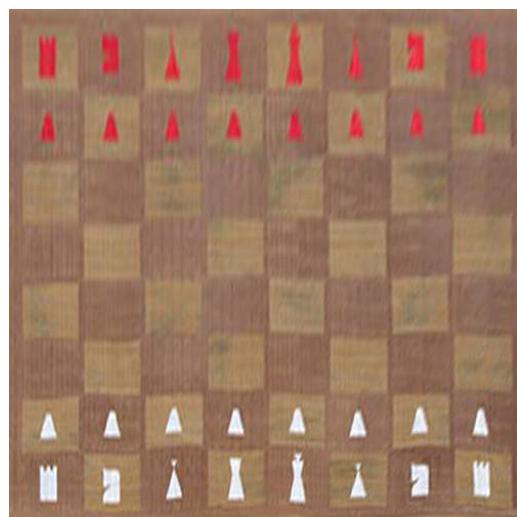
(b) A világ legkisebb (1.2 cm oldalhosszúságú) kézzel készített sakktáblája 2005-ben.



(c) A világ legnagyobb (400 m oldalhosszúságú) sakktáblája 2009-ben.



(d) ... és perspektív rektifikálást követően



(e) ... és perspektív rektifikálást követően

4.8. ábra. Méretarány megkülönböztethetetlenség.

Ha két objektum csak méretarányban különbözik, de – a fenti példákkal kicsit ellentében – azonos beállításban készül róluk a kép (látószögek tekintetében), akkor a transzformációs mátrix a méretaránytól (és így távolságtól) függetlenül ugyanaz lenne.

4.4.1. Probléma megfogalmazása

Mit is szeretnénk valójában?

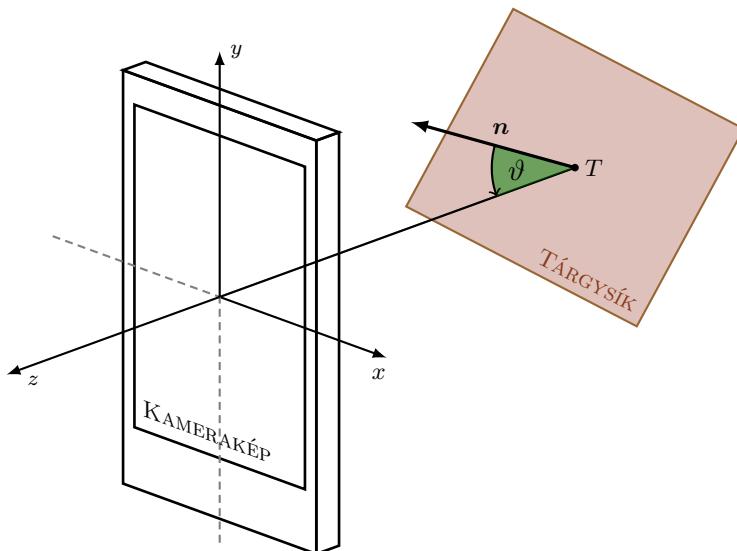
Legyen adott a *tárgysík* (például egy papírlap), amit le szeretnénk fényképezni és transzformálni.

Ezt a síkot szeretnénk „szembefordítani” a kamerával, tehát megkeresni azt a transzformációt, mellyel megszűnik a perspektív nézet. Ekkor a papírlapon lévő kört, négyzetet tényleg körnek, ill. négyzetnek láthatjuk ismét, az eredeti szögeket, arányokat megtartva, mintha csak szkenerrel digitalizáltuk volna.

Valójában persze a teljes, látott környezetet transzformálni fogjuk, mégpedig úgy, mint-ha minden ebben a síkban lenne (a síkból kimagasló tárgyak így például elnyúlnak, ráadásul méretüktől és távolságuktól függően, lásd 4.8b. ábra).

Mi szükséges hozzá?

A pontos számításhoz szükség lenne először is a kamera nézeti irányára és a papírlap (avagy a sík) normálvektorára. Olyan transzformációt – jelenleg egy térbeli forgatást – keresünk, amely a sík n normálvektorát a kamera nézeti irányával ellenkező z irányba fordítja (lásd. 4.9 ábra).



4.9. ábra. Sík szembefordítása a kamerával

Ezt követően pedig majd megvizsgáljuk az eredeti és elforgatott sík vetületei közti kapcsolatot. Nézzük meg, hogy pontosan milyen adatokra is van szükségünk, illetve miket tudunk mérni.

Paraméterek számának csökkentése

Először is, fontos megjegyezni, hogy nem lenne szükségünk abszolút irányokra, csupán a kamera és a sík irányainak eltérése szükséges számunkra. Ha a telefon koordinátarendszerében dolgozunk, akkor a telefon helyzete állandó, ennek megfelelően készül a kép is. Ebben a koordinátarendszerben kellene meghatározni a sík irányát.

Ezt azonban nem tudjuk közvetlen meghatározni a telefon segítségével, hiszen a síkról csupán a kamera által látott képi információ adott. Optikai úton történő felismerés lehetséges lenne ugyan, de csak bizonyos feltételezések mellett. Ha például tudjuk, hogy téglalap alapú lapot fényképezünk, akkor kereshetünk egy négyzetet és ezt transzformáljuk (így működik a meglévő alkalmazások többsége). Ez azonban nem nyújt általános megoldást, mivel a mintázat tetszőleges lehet. Több különböző pozícióban készített kép még talán adhatna lehetőséget a kép helyreállítására, ettől azonban most tekintsünk el.

Lehetséges lenne még a sík felmérésére speciális szenzorok, például távolságmérő használata. Ilyet használnak a Kinect-ben is, azonban telefonokban még nem áll rendelkezésre. A Google épp dolgozik a Project Tango [12] fejlesztésén, ahol több kamerával képzelnék el bemérni a tárgyak távolságát és így valós időben feltérképezni a környezetet. A telefon előlapján ugyan használnak közelségérzékelőt (proximity sensor), de nevéből adódóan ez a legtöbb esetben csak 2 állapotú mérésre alkalmas: csupán azt detektálja, hogy adott távolságon belülre kerül-e valami.

Mivel így a sík irányának közvetlen meghatározására nem igazán van lehetőség, ezért tekintsük a sík irányát rögzítettnek, feltételezzük, hogy „felfele” néz, hiszen általában asztalon fekvő, vízszintes helyzetben szeretnénk papírlapot fényképezni – vagy éppen a földfelszínt, parkettát, útburkolatot ami szintén vízszintes helyzetű. A perspektív rektifikáció ezen formáját nevezik még *madárnézetnek* (birdseye view) is, hiszen így látják a madarak fentről a földfelszínt, de így láthatjuk mi is a szatellit képeket is Google Maps-en, hívják még *ortofotózásnak* (orthophoto) is.

Később még általánosíthatjuk a sík helyzetét, de addig is nézzük meg, miként használható fel ez a feltételezés. Jelenleg ugyanis a telefon nézeti irányát is változatlannak tartjuk a telefon koordinátarendszerében és a sík helyzetét is rögzítettük a földhöz képest, a kettő kapcsolatát továbbra sem határoztuk meg.

A telefonban azonban rendelkezésünkre állhat gyorsulásmérő és iránytű, melyek alapján a telefon nézeti iránya szintén a földi koordinátarendszerben meghatározható, így megkapjuk a kapcsolatot a sík relatív helyzetére a telefon koordinátarendszerében.

Viszont a sík pontos helyzetét még nem határoztuk meg. Csak annyit feltételeztünk, hogy a síkja párhuzamos a föld síkjával, azaz ez a sík normálvektora „felfele” néz, de egy papírlapnál például azt is tudni szeretnénk merre néz a lap „teteje” (felső éle), mert végül a telefonon is ezt szeretnénk felül látni, mintha kezünkbe vettük volna a lapot.

Gondolunk viszont arra, hogy most általános mintázatot készítünk fényképezni, így ha nem papírlapról, hanem síkról beszélünk, akkor annak tulajdonképpen nincs is „iránya”, nyugodtan feltételezhetjük, hogy amilyen irányba nézünk az iránytű alapján, arra van a papírlap teteje is.

Megfigyelhetjük továbbá, hogy így már a telefonnak is lényegtelenné válik az iránytű által meghatározott irányáról beszélni, teljesen mindegy ugyanis, hogy északi irányba vagy déli irányba fotózunk egy papírlapot, ugyanazt a képet szeretnénk visszakapni.

A készülék nézeti irányának meghatározásához a gyorsulásmérőre és a magnetométere lenne szükség, viszont utóbbi ezek szerint nélkülözhető, az északi pólus tényleges helyzete teljesen irreleváns számunkra, elegendő csupán a *billentő* (*pitch*) és *csavaró* (*roll*) szögek

ismerete, melyeket a gyorsulásmérő képes mérni.

Nyugalmi állapotban a gravitációs gyorsulás „felfele”, azaz a föld középpontja felől mutat, ami földi koordinátarendszerben $(0,0,g)$, ezt a vektort méri az eszköz saját koordinátarendszerében.

Szintén „felfele”, tehát föld középpontjával ellentétes irányba néz a sík normálvektora, ennek nagysága tetszőleges, tehát gyakorlatilag azt is mondhatjuk, hogy a készülék gyorsulásmérője pontosan a sík normálvektorát méri.

Szerencsés, nagyon egyszerű helyzetben vagyunk tehát, nincs szükség semmilyen koordinátarendszer átváltásra, még koordinátatengely-cserére, előjel váltásra sem. Az API felől érkező adatok pontosan a számunkra szükséges \mathbf{n} normálvektor koordinátáit fejezik ki.

4.4.2. Forgatás (rotáció)

Egy olyan forgatást keresünk tehát a térben, ami a mért \mathbf{n} normálvektort a 4.9. ábrán látható módon egy bizonyos ϑ (ejtsd: théta) szöggel a z tengely irányába fordítja. Ezt a 4.10. ábrasorozaton láthatóan 2 (Pontosabban 3) egymás utáni, tengelyek mentén történő forgatással lehet szemléletesen előállítani.

Rajzoljuk be a 4.10a. ábrán látható módon az $\mathbf{n} = (u, v, w)$ normálvektort a telefon koordinátarendszerébe és jelöljük végpontját N -nel, az $x-y$ síkra eső vetületét pedig N_T -vel (mint talppont). Az \mathbf{n} vektor $\overrightarrow{ON_T}$ vetületét pedig jelöljük \mathbf{m} -mel.

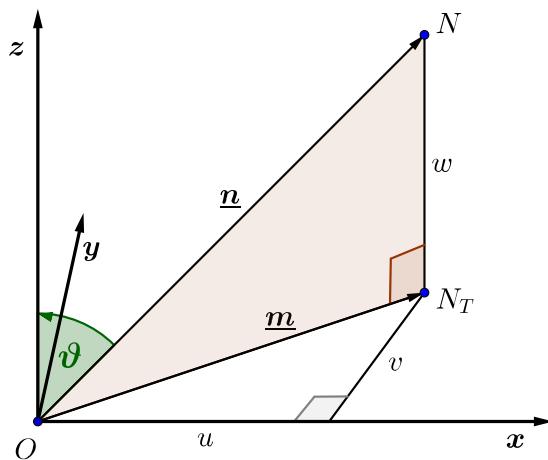
Az egyértelműség és egyszerűség kedvéért minden forgatást előjelesen, *jobbkéz-szabály* szerint értsünk (ld. D.1. függelék).

Mivel a forgatás az $\mathbf{m} - z$ síkban történik, ezért először a 4.10b. és a 4.10c. ábrán látható módon a z tengely körül forgassuk el a koordináta-rendszert egy olyan φ (ejtsd: fí) *precessziós* szöggel, hogy az x tengely képe az \mathbf{m} vektorral kerüljön egy irányba. Jelöljük ennek a forgatásnak, mint transzformációinak mátrixát \mathbf{R}_z^φ -vel.

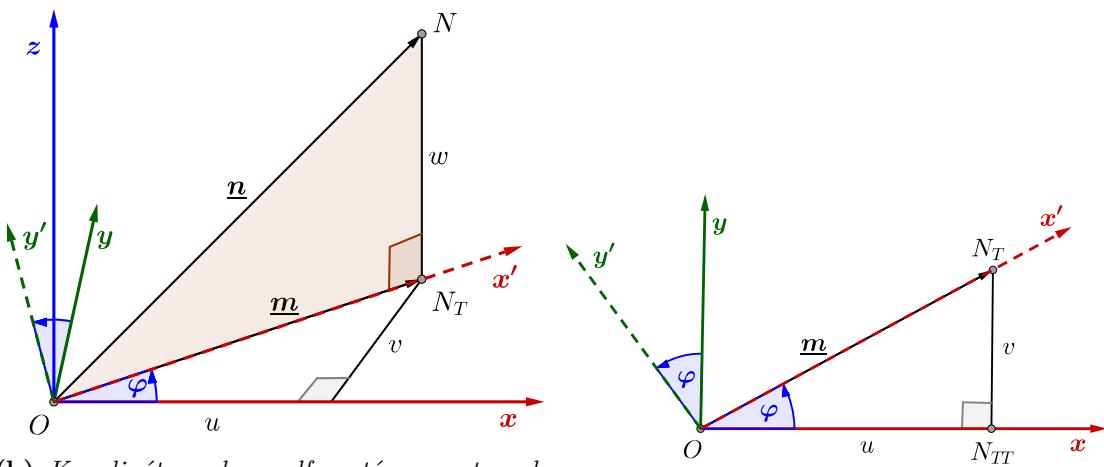
Az elforgatott koordináta-rendszerben a 4.10d. és a 4.10e. ábrán láthatóan az új y' tengely körül forgassunk olyan $-\vartheta$ *nutációs* szöggel, hogy az \mathbf{n} vektor képe a z tengellyel legyen egyirányú.

