

TDK-dolgozat

Bónus Zoltán

Kelemen Bálint

*Óbudai Egyetem
Neumann János Informatikai kar
Informatikai Automatizált Rendszerek szakirány
2010-2011*

MoBRo

Háromdimenziós környezetleképző robot



*Bónus Zoltán
Kelemen Bálint
Témavezető: Dr. Vámosy Zoltán, egyetemi docens*

Tartalomjegyzék

1. BEVEZETŐ	5
2. CÉLKITŰZÉS	5
3. HASONLÓ RENDSZEREK	5
3.1. SZTEREO LÁTÁS ÉS TÉRKÉPEZÉS SZINKRONIZÁLATLAN KAMERÁKKAL	5
3.2. ROBOT OPERATING SYSTEM (ROS)-EN ALAPULÓ RENDSZEREK.....	6
3.2.1. Térképezés és navigálás egy quadrotoros helikopterrel.....	6
3.2.2. Automatikus háromdimenziós térképezés földi egységgel	6
3.3. PROFORMA.....	7
3.4. REAL-TIME 3D MODEL ACQUISITION	7
3.5. 4D VIEW SOLUTIONS.....	7
3.6. AUTOMATIKUS MODELLÉPÍTÉS FOLYAMATOS KAMERAKÉPEK ALAPJÁN	8
3.7. ÖSSZEFOGLALÓ.....	8
4. RENDSZER FELADATAI	9
4.1. KÉPEK BEOLVASÁSA	9
4.2. KOMMUNIKÁCIÓ	10
4.3. VEZÉRLÉS.....	12
4.4. IRÁNYÍTÁS.....	13
4.5. KAPCSOLATTARTÁS A FELHASZNÁLÓVAL	14
4.6. KÉPEK ELŐFELDOLGOZÁSA	14
4.7. PONTOK ELHELYEZÉSE A TÉRBEN	14
4.8. MODELLEZÉS ÉS TEXTÚRÁZÁS	14
4.9. A MODELL MEGJELENÍTÉSE.....	15
4.10. PUBLIKUS ADATOK KÖZZÉTÉTELE	15
5. RENDSZER MODUL TERVE KRONOLÓGIA SORRENDEN	16
6. A RENDSZER BEMUTATÁSA.....	18
6.1. ROBOT LEÍRÁSA, ISMERTETÉSE	18
6.2. A SZERKEZET MŰSZAKI PARAMÉTEREI.....	18
6.3. A ROBOTVEZÉRLŐ ELEKTRONIKA.....	18
6.3.1. Tápellátás	18
6.3.2. Mikrokontroller	19
6.3.3. Kapcsolattartás az elektronikával	20
6.3.4. Robot vezérlése.....	21
6.3.5. Emulátor	22
6.3.6. A robot biztonsági megoldása.....	22
6.4. FELADATOK MEGVALÓSÍTÁSA	24
6.4.1. Kommunikáció soros porton.....	24
6.4.2. Hálózati kommunikáció	24
6.4.3. Képkinyerés kamerából	25
6.4.4. Irányítás.....	26
6.4.5. Adatküldési problémák a hálózaton.....	27
6.4.6. Publikus adatok, képek szolgáltatása.....	27

6.5.	MODELLÉPÍTÉS ÉS MEGJELENÍTÉS	27
7.	A MODELLÉPÍTÉSHEZ HASZNÁLT KÉPFELDOLGOZÁSI MÓDSZEREK.....	28
7.1.	AZ ELŐFELDOLGOZÁS MÓDSZEREI.....	28
7.1.1.	Jellegzetes pontok detektálása.....	28
7.1.2.	FAST: Features from Accelerated Segment Test.....	28
7.1.3.	SURF: Speeded Up Robust Features	29
7.1.4.	Jellegzetes pontok követésének problémája	29
7.1.5.	A pontkeresés megvalósítása	30
7.2.	PONTPÁROK ELHELYEZÉSE A TÉRBEN	32
7.2.1.	A projektív kamera	32
7.2.2.	A kameramátrix felbontása.....	34
7.2.3.	Pontok elhelyezése a térben	34
7.2.3.1.	A rekonstrukció bizonytalansága	35
7.2.3.2.	Projektív rekonstrukció.....	36
7.2.3.3.	Affin rekonstrukció	36
7.2.3.4.	Metrikus rekonstrukció	37
7.2.3.5.	Közvetlen metrikus rekonstrukció.....	38
7.2.4.	Az epipoláris geometria.....	39
7.2.4.1.	Az F alapmátrix	40
7.2.4.2.	Az esszenciális (E) mátrix	41
7.2.5.	Az F mátrix számítása.....	42
7.2.5.1.	A 7 pontos algoritmus	42
7.2.5.2.	A normalizált 8 pontos algoritmus	43
7.2.5.3.	„Arany középút” módszer	43
7.2.5.4.	RANSAC.....	44
7.2.5.5.	F mátrix számítása speciális esetekben	45
7.2.5.6.	Kameramátrixok számítása.....	46
7.2.5.7.	Kameramátrixok az E mátrixból.....	46
7.2.5.8.	Háromszögelés	47
7.2.6.	Kamera kalibráció.....	48
7.2.7.	Pontelhelyezés a gyakorlatban.....	49
7.3.	VIZUALIZÁCIÓ JAVÍTÁSA: TESTHÁLÓ ÉS TEXTÚRÁZÁS.....	49
7.3.1.	Delaunay-háromszögek.....	50
7.3.2.	Alfa-háló	50
7.3.3.	Textúrázás	51
7.4.	KIEGÉSZÍTÉSEK.....	52
7.4.1.	DLT (Direct Linear Transformation) algoritmus.....	52
7.4.2.	SVD (Singular Value Decomposition) algoritmus.....	53
7.4.3.	Givens-forgatás és mátrixok RQ dekompozíciója	53
8.	MEGJELENÍTÉS.....	55
9.	TESZTELÉS	56
9.1.	ELSŐ TESZTELÉSEK NOTEBOOK-NOTEBOOK ADAT ÁTVITEL	56
9.2.	MÁSODIK TESZT: NOTEBOOK-NOTEBOOK KÉPÁTVITEL	56
9.3.	μC PROGRAM TESZTELÉS:	56
9.4.	ÜZENETKÜLDÉSI TESZT A ROBOTNAK WIFI-RŐL	56
9.5.	μC TOVÁBBFEJLESZTÉSE MOTORVEZÉRLÉSI TESZTTTEL	56
9.6.	IRÁNYÍTÁSI MODULTESZTELÉS (JOYSTICK).....	57

9.7.	JELLEGZETES PONTKERESŐ ÉS PONTKÖVETŐ TESZTELÉSE	57
9.8.	PONTOK ELHELYEZÉSE A TÉRBEN	58
9.9.	TESTHÁLÓ GENERÁLÁSA	61
9.10.	TEXTÚRÁZÁS	63
10.	EREDMÉNYEK ÉRTÉKELÉSE	65
11.	ÖSSZEFOGLALÁS.....	67
	IRODALOMJEGYZÉK.....	68
	ÁBRAJEGYZÉK.....	70

1. Bevezető

Tanulmányaink során mindig is vonzottak minket az olyan összetett megoldások, amik a hardvert és szoftvert ötvözve, kilépnek a számítógép-monitor dimenziójából. Ezért egy olyan rendszer megalkotásán kezdtünk gondolkodni, ami valamilyen interakcióba képes lépni a környezetével. A másik igen fontos terület, ami az érdeklődési körünkbe tartozik az a számítógépes képfeldolgozás, ezen területek ötvözésével sikerült egy olyan rendszer tervét és prototípusát kidolgozni, ami ezen dolgozat tárgyát képezi.

2. Célkitűzés

A mai világban nagy hangsúlyt fektetünk a háromdimenziós ábrázolásra. Emiatt is egyre jobban terjednek azok a megoldások, amik segítségével egyre könnyebben lehet háromdimenziós teret leképezni a számítógépi modellre. Mi is úgy döntöttünk, hogy egy, a környezetet leképező rendszert kívánunk kidolgozni. Szeretnénk elérni, hogy a modellezés feladatát teljesen átvegye a számítógép és az általa készített metrikus modell további emberi finomítás nélkül legyen használható. Egy ilyen paraméterekkel rendelkező megoldás használható lenne akár olyan feltérképezési feladatoknál, ahol az emberi jelenlét nem lehetséges, vagy veszélyes a terep ismerete nélkül. A rendszerhez a képfeldolgozáson kívül egy robot is kapcsolódik. Ezen az egységen elhelyezett kamera képét továbbítva egy nagyobb teljesítményű szerver felé, az elkészíti a környezet modelljét. A felhasználó irányítja a robotot, ezáltal egy célzott területet képes lemodellezni.