Az \mathbf{R}_z^φ -vel elforgatott koordináta-rendszerben felírt $\mathbf{R}_y^{-\vartheta}$ transzformációt az eredeti koordináta-rendszerben felírva a D.2.3. függelékben $\mathbf{R}_z^\varphi \{\mathbf{R}_y^{-\vartheta}\}$ -vel jelölve a következő módon írhatjuk fel:

$$\mathbf{R} = \mathbf{R}_z^\varphi \{\mathbf{R}_y^{-\vartheta}\} = \mathbf{R}_z^\varphi \cdot \mathbf{R}_y^{-\vartheta} \cdot \mathbf{R}_z^{-\varphi} \quad (4.1)$$

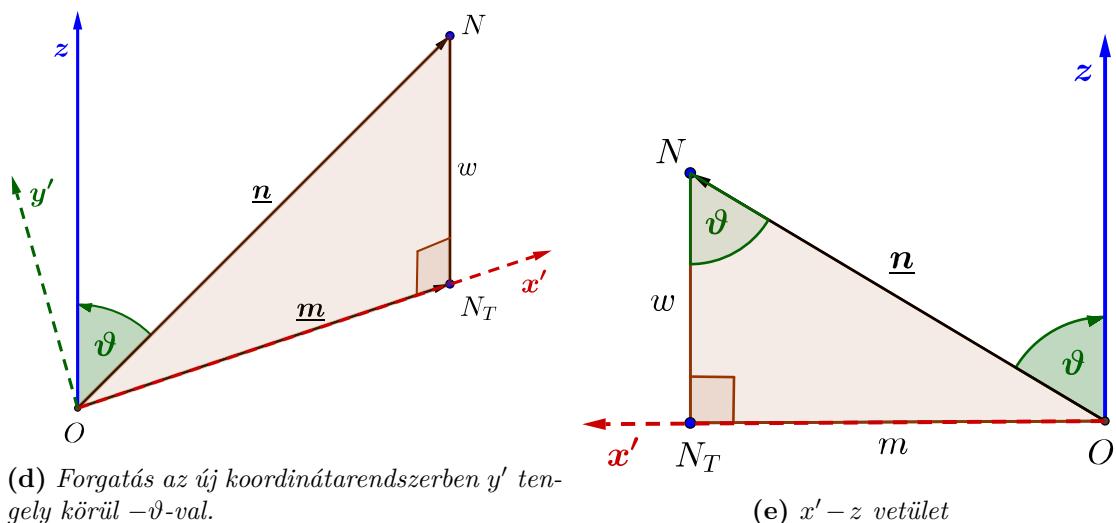


(a) A szükséges forgatás, ami a normálvektort a z tengelyre fordítja



(b) Koordinátarendszer elforgatása a z tengely körül $+\varphi$ -vel.

(c) $x-y$ vetület



(d) Forgatás az új koordinátarendszerben y' tengely körül $-\vartheta$ -val.

(e) $x'-z$ vetület

4.10. ábra. A szükséges forgatás, ami a normálvektort a z tengelyre fordítja és annak előállítása

A z tengely körüli forgatás

Egy, a z tengely körüli φ szöggel forgatás mátrixa:

$$\mathbf{R}_z^\varphi = \begin{pmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4.2)$$

Mivel az \mathbf{n} vektor koordinátánként adott (u, v, w) , ezért a φ szögfüggvényét ezekkel kifejezve:

$$\begin{aligned} \sin \varphi &= \frac{v}{m} \\ \cos \varphi &= \frac{u}{m} \end{aligned}$$

Visszahelyettesítve a (4.2) mátrixba és kiemelve $\frac{1}{m}$ -et:

$$\mathbf{R}_z^\varphi = \frac{1}{m} \begin{pmatrix} u & -v & 0 \\ v & u & 0 \\ 0 & 0 & m \end{pmatrix} \quad (4.3)$$

Az y tengely körüli forgatás

Egy, az y tengely körül ϑ szöggel forgatás mátrixa a következő lenne:

$$\mathbf{R}_y^\vartheta = \begin{pmatrix} \cos \vartheta & 0 & \sin \vartheta \\ 0 & 1 & 0 \\ -\sin \vartheta & 0 & \cos \vartheta \end{pmatrix}$$

Esetünkben ($\vartheta = -\vartheta$, így $\cos \vartheta = \cos \vartheta$ és $\sin \vartheta = -\sin \vartheta$):

$$\mathbf{R}_y^{-\vartheta} = \begin{pmatrix} \cos \vartheta & 0 & -\sin \vartheta \\ 0 & 1 & 0 \\ \sin \vartheta & 0 & \cos \vartheta \end{pmatrix} \quad (4.4)$$

A ϑ szög szögfüggvényeit a 4.10d. ábrán látható piros háromszögében található $\widehat{ONN_T}$ váltószöge segítségével a következőképp írhatjuk fel:

$$\begin{aligned} \sin \vartheta &= \frac{m}{n} \\ \cos \vartheta &= \frac{w}{n} \end{aligned}$$

Amit a mátrixba visszahelyettesítve és $\frac{1}{n}$ -et kiemelve:

$$\mathbf{R}_y^\vartheta = \frac{1}{n} \begin{pmatrix} w & 0 & -m \\ 0 & n & 0 \\ m & 0 & w \end{pmatrix} \quad (4.5)$$

A térbeli forgatás

A kapott tengelyenkénti forgatásokat a 4.1. egyenletbe visszahelyettesítve

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_z^\varphi \cdot \mathbf{R}_y^{-\vartheta} \cdot \mathbf{R}_z^{-\varphi} \\ &= \frac{1}{m} \begin{pmatrix} u & -v & 0 \\ v & u & 0 \\ 0 & 0 & r \end{pmatrix} \cdot \frac{1}{n} \begin{pmatrix} w & 0 & -m \\ 0 & n & 0 \\ m & 0 & w \end{pmatrix} \cdot \frac{1}{m} \begin{pmatrix} u & v & 0 \\ -v & u & 0 \\ 0 & 0 & r \end{pmatrix} \\ &= \frac{1}{n \cdot m^2} \begin{pmatrix} nv^2 + u^2w & uvw - nuv & -m^2u \\ uvw - nuv & nu^2 + v^2w & -m^2v \\ m^2u & m^2v & m^2w \end{pmatrix},\end{aligned}$$

mely homogén koordinátákkal (ld. D.3. függelék) kifejezve:

$$\boxed{\mathbf{R} = \begin{pmatrix} nv^2 + u^2w & uvw - nuv & -m^2u & 0 \\ uvw - nuv & nu^2 + v^2w & -m^2v & 0 \\ m^2u & m^2v & m^2w & 0 \\ 0 & 0 & 0 & n \cdot m^2 \end{pmatrix}} \quad (4.6)$$

4.4.3. Eltolás (transzláció)

Az iménti forgatás azonban a definiált koordináta-rendszerünkben lévő $O(0,0,0)$ pont (telefon origója) körül történne, mi viszont a kinézett t tárgytávolságra lévő $T(0,0,-t)$ tárgypont körül szeretnénk elvégezni, így ismét egy koordináta-rendszer transzformációra (ld. D.2.3. függelék) – egy T -vel jelölt eltolásra (translation) – van szükségünk. Az így kapott eltolásos forgatás transzformációja:

$$\mathbf{TransRot} = \mathbf{T}\{\mathbf{R}\} \stackrel{(D.14)}{=} \mathbf{T} \cdot \mathbf{R} \cdot \mathbf{T}^{-1} \quad (4.7)$$

Viszont az eltolás nem lineáris transzformáció, ezért nem írható fel 3×3 -as mátrix segítségével. Szerencsére könnyen megoldható a probléma egy 4×4 -es mátrix segítségével, ha ún. *homogén koordinátákra* váltunk (ld. D.3. függelék).

Homogén koordinátákkal egy z irányú t -vel eltolás mátrixa:

$$\mathbf{T}_t := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.8)$$

Esetünkben $\mathbf{T} = \mathbf{T}_{-t}$, valamint tudjuk, hogy $\mathbf{T}_t^{-1} = \mathbf{T}_{-t}$, így az eltolásos forgatás egy

általános forgatásra felírva:

$$\begin{aligned}
 \mathbf{TransRot}_{(gen)} &\stackrel{(4.7)}{=} \mathbf{T}_{-t} \cdot \mathbf{R}_{(gen)} \cdot \mathbf{T} \\
 &\stackrel{(4.8)}{=} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -t \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & r_{44} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{13} \cdot t \\ r_{21} & r_{22} & r_{23} & r_{23} \cdot t \\ r_{31} & r_{32} & r_{33} & r_{33} \cdot t - t \cdot r_{44} \\ 0 & 0 & 0 & r_{44} \end{pmatrix}, \tag{4.9}
 \end{aligned}$$

ami esetünkben (ld. 4.6):

$$\boxed{\mathbf{TransRot} = \begin{pmatrix} nv^2 + u^2w & -nuv + uvw & -m^2u & -m^2u \cdot t \\ -nuv + uvw & nu^2 + v^2w & -m^2v & -m^2v \cdot t \\ m^2u & m^2v & m^2w & m^2w \cdot t - n \cdot m^2 \cdot t \\ 0 & 0 & 0 & n \cdot m^2 \end{pmatrix}} \tag{4.10}$$

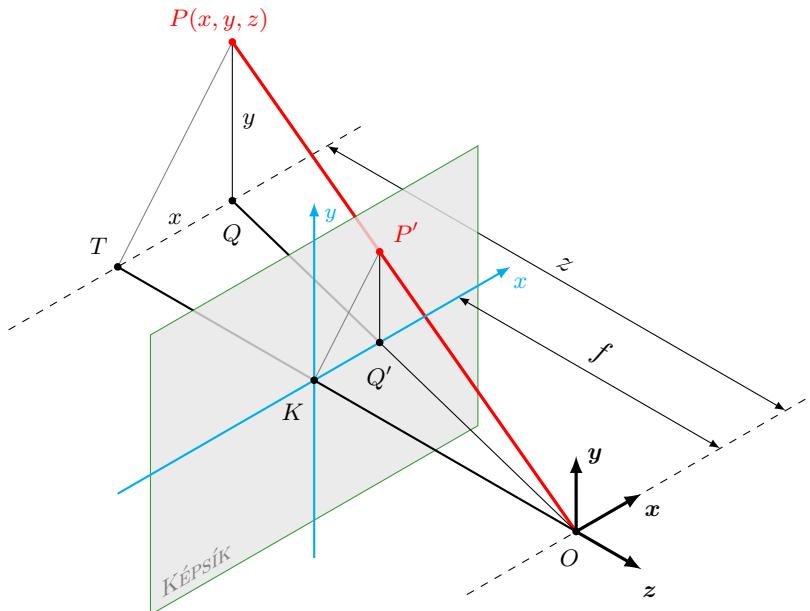
4.4.4. Középpontos vetítés (projekció)

Mivel a térbeli pontok helyett mi csak a kamera érzékelőjére eső vetületet vagyunk képesek érzékelni, ezért szükség van az ezt leíró transzformáció vizsgálatára is.

A kamerák képalkotása során keletkezett kép valójában egy origó középpontú középpontos vetítés hatására történik. Egy adott pont képét úgy kaphatjuk meg, hogy kiszámítjuk, hogy hol metszi a képsíkot az origóval összekötött egyenes. Ezt matematikailag hasonló háromszögek keresésével írhatjuk fel (ld. 4.11. ábra).

$$\frac{x'}{x} = \frac{-f}{z}, \tag{4.11}$$

$$\frac{y'}{y} = \frac{-f}{z}. \tag{4.12}$$



4.11. ábra. Középpontos vetítés

Egy általános (x, y, z) pont képe a $\left(-\frac{fx}{z}, -\frac{fy}{z}, -f\right)$ lesz.

Mivel ez újfent nem lineáris transzformáció lenne, ezért homogén koordinátákat (ld. D.3. függelék) alkalmazunk.

Egy általános $(x, y, z, 1)$ pont képe a $\left(-\frac{fx}{z}, -\frac{fy}{z}, -f, 1\right)$ kell, hogy legyen, viszont nekünk az ezzel ekvivalens $(-fx, -fy, -fz, z)$ sugár is megfelelő, amire így már könnyedén található mátrix:

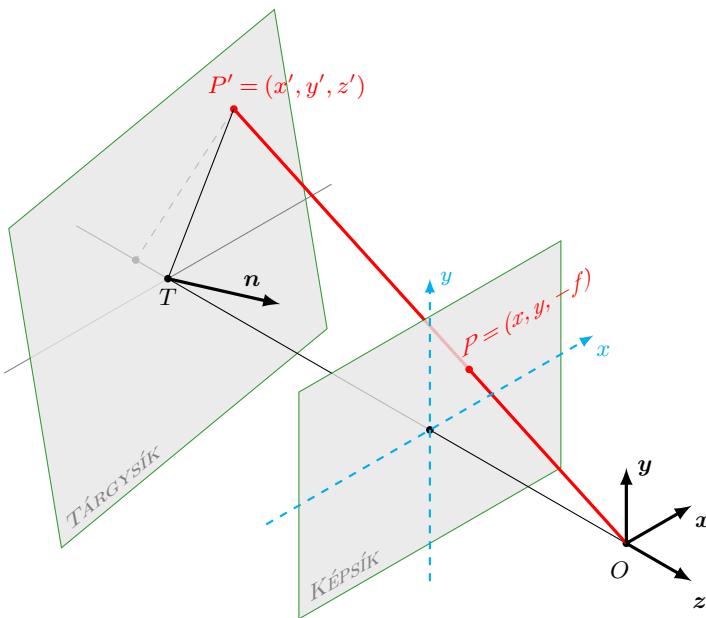
$$\mathbf{P} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (4.13)$$

Ezzel viszont az a baj, hogy *szinguláris* (azaz 0 determinánsú, hiszen utolsó oszlopa csupa 0-ból áll), tehát nincs inverze. Ez persze várható is volt: ugyanazon vetületi ponthoz egy egész egyenesnyi pont rendelődik.

De nekünk nem akármilyen inverze kell, hanem az, amelyik a t tárgytávolságú (u, v, w) normálvektorú síakra vetíti vissza a pontokat, hiszen azt feltételezzük, hogy minden látott pont erre illeszkedik (ld. 4.12.ábra).

Az O -ból P -n átmenő egyenes pontjai $(\lambda \cdot x, \lambda \cdot y, -\lambda \cdot f)$, ahol λ az ismeretlen.

Viszont tudjuk, hogy a T -vel összekötve a síakra illeszkedik, tehát az \mathbf{n} normálvektorra merőleges. Ami viszont ekvivalens azzal, hogy skalár szorzatuk 0.



4.12. ábra. Projekció inverze

$$\begin{aligned}
 \overrightarrow{TP} \perp \mathbf{n} & \quad / \text{ közrezárt szögük } 90^\circ, \text{ aminek koszinusza } 0 \\
 \overrightarrow{TP} \cdot \mathbf{n} = 0 & \quad / \text{ helyvektorokkal kifejezve} \\
 (\mathbf{P} - \mathbf{T}) \cdot \mathbf{n} = 0 & \quad / \text{ koordináták behelyettesítése} \\
 ((\lambda x, \lambda y, -\lambda f) - (0, 0, -t)) \cdot (u, v, w) = 0 & \quad / \text{ összevonás} \\
 (\lambda x, \lambda y, t - \lambda f) \cdot (u, v, w) = 0 & \quad / \text{ vektoriális szorzat kifejtése} \\
 u\lambda x + v\lambda y + w(t - \lambda f) = 0 & \quad / \lambda \text{ szerint csoportosítás} \\
 \lambda(ux + vy - wf) = -wt & \quad / \lambda \text{ kifejezése} \\
 \lambda = \frac{-wt}{ux + vy - wf}
 \end{aligned}$$

Tehát egy képsíkon lévő $(x, y, -f)$ pont képe a tárgysíkon

$$\begin{pmatrix} -\frac{wtx}{ux+vy+wz} \\ -\frac{wty}{ux+vy+wz} \\ \frac{wtf}{ux+vy+wz} \end{pmatrix} \quad (4.14)$$

Ez a transzformáció a következő mátrix-szal fejezhető ki:

$$\mathbf{P}_{inv} = \begin{pmatrix} -wt & 0 & 0 & 0 \\ 0 & -wt & 0 & 0 \\ 0 & 0 & -wt & 0 \\ u & v & w & 0 \end{pmatrix} \quad (4.15)$$

4.4.5. Az eredő síktranszformáció

A kapott eredményeket összegezve tehát adott számunkra egy kamera által látott kép, visszatranszformáljuk a tárgysíkra, eltoljuk a kamera origójába, szembefordítjuk a kamerával, visszatoljuk az eredeti helyére és ismét rávetítjük a kamera képsíkjára, hogy mit látnánk, ha így szemből állna a kép. Az ezt leíró transzformáció:

$$ProjTransRot = P\{T\{R\}\} = P \cdot \underbrace{T \cdot R \cdot T^{-1}}_{TransRot} \cdot P_{inv} \quad (4.16)$$

Ha R helyére behelyettesítjük egy általános forgatás mátrixát:

$$\begin{aligned} ProjTransRot_{(gen)} &= P \cdot TransRot_{(gen)} \cdot P_{inv} = \\ &= \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{13} \cdot t \\ r_{21} & r_{22} & r_{23} & r_{23} \cdot t \\ r_{31} & r_{32} & r_{33} & r_{33} \cdot t - t \cdot r_{44} \\ 0 & 0 & 0 & r_{44} \end{pmatrix} \cdot \begin{pmatrix} -wt & 0 & 0 & 0 \\ 0 & -wt & 0 & 0 \\ 0 & 0 & -wt & 0 \\ u & v & w & 0 \end{pmatrix} \\ &= \begin{pmatrix} f(w \cdot r_{11} - u \cdot r_{13}) & f(w \cdot r_{12} - v \cdot r_{13}) & 0 & 0 \\ f(w \cdot r_{21} - u \cdot r_{23}) & f(w \cdot r_{22} - v \cdot r_{23}) & 0 & 0 \\ f(w \cdot r_{31} - u \cdot r_{33} + u \cdot r_{44}) & f(w \cdot r_{32} - v \cdot r_{33} + v \cdot r_{44}) & f \cdot w \cdot r_{44} & 0 \\ -w \cdot r_{31} + u \cdot r_{33} - u \cdot r_{44} & -w \cdot r_{32} + v \cdot r_{33} - v \cdot r_{44} & -w \cdot r_{44} & 0 \end{pmatrix} \quad (4.17) \end{aligned}$$

Ez azonban még egy 4×4 -es mátrix, nekünk pedig 2 dimenziós a látott képünk. Nézzük meg mi lesz tehát egy képsíkon lévő $P(x, y)$ pontból. Mivel a képsík f távolságra van, ezért a három dimenziós koordinátái (x, y, f) . Ezt egészítjük ki egy koordinátával, hogy homogén koordinátaként kezelhessük: $P(x, y, z, -f, 1)$. Erre alkalmazzuk a (4.17) transzformációt:

$$\begin{aligned} ProjTransRot_{(gen)} \cdot (x, y, z, -f, 1) &= \\ &= \begin{pmatrix} f(r_{11}w - r_{13}u)x + f(r_{12}w - r_{13}v)y \\ f(r_{21}w - r_{23}u)x + f(r_{22}w - r_{23}v)y \\ f(r_{31}w - r_{33}u + r_{44}u)x + f(r_{32}w - r_{33}v + r_{44}v)y - f(f \cdot r_{44}w) \\ -(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w) \end{pmatrix} \quad (4.18) \end{aligned}$$

Ha normáljuk a kapott sugarat a negyedik koordinátájával, a következő vektort kapjuk:

$$\begin{pmatrix} \frac{f(r_{11}w - r_{13}u)x + f(r_{12}w - r_{13}v)y}{-(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w)} \\ \frac{f(r_{21}w - r_{23}u)x + f(r_{22}w - r_{23}v)y}{-(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w)} \\ -f \\ 1 \end{pmatrix} \quad (4.19)$$

Ahol szerencsére a 3. koordináta valóban $-f$ lett, ahogy elvártuk. Az első két koordináta

pedig pont a transzformált pontot írja le:

$$P' = \begin{pmatrix} f(r_{11}w - r_{13}u)x + f(r_{12}w - r_{13}v)y \\ -(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w) \\ f(r_{21}w - r_{23}u)x + f(r_{22}w - r_{23}v)y \\ -(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w) \end{pmatrix} \quad (4.20)$$

Próbáljuk meg előállítani mátrixos formában. Ehhez a közös nevezőket homogén 3. koordinátaként használjuk fel, az így kapott 3 koordinátát pedig gyűjtsük össze x és y szerint:

$$P'_H = \begin{pmatrix} f(r_{11}w - r_{13}u)x + f(r_{12}w - r_{13}v)y \\ f(r_{21}w - r_{23}u)x + f(r_{22}w - r_{23}v)y \\ -(r_{31}w - r_{33}u + r_{44}u)x - (r_{32}w - r_{33}v + r_{44}v)y + (f \cdot r_{44}w) \end{pmatrix} \quad (4.21)$$

Ebből már könnyedén felírható az ezt előállító mátrix:

$$\text{PlaneRotate} = \begin{pmatrix} f(w \cdot r_{11} - u \cdot r_{13}) & f(w \cdot r_{12} - v \cdot r_{13}) & 0 \\ f(w \cdot r_{21} - u \cdot r_{23}) & f(w \cdot r_{22} - v \cdot r_{23}) & 0 \\ -w \cdot r_{31} + u \cdot r_{33} - u \cdot r_{44} & -w \cdot r_{32} + v \cdot r_{33} - v \cdot r_{44} & f \cdot w \cdot r_{44} \end{pmatrix} \quad (4.22)$$

Fotóforgatási transzformáció (Rotated Photo Transfromation)

Legyen egy kamera koordináta-rendszerében egy t távolságra látott T tárgypont egy bizonyos síkjának normálvektora $\mathbf{n} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$, valamint a kamera fókusztávolsága

f . Ha a síkot elforgatjuk a T pont körül egy $R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & r_{44} \end{pmatrix}$ 3D-s (homogén koordinátás alakban felírt) forgatásmátrix-szal, akkor az így látott 2D-s kép és eredeti 2D-s kép közötti transzformációt az alábbi homogén koordinátás mátrix írja le:

$$\mathcal{RPT} = \begin{pmatrix} f(w \cdot r_{11} - u \cdot r_{13}) & f(w \cdot r_{12} - v \cdot r_{13}) & 0 \\ f(w \cdot r_{21} - u \cdot r_{23}) & f(w \cdot r_{22} - v \cdot r_{23}) & 0 \\ -w \cdot r_{31} + u \cdot r_{33} - u \cdot r_{44} & -w \cdot r_{32} + v \cdot r_{33} - v \cdot r_{44} & f \cdot w \cdot r_{44} \end{pmatrix} \quad (4.23)$$

az $n = \sqrt{u^2 + v^2 + w^2}$, valamint $m = \sqrt{u^2 + v^2}$ rövidítésekkel élve.

Helyettesítsük be a valódi, merőleges nézebe forgató (4.6) mátrixot:

$$\begin{pmatrix} f(w \cdot (nv^2 + u^2w) - u \cdot (-m^2u)) & f(w \cdot (uvw - nuv) - v \cdot (-m^2u)) & 0 \\ f(w \cdot (uvw - nuv) - u \cdot (-m^2v)) & f(w \cdot (nu^2 + v^2w) - v \cdot (-m^2v)) & 0 \\ -w \cdot (m^2u) + u \cdot (m^2w) - u \cdot (n \cdot m^2) & -w \cdot (m^2v) + v \cdot (m^2w) - v \cdot (n \cdot m^2) & f \cdot w \cdot (n \cdot m^2) \end{pmatrix} \quad (4.24)$$

Ha elvégezzük a szorzásokat, visszahelyettesítjük $m^2 = u^2 + v^2$ és $n^2 = u^2 + v^2 + w^2$ helyettesítéseket, majd újra egyszerűsítjük a kifejezést és leosztunk f -vel, a 4.25-os alakra hoz-

ható. Valamint egyszerű mátrix-szorzással ellenőrizhető, hogy ennek inverze a 4.26 mátrix (ugyan nem egységmátrix az eredmény, de mivel homogén mátrix, ezért $n \cdot w \cdot m^4$ -gyel egyszerűsíthetünk).