3. Hasonló rendszerek

Annak érdekében, hogy rendszerünk pontos tervét kialakítsuk, elsőként megvizsgáltuk, hogy milyen következtetéseket lehet levonni a hasonló programok megközelítéséből, az alkalmazott módszereikből.

3.1. Sztereo látás és térképezés szinkronizálatlan kamerákkal

Az első vizsgált megoldás egy MIT-n (Massachusetts Institute of Technology) készült dolgozat, aminek a címe Stereo Vision And Mapping With Unsyncronized Cameras[1]. A rendszer a környezeti felderítést tűzi ki célul, hogy egy robot képes legyen a környezetét érzékelni és így navigálni abban. Alapfelvetése, hogy csökkenteni kell az irányításba történő emberi beavatkozást és csak a célállomás pozícióját kelljen a robotnak megadni. A megoldást három problémakörre bontja: a képek elkészítése, azok zajszintjeinek kezelése, valamint a háromdimenziós modell elkészítése és tárolása. A méréseiből az derül ki, hogy a szinkronizált kamerás rendszerek sokkal pontosabb eredményeket adnak, amit a mi tesztjeink is igazoltak a későbbiekben. A szerző mind mono, mind sztereo kamerarendszerekkel végzett méréseket. Következtetése, hogy a sztereo rendszer robosztusabb és hibátűrőbb eredményeket produkált. A mi rendszerünk sztereo képfeldolgozására egyelőre nem lesz alkalmas, mert a roboton egykamerás rendszert kívánunk elhelyezni, lehetőleg valamilyen okostelefont

3.2. Robot Operating System (ROS)-en alapuló rendszerek

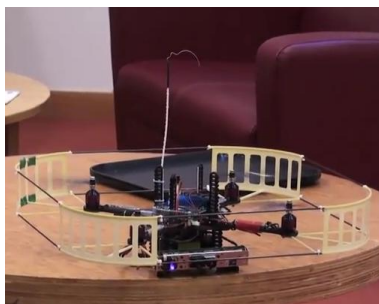
Kutatásaink során rábukkantunk egy olyan gyűjtő oldalra ahol egy speciális platformra az ún. ROS-ra épülő projekteket fogják össze. Ez a speciális platform egy operációs rendszer robotok számára. Működését tekintve azokat a szolgáltatásokat biztosítja, amit egy operációs rendszertől a felhasználó elvár. Hardveres absztrakciót lát el és alacsony szinten vezérli az eszközöket, valamint megvalósít széleskörűen használt funkciókat. Biztosít folyamatok közötti üzenetváltásokat, valamint csomagkezelést.

3.2.1. Térképezés és navigálás egy quadrotoros helikopterrel¹

Egy quadrotoros repülő eszközön (3-1. ábra, bal) elhelyezett Kinect szenzor (3-2. ábra, közép) segítségével építettek egy olyan rendszert, ami olyan helyen is képes navigálni és térképezni ahol a GPS (Global Positioning System) rendszerek használata nem lehetséges. A navigáló és feldolgozó egység magán a rotoros egységen van elhelyezve. Az eszközön azokat az adatokat nyerik ki, amik magához a navigációhoz szükségesek. A MIT-n megalkotott berendezés, képi feldolgozó algoritmusok segítségével navigál és vezérelve az elektronikát, stabilizálja a gépet. Képes meghatározni az aktuális pozícióját és nem csak stabilizál, de térbeli mozgást is vezérli. A kapott eredmények és képek továbbítódnak a földi állomásra és ott történik magának a háromdimenziós modellnek az építése a Washingtoni egyetemen (University of Washington) fejlesztett RGBD-SLAM (Simultaneous Localization And Mapping) programkönyvtárának felhasználásával. A kapott modell valójában egy halmaz, ami metrikusan elhelyezett képpontokból épül fel a térben.

3.2.2. Automatikus háromdimenziós térképezés földi egységgel

Waterlooi egyetem (University of Waterloo) hallgatói által jegyzett projektben egy Clearpath Husky A200-ra (3-3. ábra) helyezett Kinect szenzort és számítógépet alkalmaztak a képfeldolgozási és navigálási feladatok megoldására. Valós időben háromdimenziós színes ponthalmazt képezik le a látottakat. A rendszer külső beavatkozás nélkül képes bejárni a beltéri környezetet. A projektben a következő modulokat használják fel: ROS, OpenCV – nyílt forráskódú képfeldolgozó programkönyvtár-, GPUSURF – a SURF (7.1.3 fejezet) algoritmus CUDA² rendszerre optimalizált változata. A grafikus processzor felhasználásával jelentősen javítottak a rendszer összteljesítményén.



3-1. ábra: Quadrotor
[<http://groups.csail.mit.edu/rrg/>]



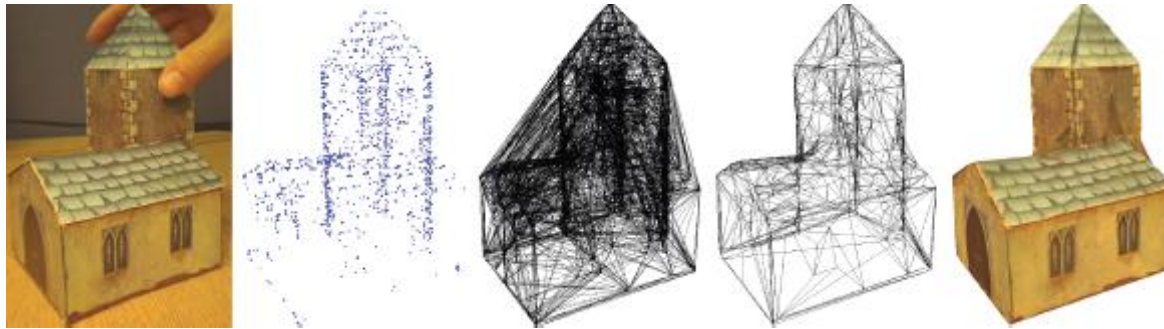
3-2. ábra: Kinect Szenzor



3-3. ábra: Clearpath Husky A200
[<http://www.clearpathrobotics.com/husky>]

¹ <http://groups.csail.mit.edu/rrg/index.php?n=Main.VisualOdometryForGPS-DeniedFlight>

² NVIDIA által fejlesztett párhuzamos feldolgozást megvalósító architektúra, ami a grafikus processzort használja fel a számítások elvégzésére



3-4. ábra: A ProFORMA rendszer modellezési lépései

3.3. ProFORMA

A Cambridge-i egyetem hallgatója által kifejlesztett rendszer egyetlen kamera képét használja fel a modellépítéshez. A program fő célja ugyan nem a teljes környezet, csupán egy, a kamera előtt forgatott tárgy modellezése, a felhasznált módszerek viszont jó kiindulási alapként szolgálnak saját céljainkhoz. A folyamat során első lépésként a képeken található jellegzetes pontok keresése és párosítása történik meg, FAST módszer használatával. Ezután a pontok térbeli pozíciójának kiszámítása következik, majd Delaunay-háromszögeléses módszerrel készül el a pontok által határolt test hálója. A valójában nem létező élek eltávolítása után kerül fel a hálót alkotó háromszögekre a megfelelő textúra.

3.4. Real-Time 3D Model Acquisition

A módszer [2] alapvetően egy lézer pásztát vetít ki a modellezendő objektumra, majd annak kivetülését egy kamerával érzékeli. Az egymás utáni kameraképekből számítja ki az objektum pontjainak háromdimenziós pozícióját, majd valós időben jeleníti meg a pontfelhőt. A teljes modell elkészítéséhez felhasználói interakció szükséges, a tárgyat manuálisan kell körbeforgatni a kamera előtt. A modell folyamatosan bővül az új, kiszámított pontokkal, amelyek vizsgálat után kerülnek beillesztésre. Ha a tárgy egy időre kikerülne a kamera látószögéből, a program jelzi, hogy hová kell visszahelyeznünk, hogy a modellezési folyamat folytatódhasson. A folyamat eredményeképpen létrejövő modell sajnos nem teljes, vannak benne „lyukak”, amikről nem sikerült információt szerezni. Ezt a készítők utófeldolgozás útján javítják, sokkal élethűbb modellt létrehozva. A szükséges rendszerkövetelmények: egy webkamera (640x480-as felbontású), legalább 2,4 GHz-es dual core processzor.