A kapott eredmény tehát:

Ortofotó transzformáció (OrthoPhoto Transformation)

Legyen egy kamera koordináta-rendszerében egy t távolságra látott T tárgypont egy bizonyos síkjának normálvektora $\mathbf{n} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$, valamint a kamera fókusztávolsága f . Ha a síkot elforgatjuk a T pont körül úgy hogy a normálvektor a kamera $-z$ irányába nézzen, akkor az így látott 2D-s kép és eredeti 2D-s kép közötti transzformációt az alábbi homogén koordinátás mátrix írja le:

$$\mathcal{OPT} = \begin{pmatrix} u^2n + v^2w & u \cdot v(n-w) & 0 \\ u \cdot v(n-w) & u^2w + v^2n & 0 \\ -\frac{u \cdot m^2}{f} & -\frac{v \cdot m^2}{f} & w \cdot m^2 \end{pmatrix} \quad (4.25)$$

az $n = \sqrt{u^2 + v^2 + w^2}$, valamint $m = \sqrt{u^2 + v^2}$ rövidítésekkel élve.

Ennek inverze pedig a következő mátrix:

$$\mathcal{IOP}T = \begin{pmatrix} u^2w + v^2n & u \cdot v(n-w) & 0 \\ u \cdot v(n-w) & u^2n + v^2w & 0 \\ \frac{u \cdot m^2}{f} & \frac{v \cdot m^2}{f} & n \cdot m^2 \end{pmatrix} \quad (4.26)$$

4.5. Smart Zoom

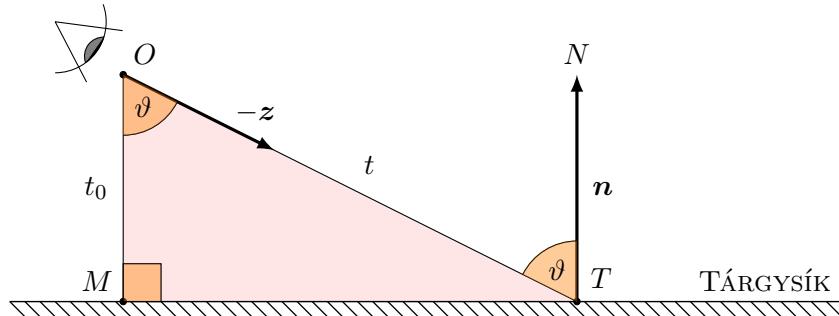
A 4.4. fejezetben kiszámolt mátrix segítségével már könnyedén készíthetünk a síkról úgy képet, mintha felülről készítettük volna. Időnként szükséges lehet ránagyítani egy kívánt részletre, mint ezt általában megszoktuk. Azonban ha folyamatosan pásztázzuk a síkot, szembesülünk azzal az amúgy megszokott akadálytal, hogy a különböző távolságokra lévő dolgok más más méretűnek látszódnak, a távolabbi helyekre jobban rá kell nagyítani („zoom”-olni), mint a közelebbi tárgyakra, ez a folyamatos változatás pedig zavaró lehet.

Szerencsére egy nagyon egyszerű, frappáns trükkkel még könnyebbé tehetjük a sík pásztázását. Azt ugyan nem tudjuk, hogy pontosan milyen messze van egy kinézett pont, de egy adott zoom beállítás mégis megőrizhető. A kamera középpontos vetítésen alapuló működési elvából adódóan a hasonló háromszögekre vonatkozó tulajdonságból következően a fordított arányosság lineáris: egy k -szor messzebbi tárgy k -ad akkorának fog tűnni, így k -szoros nagyítás szükséges.

A 4.13. ábrán látható módon jelöljük t_0 -val a sík távolságát (a legközelebbi M merőleges

talpponttól vett távolság). Ha ehhez viszonyítjuk a nagyítás mértékét, akkor a sík éppen szemlélt T tárgypontjának távolságát t -vel jelölve a szükséges nagyítás mértéke:

$$\text{RelZoom} = \frac{t}{t_0} \quad (4.27)$$



4.13. ábra. Relatív távolság számítása

Ugyan se a kinézett pont t távolságát nem tudjuk, se a sík t_0 távolságát, a kettő aránya mégis kifejezhető az OMT_{Δ} derékszögű háromszög MOT_{\triangleleft} szögének szögfüggvényével, ugyanis

$$\cos \vartheta = \frac{t_0}{t} \quad (4.28)$$

Az MOT_{\triangleleft} pedig az OTN_{\triangleleft} -el váltószögpárt alkot, így éppen megegyezik a 4. fejezetben ϑ -val elnevezett szöggel.

$$\cos \vartheta = \frac{z}{n} \quad (4.29)$$

Így a relatív zoom

$$\text{RelZoom} = \frac{n}{w} \quad (4.30)$$

Összefoglalva tehát:

Intelligens nagyítás (Smart Zoom)

Ha a kamera (egy adott nézeti irányához tartozó) koordináta-rendszerében egy sík normálvektora pillanatnyilag $\mathbf{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$, akkor a

$$\text{RelZoom} = \frac{\|\mathbf{n}\|}{n_z} = \frac{\sqrt{n_x^2 + n_y^2 + n_z^2}}{n_z} \quad (4.31)$$

nagyítás hatására a síkon látott tárgy épp akkorának tűnik, mint amekkorának látnánk, ha a sík kamerához legközelebbi pontjában lenne.

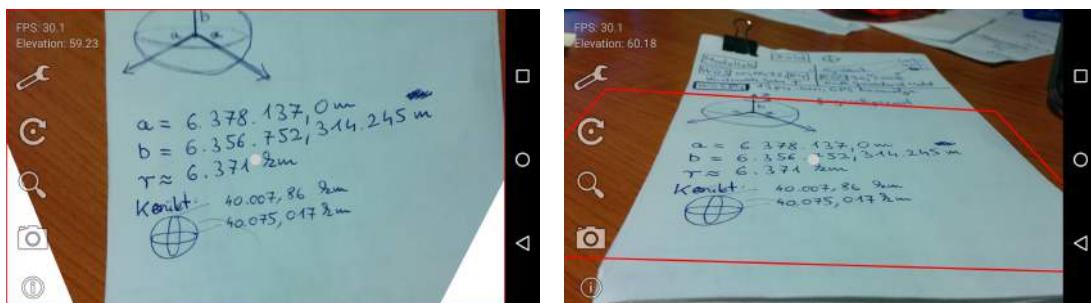
4.6. Bemutatás



4.14. ábra. OrthoCam alkalmazás indító ikonja

Az *OrthoCam* alkalmazás indítását követően megjelenik egy kamera előnézeti kép és bal oldalon egy menüsor (4.15a), melynek tetején debug jelleggel látható az FPS és emelkedési szög. A menüsorból fentről lefele haladva az alábbi lehetőségek állnak rendelkezésre:

- **Beállítások** – FPS és emelkedési szögek mutatásának bekapcsolása, elrejtése. Különböző típusú szenzorok közül lehet választani (Rotation vector, Gravity sensor, Accelerometer). Illetve a kamerakép felbontását állíthatjuk még itt.
- **Szembeforgatás** – Ennek hatására a képnézeti mód merőleges (felülnézeti) nézetbe fordul.
- **Smart zoom** – távolság kompenzációja a használható segédeljárás.
- **Fényképezés** – Fényképet készít az SD kártyára.
- **Infók** – Debug jelleggel itt olvasható minden szükséges információ a kamerá(k)ról, azok támogatott képességeiről.



(a) Előnézeti kép és a menüsor

(b) Szembeforgatás után

4.15. ábra. OrthoCam képernyőképek

4.7. Alkalmazási lehetőségek

A módszer több területen is alkalmazható lehet, amik már említve is voltak:

- Felhasználóknak ortofotózásra
- Drónoknak ortofotózásra
- Szkennelő programok
- Képfelismerések előfeldolgozási lépéseként

4.8. Továbbfejlesztési lehetőségek

Kompatibilitás

Jelenleg a gyors „proof-of-concept” jellegű fejlesztés miatt a kényelmesen használható `TextureView` használata mellett döntöttem, így csak 4.0-ás verziótól támogatott a program. Így, hogy a megvalósíthatósági próba sikeres eredménnyel zárult, alkalom nyílik a korábbi verziókra történő felkészítésre, vagy igény szerint más platformokra fejlesztésre is.

Kamera kalibráció

A transzformáció csak a perspektív torzulást állítja helyre, egy idealizált esetben, pinhole-kamerát feltételezve. Mivel a fényképezőgépek leképezése során felléphetnek bizonyos torzulások, ezekkel is számolni kell. Ilyenek a hordó (barrel), ill. párnás (pillow) torzítások. Az OpenCV toolkit például lehetőséget nyújt a kamera kalibrációra. Ezt a feldolgozás első lépéseként lehetne végrehajtani, amivel így pontosabb síkleképzést kapnánk.

Sík kalibráció

Jelenleg a program vízszintes síkfelületet feltételez. Ám ha a magnetométert is felhasználjuk szenzorként, általános irányokat is lehetne mérni. Egy sík helyzetét pedig fel lehetne mérni úgy, hogy a telefont hátlapjával (vagy előlapjával) ráhelyezzük.

Viszont a telefon hátlapja sem teljesen sík, a Nexus 5 készüléken például egy fél mm-re kiáll a kamera lencséje, ami miatt asztalra téve nem vízszintes helyzetet mér. Így a síkbemérés helyes használatához ezt is érdemes lenne kalibrálni egy teljesen sík felületre helyezéssel.

5. fejezet

Összefoglalás

Egyetemi tanulmányaim zárásaként diplomamunkám során véghezvittem egy több területet átfogó kutatás-fejlesztési feladatot. Megismerkedtem a kiterjesztett valóság témaakörével, újszerű alkalmazási módokat kerestem és implementáltam. Terveimet, elemzéseimet, eredményeimet dokumentáltam.

Először is megismertem napjaink mobil készülékeit, operációs rendszereit, melyek közül elmelélyült az *Android* platform fejlesztésében.

Megismertem a *kamera* és *szenzorok* API szintű kezelésével. Egy saját segédkönyvtárat írtam, melybe különböző AR-hez hasznos segédeljárásokat tettem.

Néhány gyanús viselkedés esetén egészen az Android forráskódjáig eljutottam, sőt meg is találtam a hibás részeket. Ezekről hibajelentést küldtem, majd *patch*-et készítettem a helyes működéshez, melyekből az egyiket már sikeresen elfogadták. Ezzel hozzájárultam az Android nyílt fejlesztéséhez. Kipróbáltam néhány keretrendszer, sokat teszteltem az egyes megoldásokat.

Véghezvittem egy hosszabb és több rövidebb matematikai számítást lineáris algebra segítségével. Eközben a *Mathematica* program kezelését is elsajátítottam. Megismerkedtem a 3D-s transzformációkkal, forgatásokkal, Euler-szögekkel, kvaternióval. Számos képalkotási és képfeldolgozási ismeretet és módszert szereztem.

A programozás során az *Eclipse* IDE használatában szereztem új tapasztalatokat. Nagyrészt *Java* kódokat írtam, de megismerkedtem az *NDK*-val, *JNI*-vel is, melyek során C++-ban is kódoltam.

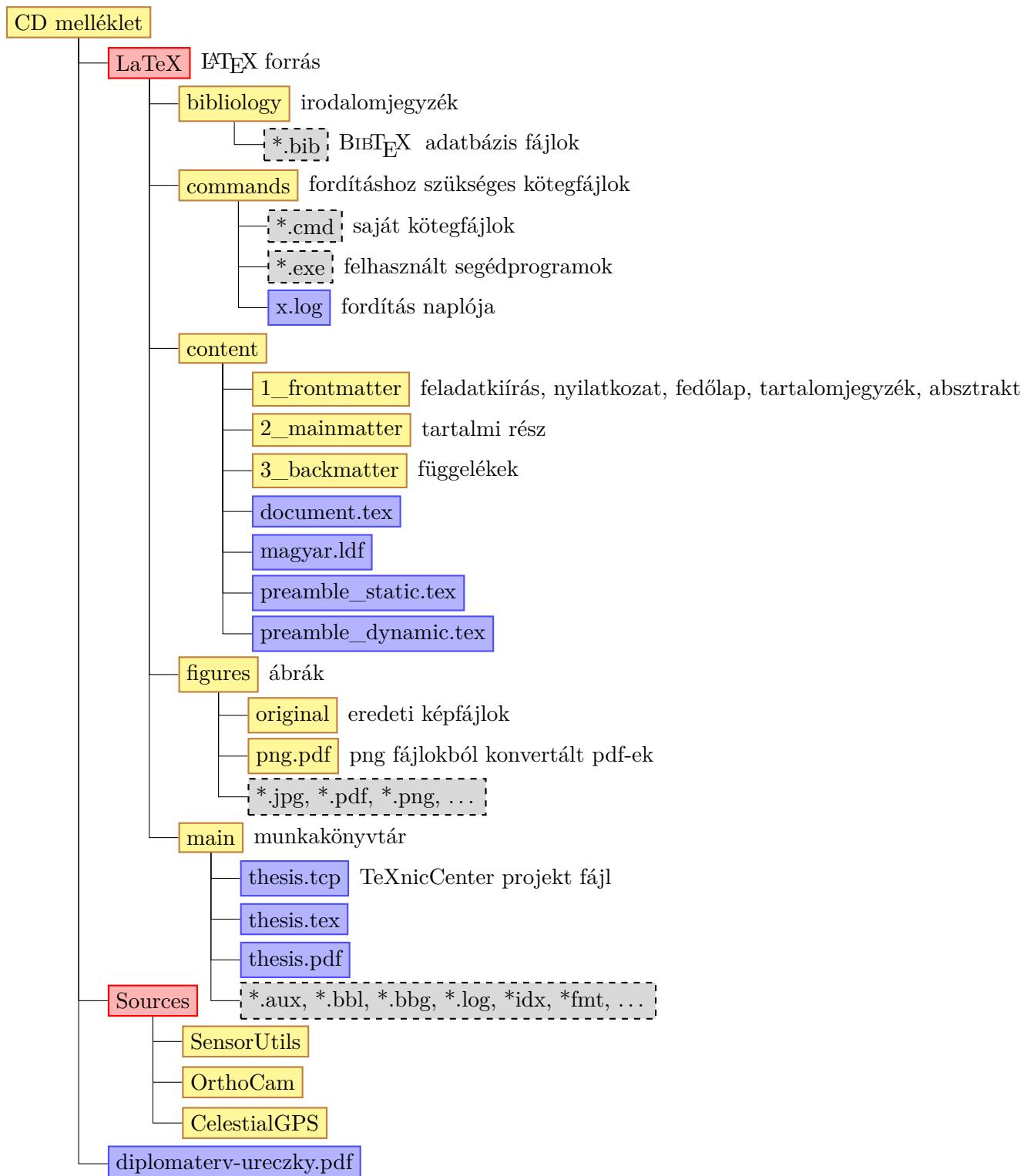
Megismerkedtem a nagyméretű dokumentumok egyik szerkesztési módjával, a nagyon változatosan használható *L^AT_EX* segítségével. Használtam a *TeXnicCenter* szerkesztőt is, de a fordításokhoz testre szabottabb, saját *batch* file-okat írtam.

Úgy gondolom diplomamunkám során sikeresen véghezvittem egy több területet átfogó feladatot, melynek során egyetemi tanulmányaim zárásaként hasznos gyakorlati tapasztatokat szerezhettem több technológia és témaakör megismerésével, használatával.

Függelékek

A függelék

CD-melléklet tartalma



A.1. ábra. CD-melléklet tartalma

B függelék

A dolgozat elektronikus változata

A dolgozat pontos mása elektronikusan, PDF¹ formátumban is elérhető a CD-mellékleten (ld. A. függelék) **diplomaterv_nyomtatott.pdf** néven, illetve található egy elektronikus olvasásra optimalizált változat is **diplomaterv_elektronikus.pdf** néven.

B.1. PDF megjelenítő

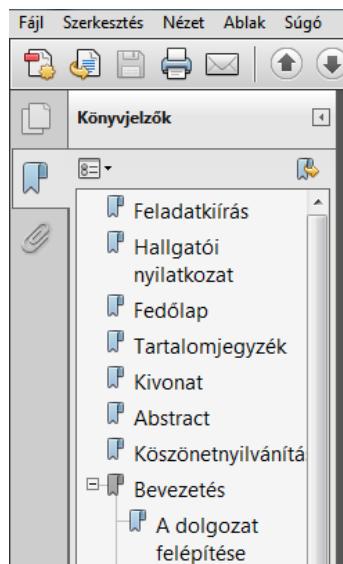
A megtekintéshez a több operációs rendszerre is elérhető *Adobe Reader* nevű PDF-megjelenítő legújabb (11.0.10-es) verziója javasolt, ami ingyenesen letölthető a <http://get.adobe.com/reader/> címről. Természetesen számos más ingyenes megjelenítő is létezik, mint a http://www.foxitsoftware.com/Secure_PDF_Reader/ oldalról letölthető *Foxit Reader*, de a most következő beállítások az Adobe megjelenítőjére vonatkoznak.

Édemes például az olvasás során elrejteni a menüsor és az eszköztárat az F9 és F8 gyorsbillentyűkkel (amelyek ugyanígy vissza is állíthatóak).

B.2. Könyvjelzők

Az elektronikus változatban könyvjelzők találhatóak, aminek megjelenítését az F4-el tudjuk bekapcsolni, ha eddig rejtve lett volna. Ennek hatására a bal oldali könyvjelző ablakban egy tartalomjegyzékfa jelenik meg (B.1. ábra), amelynek bejegyzéseire kattintva az adott fejezet/alfjejezet első oldalára jutunk. Az aktuális pozíónkat a tartalomjegyzékfában szürke kiemelés mutatja.

¹ Portable Document Format, hordozható dokumentumformátum. Az Adobe Systems által kifejlesztett bináris fájlformátum, szöveget, ábrát és képeket tartalmazó dokumentum leírására eszközfüggetlen és felbontásfüggetlen formában.



B.1. ábra. Könyvjelzők használata az Adobe Reader programban.

B.3. Hiperlinkek

Az elektronikus változatban lehetőség adódott hiperlinkek elhelyezésére a dokumentumban, ami tovább növeli a navigálási lehetőségeket és így növeli az olvashatóság mértékét, a dolgozat átláthatóságát. Különös ismertetőjelük, hogy a kurzort fölött helyezve megváltozik a kurzor ikonja, a nyíl helyett egy mancos láthatunk. Ezekre kattintva az adott fejezetre, vagy oldalra ugrik a megjelenítő.

Ilyen hiperlinkek a tartalomjegyzék, irodalomjegyzék, táblázatjegyzék, ábrajegyzék bejegyzései, valamint a tárgymutatóban található oldalszámok. A Bevezetés fejezetben található „dolgozat felépítése” alfejezetben is közvetlen elérés biztosított az ott ismertetett fejezetekhez. A dolgozat során használt *kereszthivatkozások* (pl. fejezetre, ábrára, táblázatra), a *lábjegyzetek* és irodalomhivatkozások is ilyen linkek. A dokumentumban megjelenő URL² -ek pedig az adott weblapot próbálják megnyitni.

Nyomtatott változat elektronikus formája

A `diplomaterv_nyomtatott.pdf` dokumentum a beadott eredeti diplomaterv pontos mása, 2 oldalas nyomtatásra tervezett, kötésmargóval is ellátott, ezért ennek az olvasása során a valósághű megjelenítéshez érdemes beállítani az alábbi két opciót (B.2. ábra):

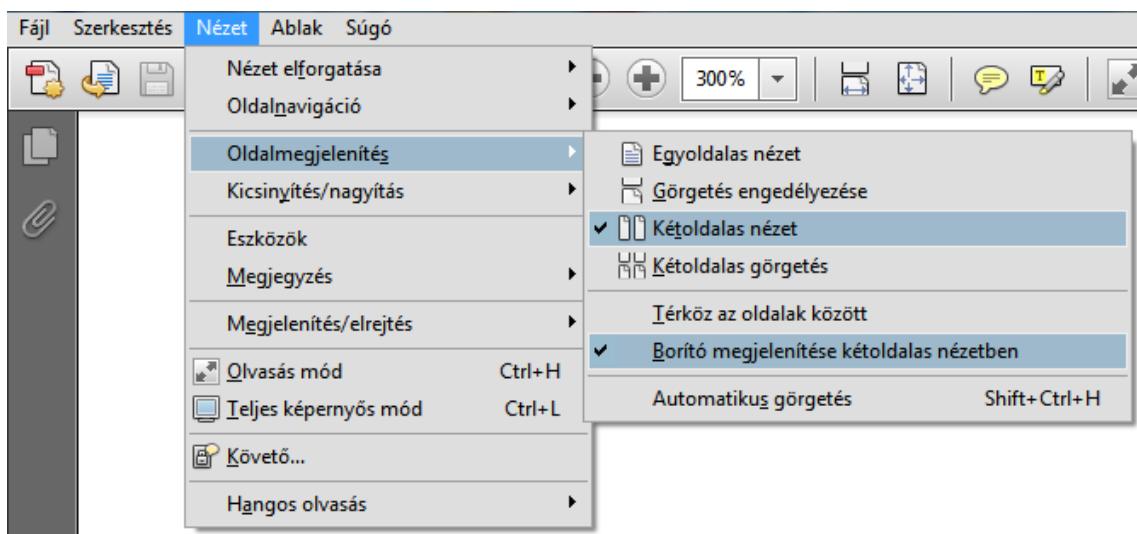
- Oldalmegjelenítés / Kétoldalas nézet

Ez az opció lehetővé teszi, hogy 2 oldal látszódjon egyszerre

- Oldalmegjelenítés / Borító megjelenítése kétoldalas nézetben

Ez az opció a fenti kétoldalas megjelenítést egészíti ki azzal, hogy külön mutatja a fedőlapot (ellenkező esetben ugyanis megcserélődnek a páros és páratlan oldalak helyzete)

² Uniform Resource Locator [egységes erőforrás-azonosító]. Az interneten megtalálható bizonyos erőforrások (például szövegek, képek) szabványosított címe, röviden *webcím*.



B.2. ábra. Két oldalas olvasáshoz javasolt beállítások Adobe Reader programban.

Elektronikus olvasásra optimalizált változat

Az elektronikus olvasáshoz optimalizálva készült egy 1 oldalas változat is diplomaterv_electronikus.pdf néven, ami nem tartalmaz kötésmargókat, de változatlan szövegszélességgel megtartja a tördelést (így azonosak az oldalszámozások). Ebben a változatban – szintén az oldalszámok megtartása mellett – nem szerepelnek az üres oldalak sem.

C függelék

Felhasznált eszközök, programok

A pontos fejlesztő és tesztkörnyezet a C.1 táblázaton látható.

Eszköz/OS/Program	Típus, verzió	Megjelenés dátuma	Licensz
Számítógép	Dell Inspiron 7520	2012.09	saját
Telefon1	Samsung Nexus 5	2013.10.31	saját
Telefon2	Samsung Nexus S i9023	2011.03	saját
PC - OS	Microsoft Windows 7 Professional x64 SP1	2011.02.22	Dreamspark @ BME
Nexus S - OS	Android 4.1.2 (Jelly Bean)	2012.03.28	ingyenes
Nexus 5 - OS	Android 5.1 (Lollipop)	2015.03.09	ingyenes
JDK	8u45 x64	2015.03.03	ingyenes
Eclipse	4.4.2 (Luna) x64	2015.02.19	ingyenes
Android ADT (Android Development Tools)	23.0.6	2015.03	ingyenes
Eclipse CDT (C/C++ Development Tools)	8.6.0	2015.02.13	ingyenes
Android NDK (Native Development Kit)	r10d	2014.12	ingyenes
OpenCV for Android	2.4.9.0 rev 1	2014.04.25	ingyenes
OpenCV for Android	2.4.10 (Manager 2.19)	2014.10.02	ingyenes
OpenCV for Android	2.4.11 (Manager 2.20)	2015.03.05	ingyenes
Adobe Photoshop	CC-x64 (14.0)	2013.04.23	próbaverzió
Geogebra	4.4.12.0	2014.02.14	ingyenes ^a
MikTeX	2.9.5105-x64	2014.01.01	ingyenes
TeXnicCenter	2.02-x64	2013.09.28	ingyenes
LaTeX	LaTeX 2ε	2011.06.24	ingyenes
BIBTeX	0.99d	2010.03	ingyenes
Adobe Reader XI	11.0.10	2014.12.09	ingyenes
Evince	2.32.0.145	2011.10.19	ingyenes

C.1. táblázat. A pontos fejlesztő és tesztkörnyezet.

^a nem üzleti célú felhasználásra

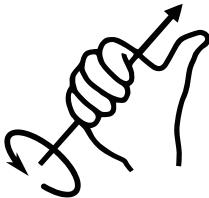
D függelék

Matematikai háttérísmerekek

Az alábbiakban a dolgozatban felhasznált főbb matematikai háttérísmerekek olvashatóak.

D.1. Jobbkéz szabály

Jobbkéz-szabálynak nevezzük azt a szemléletes szabályt, amikor a D.1. ábrán látható módon értelmezzük a vektorok (ill. tengelyek) körüli forgatásokat. Ha jobb kezünk hüvelykujja a vektor irányába néz, akkor a többi ujj által mutatott forgásirányt pozitívnak, az ellenkező irányt pedig így negatívnak definiáljuk.



D.1. ábra. Tengely körüli forgatás előjeles értelmezése jobbkéz szabály szerint

D.2. Lineáris transzformációk

D.2.1. Aktív transzformáció

Aktív transzformációinkat nevezzük egy lineáris transzformációt, ha a bázis helyben marad és a tér vektorait transzformáljuk.