3.5. 4D View Solutions

A programcsomag [3] professzionális megoldásokat kínál a háromdimenziós modellek előállítására. Akár nyolcvan kamerából álló hálózat kezelésére is képes. A felhasználni kívánt teret gyakorlatilag körbeveszik kamerákkal, ezután a térben lévő tárgyak modellje előállítható, vagy már meglévő virtuális környezetbe illeszthető a kamerák által közrefogott tér. Az elhelyezett kamerák kalibrálását a program automatikusan végzi, mindössze néhány perc alatt. Elsősorban filmes animációk gyors elkészítésére használható. A szükséges rendszerkövetelmények: több, nagy teljesítményű kamera, nagy teljesítményű számítógép az adatok feldolgozásához.

3.6. Automatikus modellépítés folyamatos kameraképek alapján

A vizsgált rendszer [4] célja egy adott környezet háromdimenziós modelljének elkészítése, kizárólag egy mozgó kamera által készített képek alapján. Alapvetően a képkockák hármass kapcsolatára épít, ezeken keres jól követhető, jellegzetes pontokat, illetve vonalakat. Ezek összefüggéseiből képes kiszámítani térbeli pozíciójukat. Ezután a kapott eredményekre síkokat illesztve próbálja megtalálni a feldolgozott jelenetet legélethűbben visszaadó modellt. Végző lépésként erre a modellre kerül fel az eredeti képkockák alapján kivágott textúra. A módszer jelentősége, hogy működéséhez semmilyen előzetes információ nem szükséges az adott jelenetet készítő kameráról, vagy magáról a jelenetről, továbbá csupán egyetlen kamerával is működik.

3.7. Összefoglaló

A felkutatott rendszerek kézzelfogható eredményeket produkálnak a Kinect szenzor segítségével, amelyen sztereo kamerarendszer és mélységérzékelő is található. Ezek segítségével igen nagy pontossággal lehet metrikus modellt alkotni, amit navigáláshoz valamint háromdimenziós modell készítéséhez is fel lehet használni. A mi rendszerük célkitűzése a modellépítés, ehhez a fejlesztés során egyetlen átlagos kamera képét kívántuk felhasználni. Mélységérzékelő hiányában a rendszer képenkénti eltérésére alapozzuk a modellezést.

A 3.4. pontban bemutatott rendszer a lassú utófeldolgozás végeztével pontos modellt készít, viszont a valós időben elkészült modellben már találhatóak lyukak, így az nem tekinthető tökéletesnek. Hátránya, hogy csak egy tárgyat képes lemodellezni, ami eléggé leszűkíti felhasználhatóságát. További problémát okoz használhatóságában, hogy a projektort össze kell hangolni a kamerával, hogy a pásztázó fény feldolgozása ne adjon téves információkat. Előnye, hogy kevés erőforrás szükséges működéséhez.

A ProFORMA rendszer nagy előnye a gyorsasága, hiszen teljes mértékben valós idejű modellépítést érhetünk el vele. További előnye, hogy nincs nagy erőforrásigénye, hiszen egyetlen webkamerával és egy ma már átlagosnak mondható asztali számítógéppel dolgozik. Hátrányként felróhatjuk, hogy az elkészült modell nem teljesen pontos, a pontok „összekötéséből” létrejövő felszíni háló előállítási algoritmusának sajátosságai miatt, valamint, hogy ez a módszer is csupán egyetlen tárgy háromdimenziós leképezésére képes egyszerre.

A 4D View Solutions cég által készített, többkamerás módszer minden igényt kielégít. Nagy felbontású kamerákkal dolgozik, így az eredmény pontosabb, akár egész szobákat modellezhetünk vele, az elkészült modelleket beépíthetjük későbbi valós környezetekbe. Hátrányként megemlítendő, hogy a komplett rendszer kezelése komoly szaktudást igényel, valamint a kamerák telepítése körülményes lehet. Meg kell említenünk még, hogy egy ilyen rendszer kiépítésének erős anyagi vonzatai is vannak.

A Fitzgibbon és Zisserman által kidolgozott rendszer gyakorlatilag egy professzionális megoldása az általunk megfogalmazott feladat modellezési részének. Megvalósítási részletei jó kiindulópontul szolgálnak saját rendszerünk megalkotásához.

4. Rendszer feladatai

4.1. Képek beolvasása

A feldolgozási folyamat kezdete a képek beolvasása. Az egyik legfontosabb feladat, hogy a rendszer számára megfelelő minőségű képeket biztosítsanak az alkalmazandó képalkotók. Ha nem megfelelő a képek minősége a feldolgozó algoritmusok sem lesznek képesek megfelelő hatékonysággal működni.

A digitális képalkotó eszközök két fő technológiát alkalmaznak: CMOS (Complementary Metal Oxide Semiconductor), illetve CCD (Charge Coupled Device) érzékelőt. Mind két technológia alapja a pixeleket alkotó fém oxid félvezető (MOS - Metal Oxid Semiconductors). Ezek eltárolják az adott pixelen generálódott töltéseket, amit a fényintenzitás gerjeszt, majd egy mintavétellel meghatározza azoknak nagyságát. Két érzékelő szenzor közti különbség felépítésükben keresendő. Egy CCD szenzor esetén a kép elkészítése után a kiolvasás szekvenciálisan történik egy közös kivezetésre, majd ezek tárolása után kiküldésre kerül a chipből. A CMOS érzékelők esetében a töltés mennyiség átalakítása közvetlenül a pixel érzékelőnél történik. Ebből következik, hogy a két szenzor felépítése, adottságai és korlátai is jelentősen eltérnek egymástól. Válaszidő szempontjából a CMOS érzékelők előnyben vannak a közvetlenebb kiolvasási technika miatt, és mivel az erősítő áramkör közvetlen a pixel mellett van alkalmazható alacsony feszültségű erősítő, mivel ez nem kivitelezhető egy CCD elvű áramkörnél, ezért jelentősebb energia felhasználással kell számolni. Zajosodás tekintetében a CCD szenzor előnyben van, mivel kevesebb aktív áramköri elem alkotja azt. Sebesség tekintetében a CMOS rendszer előnyben van a lapkákra integrált feldolgozó egységek miatt. Becsillanás mentesség (Antiblooming) annak a mértéke, hogy az adott szenzorról a szomszédos pixelek leolvasásakor mennyire hatnak egymásra. A CMOS immunis erre a problémára, hiszen minden egyes pixel egyesével értékelődik ki. Gyártási költségek tekintetében egyik technológia sem jelentősen költségesebb mint a másik. [5] A képek minőségét azonban nem csak az érzékelő, hanem maga az optikai is nagymértékben befolyásolhatja

A képek kinyerésére és a feldolgozó egység felé történő továbbításra a következő alternatívák adódnak:

- A legegyszerűbb megoldás egy rádiófrekvenciás távkamera (4-1. ábra), amivel több 10 méteres távolságban lévő vevőegységhez érkeznek be a jelek, amiket digitalizálni kell a számítógép számára, hogy az feldolgozható legyen. Ezen rendszerint a PAL illetve az NTSC szabványok szerinti analóg képet képesek biztosítani. A rendszer mellett szól a viszonylagos alacsony költsége, azonban az olcsóbb rendszerek nem biztosítanak megfelelő minőségű képet.
- Hasonló működési elvű kamerák az ún. IP kamerák. Alapvetően biztonságtechnikai kiegészítőként használják, képminőségük szintén az alacsonyabb kategóriák esetén nem megfelelő. (4-2. ábra)



4-1. ábra: Wireless Camera
[<http://www.swann.com/>]



4-2. ábra: WiFi Camera
[<http://www.alibaba.com/>]



4-3. ábra: Webkamera
[<http://www.logitech.com/>]

- Webkameráról (4-3. ábra) általánosságban elmondható, hogy az alacsonyabb árfekvésű szegmensben lassabb érzékelőket alkalmaznak, aminek a következménye, hogy mozgás hatására a kép elmosódik.
- A telefon kamerák szintén alternatívaként jelennek meg. A mai telefonok esetén már találkozni jó minőségű kamera modulokkal ellátott készülék, aminek a képminősége és sebessége elég lehet a projektben történő alkalmazáshoz.
- Fényképezőgépek esetén árszegmenstől függően a szenzorok minősége változó, azonban szinte minden esetben jobb kép készíthető, mint egy mobiltelefonnal, ennek alapvető oka a nagyobb érzékelő valamint a jobb optika.
- Videokamerák esetében a folyamatos mozgásrögzítés a cél, ezért azok gyors érzékelővel ellátott rendszerek, amelyek esetében a sebesség a szín, valamint a minőség visszaadásánál fontosabb lehet, alacsonyabb a felbontásuk, mint egy fényképezőgépé, de nagyobb sebességük.