Aktív transzformáció

Legyen v egy tetszőleges vektor, melyen egy T transzformációt hajtunk végre, melynek mátrixa T . Az így kapott v' vektor a következőképp számolható:

$$v' = T \cdot v$$

(D.1)

D.2.2. Passzív transzformáció

*Passzív transzformáció*nak nevezzünk egy lineáris transzformációt, ha csak a bázisvektorokat transzformáljuk és a tér vektorai valójában helyben maradnak, de az új bázisban akarjuk felírni őket.

Jelöljük az eredeti koordináta-rendszeret \mathcal{A} -val, bázisvektorait $\mathbf{a}[i]$ ($i=1\dots n$)-vel, melyek egy adott \mathcal{X} bázisban felírva $\mathbf{a}[i]_{\mathcal{X}}$ ($i=1\dots n$) oszlopvektorokat alkotnak. Az ezekből alkotott mátrixot jelöljük $\mathbf{A}_{\mathcal{X}}$ -szel, a bázis megjelölése nélkül egyelőre csak röviden \mathbf{A} -val.

Jelöljük T -vel azt a lineáris transzformációt, amelyet a bázisvektorokon szeretnénk elvégezni. Az így kapott új \mathcal{B} koordináta-rendszer bázismátrixa hasonlóan legyen \mathbf{B} . Az \mathcal{A} koordináta-rendszerben felírva a transzformációt:

$$\mathbf{B}_{\mathcal{A}} = \underbrace{\mathbf{T}_{\mathcal{A}} \cdot \mathbf{A}_{\mathcal{A}}}_{E} = \mathbf{T}_{\mathcal{A}} \quad (\text{D.2})$$

Egy általános \mathbf{v} vektort minden bázisban felírhatunk:

$$\mathbf{v} = \mathbf{A} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.3})$$

$$\mathbf{v} = \mathbf{B} \cdot \mathbf{v}_{\mathcal{B}} \quad (\text{D.4})$$

Melyek jobb oldalát egyenlővé téve \mathbf{B} inverzével balról szorozva kifejezhető a transzformáció:

$$\mathbf{v}_{\mathcal{B}} = \underbrace{\mathbf{B}^{-1} \cdot \mathbf{A}}_{T_{\mathcal{A} \rightarrow \mathcal{B}}} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.5})$$

Viszont még nem definiáltuk, hogy milyen bázisban is írjuk fel a bázisvektorokat. Ha a \mathcal{A} -ban írjuk fel őket, akkor az \mathcal{A} bázisa pont az egységmátrix lesz:

$$\mathbf{v}_{\mathcal{B}} = \underbrace{\mathbf{B}_{\mathcal{A}}^{-1}}_{T_{\mathcal{A}}^{-1}} \cdot \underbrace{\mathbf{A}_{\mathcal{A}}}_{E} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.6})$$

Így megkapjuk a $T_{\mathcal{A} \rightarrow \mathcal{B}} : \mathbf{v}_{\mathcal{A}} \rightarrow \mathbf{v}_{\mathcal{B}}$ transzformációt:

$$\mathbf{v}_{\mathcal{B}} = \mathbf{T}_{\mathcal{A}}^{-1} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.7})$$

Egyszerűbb jelölésekkel:

Passzív transzformáció (Bázis-transzformáció)

Legyen \mathbf{v} egy tetszőleges vektor. Ha a koordináta-rendszer bázisvektorait egy \mathbf{T} lineáris mátrix-szal transzformáljuk, akkor a \mathbf{v} vektor az új koordináta-rendszerben felírva:

$$\mathbf{v}' = \mathbf{T}^{-1} \cdot \mathbf{v} \quad (\text{D.8})$$

Látható tehát, hogy passzív transzformáció az aktív transzformációinak pont az inverze.

D.2.3. Transzformáció más koordinátarendszerben felírva

Amikor egy adott lineáris transzformációt nem a rendelkezésre álló koordináta-rendszerben a legegyszerűbb felírni, attérhetünk egy másikra.

Jelöljük az eredeti és az új koordináta-rendszert \mathcal{A} , ill. \mathcal{B} -vel, az $\mathcal{A} \rightarrow \mathcal{B}$ koordináta-transzformációt pedig K -val.

Az új \mathcal{B} koordináta-rendszerben felírva egy $T : \mathbf{v} \rightarrow \mathbf{v}'$ transzformációt:

$$\mathbf{v}'_{\mathcal{B}} = T_{\mathcal{B}} \cdot \mathbf{v}_{\mathcal{B}} \quad (\text{D.9})$$

Ha ezt \mathcal{A} -ban szeretnénk felírni, először át kell váltani $\mathbf{v}_{\mathcal{B}}$ -t \mathcal{A} -ba, majd a kapott $\mathbf{v}'_{\mathcal{B}}$ eredményt visszaváltani \mathcal{A} -ba a (D.8). egyenlet szerint:

$$\mathbf{v}_{\mathcal{B}} = K_{\mathcal{A}}^{-1} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.10})$$

$$\mathbf{v}'_{\mathcal{B}} = K_{\mathcal{A}}^{-1} \cdot \mathbf{v}'_{\mathcal{A}} \quad (\text{D.11})$$

Behelyettesítve a (D.9). egyenletbe:

$$\underbrace{\mathbf{v}'_{\mathcal{B}}}_{K_{\mathcal{A}}^{-1} \cdot \mathbf{v}'_{\mathcal{A}}} = T_{\mathcal{B}} \cdot \underbrace{\mathbf{v}_{\mathcal{B}}}_{K_{\mathcal{A}}^{-1} \cdot \mathbf{v}_{\mathcal{A}}} \quad (\text{D.12})$$

Majd $K_{\mathcal{A}}$ -val balról szorozva megkapjuk a transzformációt \mathcal{A} -ban felírva:

$$\mathbf{v}'_{\mathcal{A}} = \underbrace{K_{\mathcal{A}} \cdot T_{\mathcal{B}} \cdot K_{\mathcal{A}}^{-1}}_{\mathbf{K}} \cdot \mathbf{v}_{\mathcal{A}} \quad (\text{D.13})$$

Egy kicsit egyszerűsítve a jelölésekben tehát:

Transzformáció másik koordináta-rendszerben

Jelöljük $K\{T\}$ -vel egy K koordinátarendszer-transzformáció során keletkezett új koordináta-rendszerben felírt T transzformációt az eredeti koordináta-rendszerben felírva. Ekkor

$$\mathbf{K}\{T\} = \mathbf{K} \cdot \mathbf{T} \cdot \mathbf{K}^{-1}, \quad (\text{D.14})$$

ahol \mathbf{K} egy olyan mátrix, ami az eredeti \mathcal{A} koordináta-rendszerben a bázisvektorokat egy másik \mathcal{B} koordináta-rendszer bázisvektoraiba transzformálja és \mathbf{T} egy ebben felírt lineáris transzformáció mátrixa.

Egy ezzel ekvivalens alak (csak ott pont az itt \mathbf{T} -vel jelölt mátrix van kifejezve) van levezetve Freud Róbert Lineáris Algebra könyvében is (ld. [1]).

D.3. Homogén koordináták

Ennek a módszernek a lényege az, hogy a 3 koordináta mellett egy negyediket is használunk, ami így egy origón átmenő 4 dimenziós vektort ír le. Számunkra azonban minden 4 dimenziós vektor ekvivalens, aminek ugyanaz a 3 dimenziós vetülete, tehát a vektorok a középpontos nyújtásra nézve invariánsnak tekinthetőek, valójában egy egyenest alkotnak a négy dimenziós térben, ami a 3 dimenziós terünket egy pontban metszi. A negyedik dimenziót így 1-nek definiáljuk és ha a transzformáció során a negyedik dimenzió értéke megváltozik, akkor a kapott vektort úgy skálázzuk, hogy negyedik koordinátája ismét 1 legyen. Igazából nem szükséges elképzelni ezt a negyedik dimenziót, ez csupán egy matematikai absztrakció, amivel például továbbra is mátrixműveletekkel írható le 3 dimenziós (amúgy nem lineáris) eltolás.

E függelék

Android hibák

Jelen függelékbe gyűjtöttem azokat az Android hiányosságokat, melyekkel munkám során találkoztam. Némelyiküket bejelentettem, van amit már más is hiányolt, van amire megoldást is javasoltam, patch-et is készítettem.

E.1. RotationVector új paramétere

A részletes hibaleírás:

Issue 63268: RotationVector's new extra parameter has some unhandled effects

<https://code.google.com/p/android/issues/detail?id=63268>

Patch javaslat:

<https://android-review.googlesource.com/#/c/71400/>

E.2. Remap 4x4

A hibát az alábbi néven jelentettem:

Issue 58858: remapCoordinateSystem input and output matrices

<https://code.google.com/p/android/issues/detail?id=58858>

Patch javaslat:

<https://android-review.googlesource.com/#/c/71440/>

E.3. Fókusztávolság

A kamera autofocus callback-je <https://code.google.com/p/android/issues/detail?id=14341>

E.4. XXXHDPI ikon

Issue 63022: XXXHDPI launcher not choosed by default on Nexus 5 <https://code.google.com/p/android/issues/detail?id=63022>

F függelék

Rövidítések

Az alábbiakban a dolgozatban felhasznált rövidítések jegyzéke található.

3D	– 3 dimension
ADT	– Android Development Tools
API	– Application Programming Interface
AOSP	– Android Open Source Project
AR	– Augmented Reality
BME	– Budapesti Műszaki és Gazdaságtudományi Egyetem
CDT	– C++ Development Tools
CPU	– Central Processing Unit
CV	– Computer Vision
DVM	– Dalvik Virtual Machine
FPS	– Frames per second
GPS	– Global Positioning System
GPU	– Graphical Processing Unit
HMD	– Head Mounted Display
JDK	– Java Development Kit
NDK	– Native Development Kit
OCR	– Optical Character Recognition
OHA	– Open Handset Alliance
OpenCV	– Open Source Computer Vision Library
OpenGL	– Open Graphics Library
OpenGL ES	– OpenGL for Embedded Systems
OS	– Operating System
PC	– Personal Computer
PND	– Portable Navigation Device
SDK	– Software Development Kit
UI	– User Interface
QCAR	– Qualcomm Augmented Reality
QR code	– Quick Response code
VR	– Virtual Reality
Windows CE	– Windows Embedded Compact
WP	– Windows Phone

Ábrák jegyzéke

1.1.	Az okostelefonok és táblagépek világának kialakulása Készült: saját szerkesztés Photoshop-pal	
	Felhasznált képek: ENIAC [?], IBMPCjr [?], iMac [?], Macbook Pro [?], Macbook Air [?], iPad [?], iPhone [?], Tianhe-2 [?], DynaTAC8000x [?], Sony Ericsson K850i [?], Első telefon [?], Első fényképezőgép [?], Fonográf [?], Cinematográf [?], Mai fényképezőgép [?], Mai videókamera [?]	7
1.2.	Látni a láthatatlan	
	Képek forrásai: Éjjellátó[?], Hőkamera[?], Mordon[?], Edevis tükre[?], Távirányító: saját kép, Vízjelek[?], Occlusion Management[?], WordLens Translator: Google Play: [?]	9
1.3.	A valóság-virtualitás kontinuum Készült: LATEXPGF/TikZ csomaggal.	10
1.4.	Vonalkódok kódolása Készült: Zint Barcode Studio programmal (lásd)	11
1.5.	Zxing (Zebra Crossing): Barcode Scanner	12
1.6.	Google Goggles	12
1.7.	Követésre alkalmas markerek Készült: Microsoft Paint programmal Android target kép forrása: ARToolkit egyik markere.	13
1.8.	IKEA AR katalógusa Kép forrása: [?]	13
1.9.	Orientáció alapú alkalmazások Képek forrásai: [?], [32]	14
1.10.	Google Photosphere	14
1.11.	Layar (Reality Browser)	15
1.12.	Google: Sky Map	15
1.13.	3 dimenziós célpontok. Képek forrása: téglatest célpont: [?], [?] henger célpont: [?], [?]	16
1.14.	Google: Project Tango	16

2.1.	Mobilplatformok világpiaci megoszlása	
	Készült: L ^A T _E X, PGF/TikZ csomag	
	Táblagépek világpiaci megoszlása	
	Forrás: [?]	
	Okostelefonok világpiaci megoszlása	
	Forrás: [?]	19
2.2.	Mobilplatformok százalékos világpiaci eloszlása 2007 óta	
	Készült: TikzPicture nevű L ^A T _E Xcsomaggal.	
	Forrás: Gartner adatai alapján: [?] [?] [?] [?] [?] [?] [?] [?] [?] [?] . . .	20
2.3.	Androidos készülékek verzió szerinti eloszlása	
	Forrás: [?] Készült: PGF/TikZ csomaggal.	21
2.4.	Vuforia feature-ök	27
2.5.	Androidos készülékek verzió szerinti eloszlása	
	Forrás: [?], készült: PGF/TikZ csomaggal.	29
3.1.	Tér-idő-környezet Készült: TikzPicture csomaggal.	32
3.2.	x	33
3.3.	Tér-idő-környezet Készült: TikzPicture csomaggal.	33
3.4.	x	34
3.5.	Analemma 2010. Ladányi Tamás	35
3.6.	WMM 2015 mágneses modell	
	Forrás:	
	– http://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_D_MERC.pdf	
	– http://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_F_MERC.pdf	
	– http://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_I_MERC.pdf	37
3.7.	WGS84 ellipszoid gravitációs formula	
	Készült: Mathematica programmal	38
3.8.	Légköri refrakció szemléltetése Forrás: http://upload.wikimedia.org/wikipedia/commons/5/5d/Re	39
3.9.	Légköri refrakció hatása a holdon Forrás: http://cseligman.com/text/sky/atmosphericrefraction.htm	39
3.10.	Légköri refrakció értéke az emelkedési szög ismeretében. Forrás: [?]	40
3.11.	Holdfázisok Forrás: [?]	41
3.12.	Gömbi és Descartes koordináták megfeleltetése	43
3.13.	CeleBrations indítóikon	
	Készült: Photoshop-pal.	45
3.14.	CeleBrations képernyőképek	
	Készült DDMS Screen Capture-rel.	45

4.1. Fényes objektum digitalizálásának problémája segédfényes fényképezővel Saját törzsvásárlói kártyáról készített saját képek. Transzformáció photoshop-pal.	50
4.2. Tükörkép eltűntetése szemből fényképezés esetén Saját kép. Transzformáció photoshop-pal.	51
4.3. Közlekedési jelzések szándékos elnyújtása Saját kép. Transzformáció Photoshop-pal.	53
4.4. Julian Beever egyik leginkább torzultabb, anamorfikus aszfaltrajza. Körülbelül 15 méteres. Forrás: [?], [?]	54
4.5. Képszerkesztő programokban található transzformációs eszközök Képek forrásai: Screenshot-ok Photoshop CC-ből, saját macskáról.	55
4.6. Perspektíva rektifikációs módszerek Saját kép. Transzformáció Photoshop-pal.	56
4.7. A kameraképen azonos trapéznak látszó, de valójában 2 különböző arányú téglalap Saját kép. Transzformáció Photoshop-pal.	57
4.8. Méretarány megkülönböztethetetlenség. Ábra készült: TikzPicture csomaggal. Képek forrásai: Legkisebb sakktábla:[?], Legnagyobb saktábla:[?] Perspektív transzformáció: Photoshop	60
4.9. Sík szembefordítása a kamerával Készült: TikzPicture csomaggal.	61
4.10. A szükséges forgatás, ami a normálvektort a z tengelyre fordítja és annak előállítása Készült: Geogebra-val	64
4.11. Középpontos vetítés	68
4.12. Projekció inverze	69
4.13. Relatív távolság számítása	73
4.14. OrthoCam alkalmazás indító ikonja, Készült: Photoshop	74
4.15. OrthoPhoto képernyőképek Saját kép. Készült DDMS-sel.	74
A.1. CD-melléklet	82
B.1. Adobe Reader: könyvjelzők	84
B.2. Adobe Reader: 2 oldalas beállítás	85
D.1. Jobbkéz szabály. Forrás: [?]	89

Felhasznált képek forrásai

[1] A világ legkisebb kézzel készített sakktáblája 2005-ig. URL <http://www.kosteniuk.com/>.

com/photosenews2/tagua1.jpg. innen: <http://chessqueen.com/1021.html> (Hozzáférés: 2014. február 23.).

- [2] A világ legnagyobb sakktáblája, 2009-ben.

URL <http://en.chessbase.com/portals/4/files/news/2009/kyffhaeuser07.jpg>. innen: <http://en.chessbase.com/post/the-world-s-biggest-che-game-poibly-by-far> (Hozzáférés: 2014. február 23.).

- [3] Alastor Mordon portré.

URL http://news.bbc.co.uk/media/images/40966000/jpg/_40966996_moody.jpg. innen: http://news.bbc.co.uk/cbbcnews/hi/newsid_4390000/newsid_4393200/4393292.stm, (Hozzáférés: 2012. május 15.).

- [4] Apple iMac számítógép.

URL <https://www.apple.com/hu/imac/>. (Hozzáférés: 2014. május 15.).

- [5] Apple Macbook Air mini notebook.

URL http://images.apple.com/macbookair/images/overview_hero_gallery_everyday.png. (innen: <http://www.apple.com/macbookair/>, Hozzáférés: 2014. május 15.).

- [6] iPad Air 2 táblagép. URL <https://www.apple.com/hu/ipad-air-2/>. (Hozzáférés: 2014. május 15.).

- [7] iPhone 6 okostelefon. URL <https://www.apple.com/hu/iphone-6/>. (Hozzáférés: 2014. május 15.).

- [8] Macbook Pro notebook.

URL <https://www.apple.com/hu/macbook-pro/>. (Hozzáférés: 2014. május 15.).

- [9] Az első fényképezőgép. URL http://www.chinadaily.com.cn/photo/2007-04/13/xin_530404131030787840913.jpg. (innen: http://www.chinadaily.com.cn/photo/2007-04/13/content_849991.htm, Hozzáférés: 2014. május 15.).

- [10] Cinematográf.

URL http://www.nationalmediamuseum.org.uk/globalmedia/nmem/31765_2.JPG. (innen: <http://www.nationalmediamuseum.org.uk/Collection/Cinematography/MotionPictureEquipment/CollectionItem.aspx?id=1931-4>, Hozzáférés: 2014. május 15.).

- [11] Edevis tükre és Harry Potter. URL http://harrypotter.wikia.com/wiki/File:Harry-potter_mirror-of-erised.jpg. (letöltve: 2012. május 15.).

- [12] Edison's Tin-Foil Cylinder Phonograph.

URL http://gasdisc.oakapplepress.com/matrix/mata4_007.jpg. (innen: <http://gasdisc.oakapplepress.com/matrix1.htm>, Hozzáférés: 2014. május 15.).

- [13] ENIAC számítógép.
URL <http://ftp.arl.army.mil/ftp/historic-computers/png/eniac1.png>. (innen: <http://ftp.arl.mil/ftp/historic-computers/>, Hozzáférés: 2014. május 15.).
- [14] Graham Bell: Az első fényképezőgép.
URL <http://www.antiquetelephonehistory.com/images/centennial2.jpg>. (innen: <http://www.antiquetelephonehistory.com/centennial.html>, Hozzáférés: 2014. május 15.).
- [15] Hamis papírpénzeket és fémpénzeket bemutató független numizmatikai információs honlap. URL http://www.hamispenzek.hu/hamis_papirpenz_forint/index.php. (Hozzáférés: 2012. május 15.).
- [16] HDR-FX1 HDV Handycam kamera. URL <http://store.sony.com/wcsstore/SonyStyleStorefrontAssetStore/img/718x407/HDRFX1.png>. (innen: <http://store.sony.com/webapp/wcs/stores/servlet/ProductDisplay?partNumber=HDRFX1>, Hozzáférés: 2014. május 15.).
- [17] Hőkamerás felvételek falfűtés-hűtés rendszerről.
URL <http://www.ziment.hu/kepek/hokamera02.jpg>. innen: <http://www.ziment.hu/hokamera.htm>, (Hozzáférés: 2012. május 15.).
- [18] Holdfázisok, 2003. november. URL http://fotozz.hu/teljes_kepet_mutat?Foto_ID=5054#. Innen: http://fotozz.hu/fotot_megmutat?Foto_ID=5054 (Hozzáférés: 2014. május 15.).
- [19] IBM PCjr számítógép.
URL http://dl.maximumpc.com/galleries/25oldpcs/IBM_PC_jr_01_full.jpg. (Hozzáférés: 2014. május 15.).
- [20] IKEA: Place IKEA furniture in your home with augmented reality, 2013. július. URL <https://www.youtube.com/watch?v=vDNzTasuYEw>. Hozzáférés: 2014. május 15.
- [21] Éjjellátó készülék. URL http://www.atncorp.com/night_vision_images/products/675/images/big/01.jpg. innen: <http://www.atncorp.com/atn-nightraven-2-night-vision-binocular>, (Hozzáférés: 2014. május 15.).
- [22] Julian Beever "Kate's Last Crawl," aszfaltrajza jó szögből. URL <http://www.julianbeever.net/images/phocagallery/gallery/crawl-i.jpg>. innen: http://www.julianbeever.net/index.php?option=com_phocagallery&view=category&id=2 (Hozzáférés: 2014. február 23.).
- [23] Julian Beever "Kate's Last Crawl," aszfaltrajza rossz szögből. "Although not a large drawing, Kate's Last Crawl was one of the most extreme anamorphic distortions with the image stretched to about 15 metres.,,"

- URL <http://www.julianbeever.net/images/phocagallery/gallery/crawl--wrongview-i.jpg>. innen: http://www.julianbeever.net/index.php?option=com_phocagallery&view=category&id=2 (Hozzáférés: 2014. február 23.).
- [24] Motorola DynaTAC 8000x mobiltelefon.
URL <http://media.gdgt.com/img/products/446/9klo/9klo-640.jpg>. (innen: <http://gdgt.com/motorola/dynatac/8000x/>, Hozzáférés: 2014. május 15.).
- [25] Nikon D800E fényképezőgép.
URL http://imaging.nikon.com/lineup/dslr/d800/img/product_01a.png. (innen: <http://imaging.nikon.com/lineup/dslr/d800/>, Hozzáférés: 2014. május 15.).
- [26] Qualcomm.
URL https://developer.vuforia.com/sites/default/files/devGuide_OcclusionManagement.jpg. innen: <https://developer.vuforia.com/resources/sample-apps/occlusion-management>, (Hozzáférés: 2014. május 15.).
- [27] Qualcomm. URL https://developer.vuforia.com/sites/default/files/dev_guide_images/create_targets_devguide/special_case_cylinder.png. innen: <https://developer.vuforia.com/resources/dev-guide/cylinder-target--math>, (Hozzáférés: 2014. május 15.).
- [28] Qualcomm. URL https://developer.vuforia.com/sites/default/files/dev_guide_images/create_targets_devguide/depth.jpg. innen: <https://developer.vuforia.com/resources/dev-guide/multi-targets>, (Hozzáférés: 2014. május 15.).
- [29] Qualcomm: Multi Targets Sample App - Augmented Reality - Vuforia.
URL <https://www.youtube.com/watch?v=Rrl2T9w0ms4>. innen: <https://developer.qualcomm.com/blog/vuforia-26-now-recognizes-and-tracks-cylinder-targets> (Hozzáférés: 2014. május 15.).
- [30] Qualcomm: Vuforia SDK 2.6: Now with Cylindrical Targets.
URL <https://www.youtube.com/watch?v=LqgmlkJcqA4>. Hozzáférés: 2014. május 15.
- [31] Sony Ericsson K850i mobiltelefon.
URL http://index.es/tarhely/636/kepek/k850i_front_angle40_luminious_green_1181866196.jpg. (innen: <http://sonyericsson.index.es/subpaginas.php?a=810>, Hozzáférés: 2014. május 15.).
- [32] Tengely körüli forgatás jobb kéz szabály szerinti értelmezése. URL http://upload.wikimedia.org/wikipedia/commons/archive/3/34/20110401081905%21Right-hand_grip_rule.svg. (Hozzáférés: 2014. február 23.).

[33] Tianhe-2 nevű szuperszámítógép.

URL http://www.popsci.com/sites/popsci.com/files/styles/large_1x_/public/import/2014/Tianhe%202%20Corridor.jpg. Hozzáférés: 2014. május 15.

Táblázatok jegyzéke

2.1. Android verziók, API szintek, NDK verziók és kódnevek közötti összerendelés	23
C.1. Tesztkörnyezet, verziók	87

Kódok jegyzéke

Irodalomjegyzék

Könyvek

- [34] Freud Róbert: *Lineáris Algebra*. 2009, ELTE Eötvös Kiadó, 167. p. ISBN 963-463-471-0. 5.8.1A Tétel.

Segédprogramok

- [35] A 3d library/framework for Android using Java and OpenGL ES. <https://code.google.com/p/min3d/>, 2012. Grafikus toolkit OpenGL ES 1.0-hoz.
- [36] AndAR. <http://code.google.com/p/andar/>. Google Code projekt: forrás és dokumentáció.
- [37] Inc. ARToolworks: ARToolKit on SourceForge. <http://sourceforge.net/projects/artoolkit/>.
- [38] Inc. ARToolworks: ARToolworks Homepage. <http://www.artoolworks.com/>. hivatalos weboldal.
- [39] Inc. ARToolworks: ARToolworks Support Library. http://www.artoolworks.com/support/library/Main_Page. dokumentáció.
- [40] ARToolworks, Inc: 2D barcode marker generator for ARToolKit Professional v4.5.3 and later. <http://www.artoolworks.com/support/app/marker.php>.
- [41] Google ATAP: Project Tango. <https://www.google.com/atap/projecttango/>.
- [42] Wikitude GmbH: Wikitude Augmented Reality SDK 3.3 for Android. <http://www.wikitude.com/products/wikitude-augmented-reality-sdk-mobile/wikitude-sdk-android/>.
- [43] Google: Android open source project. URL <https://source.android.com/>.
- [44] Google: *Android SDK hivatalos, fejlesztői dokumentációja*. 2014. URL <http://developer.android.com/sdk/index.html>.
- [45] Google: Project Tango. <https://www.google.com/atap/projecttango/>, 2014. Sztereokamerával mélységérzékelő tér feltérképezése sensorfusion használatával.