Olyan eszközök esetén, amelyek nem képesek a vezeték nélküli kommunikációra valamilyen megoldást kell kidolgozni, hogy a képek továbbítódjanak a modellező szerver felé.

A mi választásunk: kezdeteknek egy Webkamerára esett olcsósága miatt, de lassúsága az elmosódás és a magas zajszint, korlátokat jelentett.

Az első prototípus elkészítése után a tapasztalt problémák számbavételével áttértünk egy iPhone típusú okostelefon alkalmazására.

4.2. Kommunikáció

A rendszerünk másik elvárt feladata, hogy olyan funkcionalitást biztosítson, amivel több egység képes hálózaton keresztül kommunikálni. Ennek révén kell megvalósítani az adatkommunikációt (képek, státusz információk, irányítási adatok), vagyis az adatcsomagok elküldését és fogadását a felek között. A következő lehetőségeket vettük szemügyre:

- *Helyi hálózat* a legnagyobb kompatibilitással bíró csatoló felület számítógépek között. Kábeles kapcsolat esetén a nagyobb átviteli sebesség az, amit kaphatunk, de sajnos akkor fel kell adnunk vele a rendszer egységeinek mobilitását, az autonóm mozgást.
- A *WIFI* ezzel szemben egy sokkal nagyobb szabadságot nyújtó rendszer. A rádióhullámok segítségével a lefedettséget biztosító területeken, szabadon mozoghatunk az eszközökkel kapcsolatvesztés nélkül. Ennek az ára, hogy az adatátviteli sávszélesség korlátozott, így aztán az átviendő adatok mennyiségére figyelni kell.

Class	Maximum permitted power		Range (m)
	(mW)	(dBm)	
Class 1	100	20	~100
Class 2	2.5	4	~10
Class 3	1	0	~5

4-4. ábra: Bluetooth osztályok
[Wikipedia]



4-5. ábra: A projecthez használt
Sparkfun BlueSMiRF modul

- *PAN (Personal Area Network) hálózatok* esetén gyakran használják a Bluetooth-t, mint kommunikációs csatornát. Ez egy aránylag lassú és bizonyos esetekben igen korlátozott kapcsolati lefedettséget biztosító rendszer. Nagy mennyiségű adat átvitelére alkalmatlan, így csak az egyéb státusz, illetve vezérlési adatok továbbítására használható fel. Három fő osztályba sorolhatóak (4-4. ábra).

Az egyik legkritikusabb területe a projektnek, hogy a megfelelő adatokat megfelelő minőségben és sebesség mellett kell eljuttatni az egyik féltől a másikig. A kezdeti tervek során egy csatorna használata volt az irányadó, ami a TCP/IP (*Transmission Control Protocol/Internet Protocol*) protokoll felhasználását tűzte ki célul. A csatorna adatátviteli sebességekvetelménye a következő: egy kép ~100 kB, 15 FPS (Frame pro second, kép per másodperc) mellett 1500 KB/sec. Ezt egy jó kapcsolatú WIFI hálózat is képes kielégíteni. A képi információk mellett egyéb adatok továbbítása is szükséges azonban ezek mennyisége elhanyagolható a képi forgalom mellett. Azonban hogy a vezérlési adatok és a kép adatok ne ütközzenek, illetve ne tartsák fel egymást, külön csatornák kialakítása szükséges. Ezen csatornákat csomagszinten a TCP/IP protokoll képes kezelni ezzel a programozás során nem kell foglalkozni.

A továbbfejlesztés részeként áttértünk az iPhone felhasználásra, aminek sajnálatos hátránya, hogy Bluetoothos kapcsolat kialakítására csak bizonyos gyártók, bizonyos termékeit engedélyezi az Apple. Fizikai kapcsolat kialakítására a csatlakozó soron keresztül, dokumentáció hiányában nem volt lehetőségünk. Ezért egy áthidaló megoldásként egy Bluetooth kommunikációs modul beszerzésre került és azt illesztettük a mikrokontrollerhez. Egy Class1-es Bluetooth modul sugárzási lefedettsége megegyezik egy WIFI-s eszköz hatókörével.

A felhasználható technológiák közül végül a WIFI illetve a Bluetooth mellett döntöttünk. Ennek oka, hogy a robotnak mindenképpen vezeték nélküli kommunikációt kell tudnia kezelni, és a fent említett kihívásokra ezzel a két technológiával képesek vagyunk megoldást nyújtani.

4.3. Vezérlés

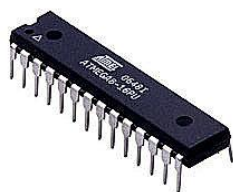
A vezérlőt fel kell készíteni arra is, hogy képes legyen kommunikálni valamilyen formában egyéb eszközökkel. A robotnak érkező adatcsomagokat a vezérlőnek fel kell dolgoznia és azokat konkrét vezérlési parancsokká kell alakítania. Ehhez egyéb eszközök integrációja is szükséges. Ilyen a motorvezérlő, valamint valamilyen kommunikációs illesztő is. Fontos hogy a beérkező adatcsomagokról visszajelzést is adjon a kapcsolattartó modulnak. Ezt lehetőleg a beérkező parancsok valamilyen nyugtázásával te meg. A rendszer állapotáról (pl.: telep energiaszint) tudjon információt adni a felhasználóknak. Fontos biztonsági koncepció, hogy bármilyen hiba – kapcsolat vesztés, hibás adatfolyam – a robot leállítását kell eredményezze.

A feladatok megvalósításához mikrokontroller alkalmazása mellett döntöttünk. Ezen eszközök nagy előnye, hogy magukban egy számítógépet képeznek. Hasonlóan a nagy számítógépekhez a rendszerek is rendelkeznek kimeneti és bemeneti perifériákkal, memóriával, CPU-val (Central Process Unit, központi feldolgozó egység), esetlegesen FPU-val (Floating Point Unit, lebegőpontos számokat feldolgozó egység), valamint a legtöbb modell EEPROM (Electrically Erasable Programmable Read-Only Memory) segítségével képes tárolni adatokat a beírt programján kívül is amik energia vesztés estén is elérhetőek maradnak. Kommunikációs eszközei lehetnek SPI (Serial Peripheral Interface) valamint USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) modulok. Az SPI protokoll esetén egy gazda-szolga rendszert kell felkonfigurálni és így lehet az eszközök között adatokat cserélni. Mivel az SPI egy soros buszcsatló felület, több eszköz használatát is lehetővé teszi egyszerre. Többek közt GPS és Bluetooth modulok is támogatják ezt a protokollt. Az USART esetén egy-egy kapcsolat építhető csak ki. Ezzel a perifériával nagyon könnyen illeszthetővé válik egy eszköz a számítógép soros portjához.

Két fő mikrokontrollereket gyártó cég termékeit vettük számításba a rendszer készítése során:

- Atmel Corporation
- Microchip Technology Inc.

Az Atmel terméke az AVR (4-5. ábra) névvel ellátott mikrokontroller (továbbiakban μC). Ezt a terméket 1996-ban kezdték fejleszteni. Három fő csoportra osztották a rendszereket: Tiny, Mega, xMega. Csoportosítás alapjául a belső memória mérete, illetve magának a perifériáknak a száma szolgál. A μC minden órajelre egy utasítást hajt végre, ami igen gyors feldolgozási sebességet jelent. Az AVR esetén a teljes memóriaterület elérhető korlátozások nélkül.[6]



4-6. ábra: AVR



4-7. ábra: PIC

A Microchip által gyártott PIC (4-6. ábra) a mai felépítésüket 1993-ban nyerték el, amikor megjelent PIC16x84-es sorozat. Ennek a sorozatnak az volt az újítása, hogy a processzor EEPROM területének törlését már a processzor vezérelhette [7]. E típusok esetén egy kisebb utasítás számú Assembly készletet kell megtanulni, valamint fizetős verziókban C nyelven is lehet fejleszteni a platformra. Az utasítás végrehajtási idő jól közelíthető az alaputasítás időigényének kétszeresével. A teljes

memóriaterület csak ún. bankolás³ segítségével érhető el, ami nagyobb odafigyelést igényel a programozó részéről.