- [46] University of Washington HIT Lab: ARToolKit. <http://www.hitl.washington.edu/artoolkit/>. hivatalos archív weboldal.
- [47] Layar: Homepage. <https://www.layar.com/>. hivatalos honlap.
- [48] Mas Dennis: Android OpenGL ES 2.0/3.0 Engine. <https://github.com/MasDennis/Rajawali>. Grafikus toolkit OpenGL ES 2.0/3.0-hoz.
- [49] OpenCV: *OpenCV*. URL <http://opencv.org/>.
- [50] OpenCV: *OpenCV for Android*.
URL <http://opencv.org/platforms/android.html>.
- [51] Qualcomm: *Computer Vision (FastCV)*.
URL <https://developer.qualcomm.com/mobile-development/add-advanced-features/computer-vision-fastcv>.
- [52] Qualcomm: *Vuforia Android SDK dokumentációja*. 2012.
URL <https://developer.vuforia.com/resources/sdk/android>.
- [53] ZXing: Official ZXing (Zebra Crossing) project home. <https://github.com/zxing/zxing>. GitHub projekt: forrás és dokumentáció.

Applikációk

- [54] Inter IKEA Systems B.V.: IKEA Catalogue. https://play.google.com/store/apps/details?id=com.ikea.catalogue.android&hl=en_GB, 2014. április. Az IKEA AR katalógusa. Hozzáférve: 2014.05.15.
- [55] Sky Map Devs: stardroid. <https://code.google.com/p/stardroid/>. Sky Map open source project.
- [56] Sky Map Devs: Sky Map. <https://play.google.com/store/apps/details?id=com.google.android.stardroid>, 2011. augusztus. Csillagászati AR alkalmazás.
- [57] Google: Google Play. <https://play.google.com/store/apps/>. A Google szoftverboltja androidos alkalmazásokhoz.
- [58] Google, Inc.: Google Fényképezőgép. <https://play.google.com/store/apps/details?id=com.google.android.GoogleCamera>, 2014. május.
- [59] Google Inc.: Google Goggles. <https://play.google.com/store/apps/details?id=com.google.android.apps.unveil>, 2013. november.
- [60] Ltd IntSig Information Co.: CamScanner - Phone PDF Creator. <https://play.google.com/store/apps/details?id=com.intsig.camscanner>, 2014. április. Ingynéres okostelefonos szkenner-alkalmazás, több mint 40 millió letöltés.

- [61] The Grizzly Labs: Genius Scan - PDF Scanner. <https://play.google.com/store/apps/details?id=com.thegrizzylabs.geniusscan.free>, 2014. április. Ingyenes okostelefonos szkenner-alkalmazás.
- [62] Layar: Layar. <https://play.google.com/store/apps/details?id=com.layar>, 2014. április. valóság bőngésző. Hozzáférve: 2014.05.15.
- [63] Sam Lu: 3D Compass+ (AR Compass). <https://play.google.com/store/apps/details?id=com.a0soft.gphone.aCompassPlus>, 2014. április. térbeli iránytű alkalmazás.
- [64] OpenCV: OpenCV Manager. <https://play.google.com/store/apps/details?id=org.opencv.engine&hl=hu>, 2014. május. OpenCV 2.4.9.0 rev 1.
- [65] Quest Visual, Inc.: Word Lens Translator. <https://play.google.com/store/apps/details?id=com.questvisual.wordlens>, 2014. április. Hozzáférve: 2014.05.15.
- [66] STOIK Soft: Mobile Doc Scanner Lite. <https://play.google.com/store/apps/details?id=com.stoik.mdscanlite>, 2014. március. Egy okostelefonos szkenner-alkalmazás ingyenes változata.
- [67] Soulbit7: AR Invaders. <https://play.google.com/store/apps/details?id=com.soulbit7.game.arinvaders>, 2011. augusztus. AR lövöldözős játék.
- [68] ZXing Team: Barcode Scanner. <https://play.google.com/store/apps/details?id=com.google.zxing.client.android>, 2014. március. A legismertebb barcode (pl. vonalkód) leolvasó alkalmazás.

Információk

- [69] *Personal Computer Family Service Information Manual*. URL <http://ibm-pc.org/manuals/ibm/5150/Pcsim1-9.pdf>. (hozzáférés: 2012.05.15).
- [70] Peter Ha: Motorola DynaTAC 8000x, 2010. október 25. URL http://www.time.com/time/specials/packages/article/0,28804,2023689_2023708_2023656,00.html. (hozzáférés: 2012.05.15).
- [71] Open Handset Alliance: Home page. <http://www.openhandsetalliance.com/>.
- [72] Howard S. Friedman: Matrix and Catalog Numbers in G&S Discography. 2009. július 12.
URL <http://gasdisc.oakapplepress.com/matrix1.htm>. (hozzáférés: 2012.05.15).
- [73] Introduction to Computer Organisation (AY2011/2012) Semester 2.
URL <http://www.comp.nus.edu.sg/~cs2100/lect/cs2100-9-Computer-Organisation.pptx>. (hozzáférés: 2012.05.15).

- [74] John Presper Eckert John Mauchly: The History of the ENIAC Computer.
URL <http://inventors.about.com/od/estartinventions/a/Eniac.htm>. (hozzáférés: 2012.05.15).
- [75] Le Prince Single-lens Cine Camera. URL <http://www.nationalmediamuseum.org.uk/Collection/Cinematography/MotionPictureEquipment/CollectionItem.aspx?id=1931-4>. (hozzáférés: 2012.05.15).
- [76] Virtual Public Library: Alexander Graham Bell életrajza.
URL <http://www.alexandergrahambell.org/>. (hozzáférés: 2012.05.15).
- [77] QR Code Features. URL <http://www.denso-wave.com/qrcode/qrfeature-e.html>. (hozzáférés: 2012.05.15).
- [78] The reflex box camera obscura by Johann Zahn, 1685, from 'The History of the Camera Obscura' (engraving).
URL <http://www.bridgemanart.com/asset/115145/English-School-19th-century/The-reflex-box-camera-obscura-by-Johann-Zahn-1685?lang=en-GB>. (hozzáférés: 2012.05.15).
- [79] The Top 500 Supercomputers list., 2014. november.
URL <http://www.top500.org/lists/2014/11/>. (hozzáférés: 2015.05.09).

Statisztikák

- [80] Gartner: Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008, 2009. 03.
URL <http://www.gartner.com/newsroom/id/910112>.
- [81] Gartner: Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009, 2010. 02.
URL <http://www.gartner.com/press-releases/2010/02/23/egham-uk-february-23-2010-view-all-press-releases>.
- [82] Gartner: Gartner Says 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011, a 19 Percent Increase Year-on-Year, 2011. május. URL <http://www.gartner.com/newsroom/id/1689814>.
- [83] Gartner: Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent, 2011. november.
URL <http://www.gartner.com/newsroom/id/1848514>.
- [84] Gartner: Gartner Says Sales of Mobile Devices in Second Quarter of 2011 Grew 16.5 Percent Year-on-Year; Smartphone Sales Grew 74 Percent, 2011. augusztus. URL <http://www.gartner.com/newsroom/id/1764714>.
- [85] Gartner: Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012, 2012. február. URL <http://www.gartner.com/newsroom/id/2335616>.

- [86] Gartner: Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009, 2012. május. URL <http://www.gartner.com/newsroom/id/2017015>.
- [87] Gartner: Gartner Says Worldwide Sales of Mobile Phones Declined 2.3 Percent in Second Quarter of 2012, 2012. augusztus.
URL <http://www.gartner.com/newsroom/id/2120015>.
- [88] Gartner: Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent, 2012. november.
URL <http://www.gartner.com/newsroom/id/2237315>.
- [89] Gartner: Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth, 2012. február.
URL <http://www.gartner.com/newsroom/id/1924314>.
- [90] Gartner: Gartner Says Smartphone Sales Surpassed One Billion Units in 2014, 2015. 03. URL <http://www.gartner.com/newsroom/id/2996817>.
- [91] Google: Dashboards, Open GL Version. Data collected during a 7-day period., 2015. május. URL <http://developer.android.com/about/dashboards/index.html>.
- [92] Google: Dashboards, Platform Versions. Data collected during a 7-day period., 2015. május. URL <http://developer.android.com/about/dashboards/index.html>.
- [93] IDC: Android and iOS Squeeze the Competition, Swelling to 96.3Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC, 2015. 02. URL <http://www.idc.com/getdoc.jsp?containerId=prUS25450615>.
- [94] IDC: Worldwide Tablet Growth Hits the Brakes, Slowing to the Low Single Digits in the Years Ahead, According to IDC, 2015. 03.
URL <http://www.idc.com/getdoc.jsp?containerId=prUS25480015>.

Tárgymutató

- L^AT_EX, 31
- 3D Compass+, 14
- Adobe Reader, 83
- ADT, 31
- aktív transzformáció, 89
- alkalmazásbolt, 20
- analemma, 35
- AndAR, 26
- Android, 21
- Android Inc., 21
- AOSP, 22
- AOT, 22
- API, 22
- App Store, 24
- AR, 8
- AR Invaders, 14
- ART, 22
- ARToolkit, 26
- ARToolkitPlus, 26
- atmospheric refraction, 39
- Augmented Reality, 8, 10
- Augmented Virtuality, 10
- azimut, 33
- Barcode Scanner, 12
- Bennett, 39
- billentő, 62
- birdseye view, 62
- Compass, 14
- Computer Vision, 28
- csavaró, 62
- CV, 28
- Dalvik VM, 22
- deklináció, 36
- DVM, 22
- elevation, 33
- Ellipszoid távolság, 44
- FastCV, 28
- feature-phone, 5
- Foxit Reader, 83
- fragment, 21
- Fragment API, 22
- fragmentáció, 22
- fragmentáltság, 21
- frame marker, 13
- gömbi távolság, 43
- gépi látás, 28
- Geometria alapú AR, 16
- Goggles, 12
- Google, 21
- GPS, 15
- greenbox, 10
- gyorsulásmérő, 5
- Haversine formula, 44
- Head Mounted Display, 10
- helyi polgári idő, 34
- hiperlink, 84
- HMD, 10
- Homogén koordináták, 92
- homogén koordináta, 66
- HTC Dream, 21
- image target, 13
- inklináció, 36
- iOS, 24
- iPad, 24
- iPhone, 24
- iPod Touch, 24
- Java, 22

- JIT, 22
JNI, 22
jobbkéz-szabály, 63
könyvjelző, 83
középpontos vetítés, 67
Karney, 44
kereszthivatkozás, 84
keystone correction, 56
keystone-effect, 54
keystoning, 56
kibővített valóság, 8
kiterjesztett valóság, 8, 10
kiterjesztett virtualitás, 10
légköri refrakció, 39
latitude, 33
Layar, 15
Linux, 22
local terrestrial time, 34
longitude, 33
mágneses deklináció, 36
mágneses elhajlás, 36
mágneses inklináció, 36
madárnézet, 62
marker, 13
metadata, 58
min3d, 30
Mixed Reality, 10
mobilkészülék, 20
NDK, 22
netbook, 5
notebook, 5
okostelefon, 5
OpenCV, 28
OpenGL ES, 29
OrthoCam, 74
orthophoto, 62
ortofotózás, 62
OS, 19
passzív transzformáció, 90
PC, 5
PDA, 5
PDF, 83
perspective distortion, 53
perspective rectification, 56
perspektív rektifikáció, 56
perspektív torzulás, 53
PhotoSphere, 14
pitch, 62
pontkód, 11
QCAR, 27
QR-kód, 11
Qualcomm, 27
Rajawali, 30
sík távolságképlet, 42
Sæmundsson, 39
SDK, 21, 24
sensor fusion, 47
Sky Map, 15
smartphone, 5
subnotebook, 5
Sudoku, 12
T-mobile G1, 21
táblagép, 5
Tango, 16
target, 13
tombstone correction, 56
tombstone-effect, 54
toolkit, 19
tracker, 13
trapéztorzítás, 56
trapéztorzulás, 54
ultrabook, 5
UML, 31
URL, 84
Vincenty formula, 44
virtuális valóság, 10
Virtual Reality, 10
vonalkód, 11

Vuforia, 27

WGS84, 44

XCode, 24

zónaidő, 34

Zxing, 29