Választásunk azért esett az Atmel termékére, mert a jól dokumentált, részletes regiszterleírásokat tartalmazó termékdokumentációit kiegészítik az adott periféria programozási példáival is. A programozó elektronika olcsón és egyszerűen megépíthető, és a szimulátor – ami a fejlesztőkörnyezet része – egyszerűen használható. A programozási feladatokat alapvetően C nyelven lehet elvégezni, amit a termék fejlesztője natívan támogat és folyamatosan fejleszt az assembly-s kódoláshoz képesti minél kisebb sebesség különbség elérése céljából. A mikrokontrollerre nem fog számításkritikus feladat jutni, ezért a már ismert fejlesztői (C) nyelv fontosabb a fejlesztés szempontjából.

Kiterjedt és segítőkész szakmai fórumok állnak a fejlesztő rendelkezésére mind két gyártó termékéhez kapcsolódóan. Figyelembe véve a fejlesztéshez szükséges összes eszköz költségét, szintén az AVR került előnyösebb pozícióba.

4.4. Irányítás

A felhasználó számára biztosítani kell a rendszer irányítását. Az irányítással kapcsolatos adatok beolvasását és átalakítását kell elvégezni, hogy a robot vezérlője azt képes legyen feldolgozni. A kommunikációért felelős egységekkel kapcsolatot kell teremteni, hogy az adatokat továbbítani lehessen a robotnak.

A billentyűzetes adatok feldolgozása igen egyszerű eljárás. Így azonban nem vagyunk képesek megfelelő pontossággal irányítani a rendszert, a mozgás így nem lenne kellően finoman szabályozott. Mivel minden rendszer rendelkezik ezzel a perifériával azért a kezelése nem elhagyható.

Egy játékvezérlő (4-7. ábra) használata sok szempontból tűnik jó választásnak. Ezek az eszközök sok esetben rendelkeznek „analog” botkormánnyal. Nehézség lehet magának a kontroller adatainak kinyerése, ami nem támogatott a .NET-es keretrendszerben DirectX es bővítmények nélkül.

Az egér használata nem jó elgondolás, mert folyamatos egérmozgatás lenne szükséges, bizonyos részterületeken használható lenne, de összességében az irányításra nem alkalmas. Végül a billentyűzetes és játékvezérlős irányítás megvalósítása mellett döntöttünk. Billentyűzet minden gépnél található, viszont egy gamepad analóg botjával nagyon precíz irányítás érhető el.



4-8. ábra: Joystick, Gamepad

³ A teljes memória terület csak részenként érhető el. A címzésnél e memória részleteket kell megcímezni és a blokkon belül már közvetlenül címezhető a memória.

4.5. Kapcsolattartás a felhasználóval

A rendszer és a felhasználó közötti kapcsolattartás megvalósítása érdekében egy olyan felhasználó felületet kell elkészíteni, aminek a használata egyszerű. Ez a feladatrészt foglalkozik a felhasználó utasításainak a feldolgozásával, amik alapján a rendszer vezérlése történik. Biztosítani kell bizonyos visszajelzést is a felhasználó számára. A rendszer logolásának megjelenítésére, követésére is lehetőséget kell biztosítani.

4.6. Képek előfeldolgozása

A képek előzetes feldolgozása biztosítja a szoftver modellépítési funkciójához szükséges információkat. A robot oldali képfeldolgozás csupán a képek kiolvasása és azok átadása a kommunikációs modulnak. A szerver oldalon történik a kapott képek további hasznosítása. A háromdimenziós modell megalkotásához először jellegzetes pontokat kell keresnünk a kameraképeken, és ezeket beazonosítani az egymást követő képkockákon. Erre az úgynevezett pontkövetésre többféle módszer ismert. A megfelelő követési elv kiválasztásánál fontos szerepet játszik a módszer megbízhatósága és gyorsasága. Mivel mindkét szempontból tökéletes módszer nem létezik, a megvalósítás során kompromisszumot kellett kötnünk, és inkább a megbízható, ám lassabb algoritmust alkalmazzuk.

4.7. Pontok elhelyezése a térben

A jellegzetes pontpárok megtalálása csak az első lépést jelenti a modell felépítése során. Ha rendelkezünk a különböző kamerapozíciókból készült képeken lévő néhány pont adataival, a megfigyelhető elmozdulások alapján, megfelelő matematikai módszerekkel kiszámíthatjuk a képpontok térbeli helyzetét. A folyamat során először a kamerák egymáshoz viszonyított pozícióját számoljuk ki, melynek végrehajtására több módszer is ismert, a gyakorlatban pedig ezek kombinációját alkalmazzuk. Ezután a képpontokat a kamerák ismeretével visszavetítjük a térbe, így kapjuk meg háromdimenziós helyzetüket. A pontos helymeghatározáshoz szükséges a képeket készítő kamera belső paramétereit leíró mátrix ismerete, amelyet lehetőségeinkhez mérten önkalibrációval, vagy a kamera előzetes kalibrációjával kaphatunk meg. A számítások pontosságát mindvégig hátráltatják a képkockákon megjelenő különféle zajok, melyek kiküszöbölésére is figyelmet kell fordítanunk, hogy minél pontosabb modellt kapjunk végeredményül.

4.8. Modellezés és textúrázás

A program fő célja, hogy a beérkező képek alapján egy olyan háromdimenziós modellt építsen fel, amely tükrözi a valós környezet felépítését és arányait. A térben lebegő pontfelhő helyett jobb vizualizációs megoldást nyújt, ha a pontok alapján elkészítjük a fényképezett jelenet testhálóját, majd a képkockák adataival textúrázzuk azt. A testháló felépítése igen sarkalatos kérdés, mivel gyakorlatilag ez fogja pótolni a modell azon részeit, amelyeken nem találtunk jellegzetes pontokat az előfeldolgozás során, így térbeli információt nem sikerült kinyernünk róluk. Gyakorlatban a legmegfelelőbb módszer erre a pontok közötti Delaunay-háromszögekkel felépített háló későbbi finomítása bizonyos paraméterek alapján. Végző lépésként a háló háromszögeihez tartozó megfelelő részletet kivágjuk a kamera által készített képekből, és ráillesztjük a háromszögre.

4.9. A modell megjelenítése

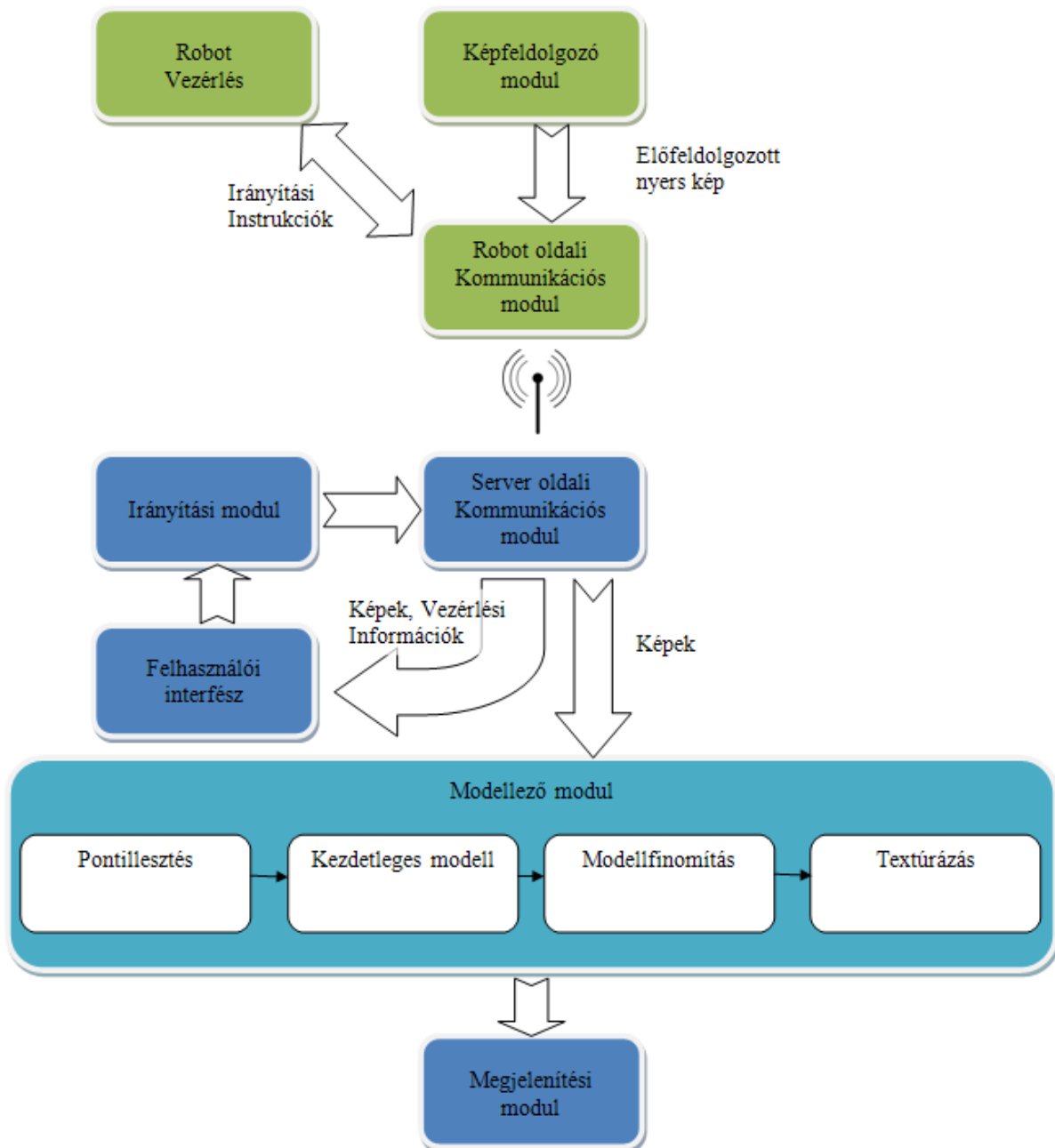
A felépített háromdimenziós modellt képezi le a felhasználó számára megtekinthető kétdimenziós formába. A felépített környezet billentyűzet illetve egér segítségével mozgatható, bejárható. Alapvetően a modellezés során felhasznált fejlesztőkörnyezet (MATLAB) jól használható megoldásokat kínál erre a feladatra, mivel beépítetten kezeli az OpenGL által elérhető háromdimenziós vizualizációt támogató funkciókat.

4.10. Publikus adatok közzététele

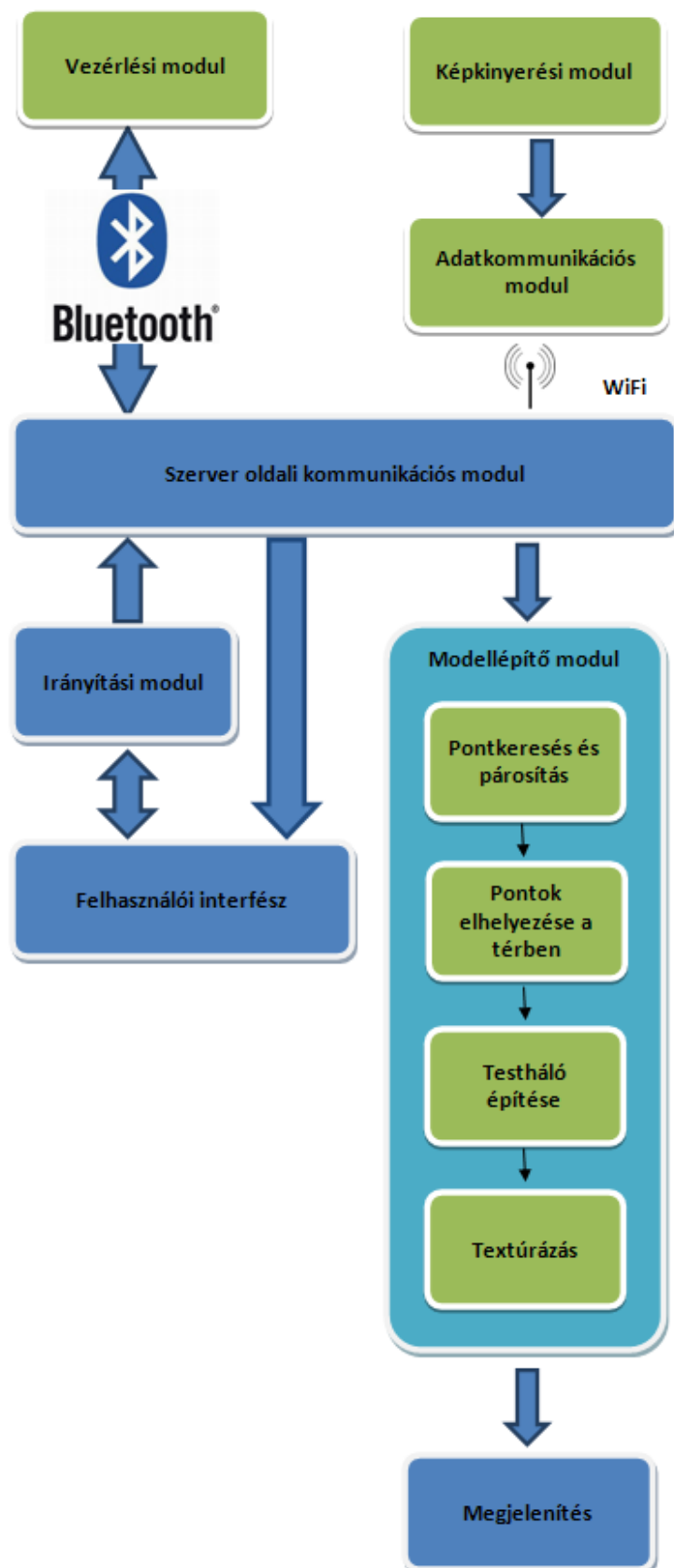
Feladat egy olyan részrendszer megalkotása, aminek a segítségével nyomon követhető egyszerűen a robot állapot, esetlegesen az, hogy mit is lát maga előtt.

Ennek legkézenfekvőbb alternatívája, hogy egy egyszerű webszerver segítségével megjelenítsünk bizonyos adatokat, amik megtekintése nem igényel specializált programokat. Így bármilyen böngészővel, rendszerkezelő eszközzel képesek vagyunk adatokat közölni a rendszer állapotairól. Ez a részrendszer publikus adatok közzétételét szolgálja, nem a felhasználói interfész részeként funkcionál.

5. Rendszer modul terve kronológia sorrendben



5-1. ábra: Előzetes rendszerterv



5-2. ábra: Az újragondolt rendszer terve

6. A rendszer bemutatása

6.1. Robot leírása, ismertetése

A robot egység két fő részre bontható: hordozó és a rajta elhelyezett képi adatok beolvasásért, valamint azok továbbításáért felelős egységre. Kezdetben az elgondolás az volt, hogy egy notebookot és az ahhoz kapcsolt kamera párost elhelyezve a hordozón fogjuk megoldani mind a képek előfeldolgozását, mind azok továbbítását a távolban lévő gépnek. A notebookhoz egy soros porton keresztül csatlakoztatjuk a robot-vezérlő elektronikáját. A robot építése során azonban a méretproblémák miatt egy alternatív megoldás megvalósítására adtuk a fejünket; mégpedig egy okostelefon használata vetődött fel. Előnye hogy nem igényel nagyméretű hordozót, rendelkezik a kommunikációhoz szükséges portokkal és kamerával is.

6.2. A szerkezet műszaki paraméterei

A robot (6–1. ábra) 20x25 cm alapterületű. A tápellátásról 6 darab AA méretű ceruzaelem gondoskodik, amik így 9V bemeneti feszültséget állítanak elő az elektronika számára. A két motor meghajtása közvetlenül a telepekről történik. Az elektronika számára egy speciális feszültség konverter állítja elő az üzemi 5V feszültséget. A robot súlya elemekkel együtt: ~1kg

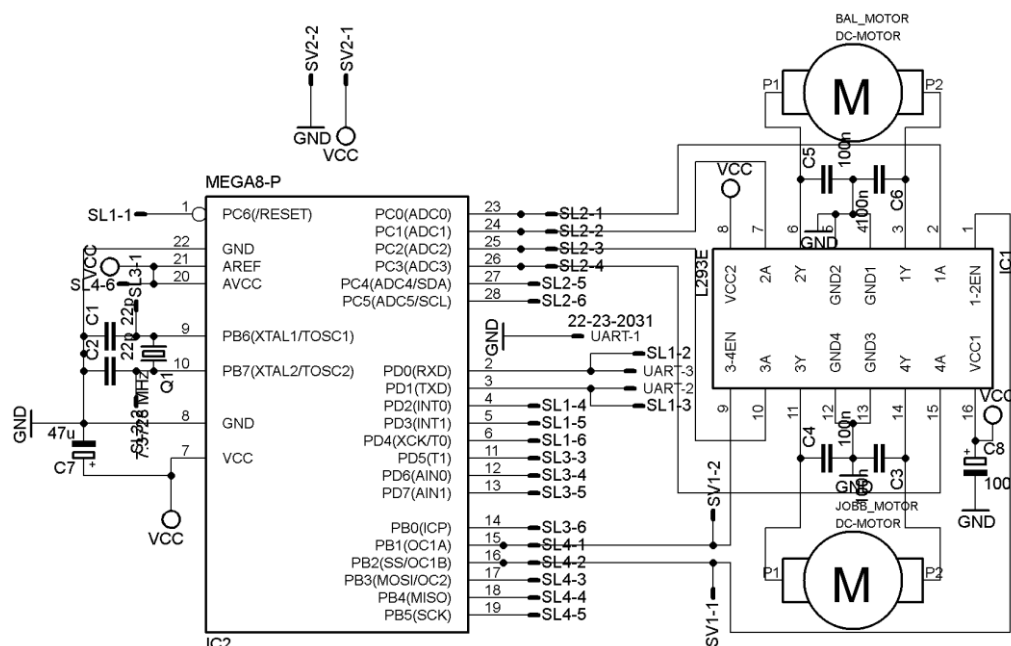
6.3. A robotvezérlő elektronika

6.3.1. Tápellátás

A tápellátás alapproblémája hogy a beérkező 9V feszültséget az elektronika számára 5V üzemi feszültségre kell csökkenteni. Lehetséges megvalósítás le lehetne egy ellenállás alapú feszültségcsökkentés. Ezen áramkörökből lehet fix kimeneti feszültségűt kapni (5, 12, 15V). Azonban ezek a chippek alacsony áramellátást képesek biztosítani. A másik hátrányuk a relatív alacsony hatásfok. Ennek oka, hogy az elektronika ellenállás segítségével állítja be a kimeneti feszültséget, ami azt jelenti, hogy azokon átfolyó áram hőt termel és ez jelentős energia veszteséggel jár. A hőtermelés miatt külön hűtőborda használata is szükségessé válhat. Ennek a módszernek egy egyszerűsített



6-1. ábra: Robot



6-2. ábra: Vezérlő kapcsolási rajza

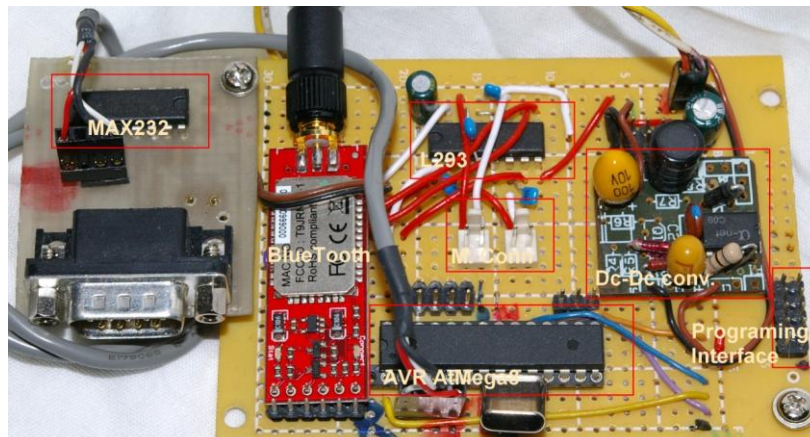
alternatívája a Zener dióda és egy áram korlátozó ellenállás használata. A Zener dióda lényege, hogy a hagyományos dióda letörési feszültségét állítják be pontosan egy kijelölt feszültségértékre. Így érhető az el, hogy csak bizonyos feszültséget engedjen át helyes bekötés esetén. Ha egy Zener diódát fordítva kötünk be az áramkörbe, akkor úgy viselkedik, mint egy hagyományos dióda, nyitófeszültsége fordított irányba 0,3–0,7V közé esik típustól függően. Ennek a módszernek is az a hátránya, hogy a diódán és az ellenálláson is esik feszültség és az átfolyó áram hő termel a komponenseken.

Az előző megoldásnál sokkal hatékonyabb megoldások léteznek, a kapcsoló üzemi tápegység speciális változatai. A lényeg, hogy ha kellően gyorsan fel-le kapcsolgatjuk az egyenfeszültséget, akkor azt a jelet kiintegrálva előállítható a kimeneten egy stabil egyenfeszültség. Ezeket Step-Down illetve Buck konvertereknek is nevezik [8]. Ennek előfeltétel egy olyan integráló, energiatároló egység, mint a kondenzátor. Tipikusan 30KHz – 10MHz-es tartományban mozognak e típusú áramkörök működési frekvenciái. A kimenetet egy L-C taggal megszűrve az elektronika tüskéktől mentes kimeneti feszültség szintet állít be. A manapság legfejlettebb áramkörök a 94%-os⁴ hatékonyságot is képesek elérni. Ezeknek a rendszereknek másik nagy előnye, hogy gyakorlatilag hőtermelés nélkül képesek ellátni a feladatukat, akár 16 amperes áramerősség mellett is. Az alkatrészek költsége magasabb az ellenállásos feszültség korlátozóknál, valamint fizikailag is több kiegészítő alkatrészre van szükség. Azonban a most említett negatív tulajdonságok sokkal kevésbé fontosak egy akkumulátorról működtetett alkalmazás esetén, ahol a fogyasztás az egyik leglényegesebb pont a kritériumok közül.

6.3.2. Mikrokontroller

A hordozóhoz felhasznált elektronika „lelke” egy mikrokontroller (μC). A mikrokontrollerek speciális programozható mikroszámítógépek, amik rendelkeznek felparaméterezhető perifériákkal, operatív tárral programozható I/O portokkal. Ezeket a mikroszámítógépet alapvetően beágyazott rendszerek vezérlésére használják. Ehhez a munkához az Atmel® cég AVR® típusú processzoraira esett a

⁴MAXIM MAX1951A



6-3. ábra Vezérlő elektronika

választás. A processzor családok közül a Mega sorozat 8-as szériáját választottuk (6–3. ábra). Ez az egység egy CMOS alapú kisfeszültségű 8 bites vezérlő. Architektúrájának alapját egy RISC processzor mag jelenti. Általános számítási kapacitása 1 MIPS per MHz. Ez azt jelenti, hogy egy óraciklus alatt egy utasítást képes végrehajtani.

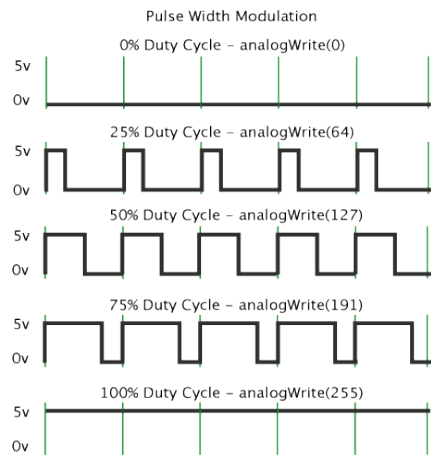
A beépített perifériákat regisztereken keresztül lehet fel programozni. Egyes részegységek felprogramozhatók úgy, hogy a kiadott műveletvégzés után ún. interruptok hívódjanak meg. Ilyen funkció lehet egy konvertálás, mivel egy A/D konverzió jelentős ideig tart az egyszerű műveletekhez képest, ezért jó lenne, ha nem kéne megvárni a folyamat végét, hanem addig folytatni lehetne a fő programot. Ha végzett a művelettel a konverter akkor majd meghív egy interruptot és a megfelelő programrész lefut. A szoftveres interruptokba írt algoritmusok segítségével célszerű az adott művelet eredményét feldolgozni. A perifériák a fő programfolyamtól függetlenül párhuzamosan képesek végezni a dolgukat, így a feladat kiadása után csak az eredmény megvárása a feladat. Ez hasonlít a .NET-os környezet eseménykezelésére, mint amikor egy eljárás feliratkozik egy másik eseményre, hogy az hívódjon meg az esemény bekövetkeztekor.

A fejlesztés során mi is az interruptok használata mellett döntöttünk. Minden AVR típusú μC rendelkezik belső órajel generátorral, ami programozható. Felprogramozás után controllernek másra nincs is szüksége csak tápellátásra.

Az eszköz rendelkezik belső órajel generátorral. Azonban ez az áramkör igen érzékeny a tápfeszültség zavarokra. Sajnos nem képes kellően stabil órajelet generálni az olyan érzékeny időzítést igénylő perifériáknak, mint a kommunikációhoz felhasznált USART vezérlő.

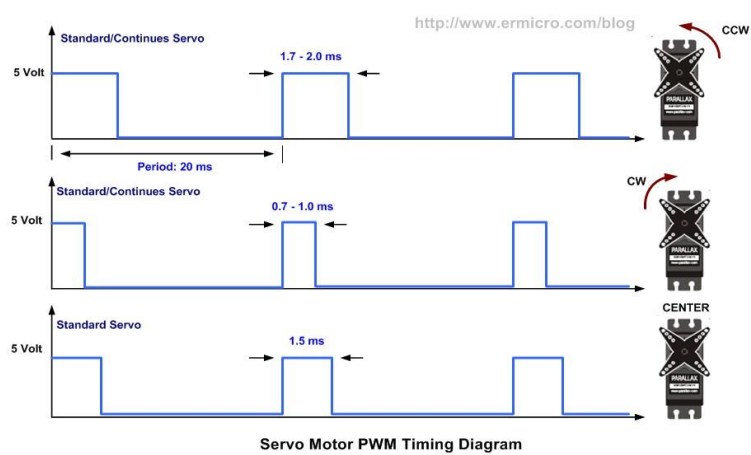
6.3.3. Kapcsolattartás az elektronikával

A kommunikációhoz felhasznált periféria az USART modul. Ehhez a perifériához rendkívül sokféle hardver illeszthető, ami ezt a protokollt képes implementálni. Tipikusan felhasználási területek: soros port, USB, Bluetooth. A projekthez egy egyszerű soros port illesztő használata mellett döntöttünk, annak egyszerű implementálhatósága és alacsony költségei miatt. Ez a modul egy egyszerű feszültség szintillesztő, ami a μC 0–5V-os üzemi feszültségét -12–12V-os soros porti feszültség szintekké konvertálja [9]. Kezdetben egy egyszerű MAX232 soros porti illesztőt használtunk, de ugyan ilyen jó megoldás lett volna USB-USART konverter, ami egy virtuális soros portot hoz létre a számítógépen, aminek a kezelése programozói oldalról megegyezik a hagyományos szintillesztős megoldással. Programozási szempontból az olyan illesztők használata esetén nincs különbség, amik soros portként



6-4. ábra PWM jelek

[<http://www.arduino.cc/en/Tutorial/PWM>]



6-5. ábra szervo motor vezérlőjel

[<http://www.ermicro.com/blog>]

szerepelnek a rendszerben. Tipikus példa a Bluetooth modulok. Ezért is volt egyszerű dolgunk lecserélni a vezetékes megoldást egy ilyen modulra. Egy Class 1 egységgel az áthidalható távolság 100 méter lehet. A .NET-es környezetben egy soros portot kezelő könyvtár felhasználásával azonnal küldhetőek információk a porton keresztül.

A μC esetében speciális regiszterek jelzik a beérkező információkat. Ezen regiszterek képesek szoftveres interruptokat kiváltani, ami után speciális eljárás hívódik meg és lefutnak a beprogramozott algoritmusok. A beérkező adatokat egy checksum - ellenőrző kód - védi a hibás irányítási adatok végrehajtódásától. Amennyiben hibás adatot kap a rendszer, azonnal leállítja a hajtást. Az alkalmazott csomagfelépítés a következő: (KÓD, BAL MOTORVEZÉRLÉSI ÉRTÉK, JOBB MOTORVEZÉRLÉSI ÉRTÉK, CHECKSUM). Az elektronikába bekerül egy olyan eljárás, ami bizonyos időközönként adatokat kér a gazda rendszertől (polling), hogy megbizonyosodjon a kapcsolat meglétéről. Ha a kapcsolat megszakadt, akkor a rendszert le kell állítani, és hibajelzést küldeni a felhasználónak, hogy értesüljön róla.

A felhasznált kapcsolati beállítások a következők: 9600 BaudRate⁵, 8 adatbit, nincs paritás bit, 1 StartStopbit, nincs kézfogás (*handshake*).

6.3.4. Robot vezérlése

Az elektronika fel lett készítve mind szervomotorok és hagyományos egyenáramú motorok kezelésére. DC motorok esetén egyszerű a vezérlés, hiszen a motoroknak nem kell speciális jelet biztosítani, csak a megfelelő kitöltési tényezőt kell beállítani és a motorvezérlőnek megadni a megfelelő irányt. A kitöltési tényezőt a μC PWM [10] (6-4. ábra) jel generátorának segítségével lehet beállítani. A megfelelő regiszterben be kell állítani a kíván értéket és a rendszer a következő ciklusban már a beállított paraméterek szerint képezi a jelet a motorvezérlőnek. A PWM jel integrálásának segítségével egy feszültségszintet kapunk, ami a motor esetében pl. a fordulatszámot és nyomatékot, de egy LED (Light Emitting Diode) esetében a fényerőt határozza meg.

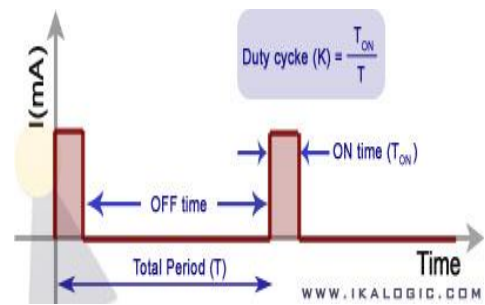
⁵ Másodpercenkénti szimbólum átvitel sebessége



6-6. ábra Sharp GP2D12



6-7. ábra Maxbotix LV-EZ0

6-8. ábra Működési ciklus
[<http://www.ikalodic.com>]

A μ C-be épített 16 bites időzítő, számláló konfigurálása után a periféria két output lábán jelennek meg az előállított jelek a jobb, illetve a bal motor számára. A felhasznált motorvezérlő IC chip (L293) miatt az irányokat is meg kell tudni adni. Ez újabb 4 port vezérlését követeli meg.

A szervo motorok vezérlése jelentősen eltér a fentebb említett DC motorok vezérlésétől. Oda a 6–4-es ábrán jelzett típusú vezérlés szükséges, aminek az előállítása igen egyszerű. Azonban egy szervo motornak speciális időzített jelsorozat kell (6–5. ábra), hogy azzal beállíthassuk a forgási irányt és a tengelyek kitérési szögét, valamint a folyamatos forgásra képes motorok esetén a tengely forgatási nyomatékát.

A szervo vezérlés megvalósítására azért volt szükségünk, mert kezdetben speciális folyamatos hajtású szervók használata mellett döntöttünk. A későbbiekben ezt elvetettük és áttértünk a hagyományos, könnyen illeszthető és beszerezhető DC motorok használatára. Azonban ekkor a következő probléma ütköztünk a két motor karakterisztikája különböző és ezt kezelni kell és szinkronba kell azokat hozni.

6.3.5. Emulátor

A programok fejlesztése szempontjából előnyös, hogy rendelkezünk egy emulátorral, amivel a fejlesztési munkálatokhoz nem szükséges maga a hardver, így mind tesztelési mind fejlesztési feladatok egyszerűsödnek. A felületen a megfelelő soros port kiválasztása után csatlakozhatunk a robot oldali szoftverkörnyezethez. A vezérléshez küldött adatok vizuális megjelenítésével egyszerűsödött a kódolási eljárások fejlesztése. Ahhoz, hogy tudjunk csatlakoztatni másik soros porti egységhez egy gépen két lehetőség van: Ha alrendszer fel van szerelve legalább kettő porttal, akkor azokat szembekötve egymással képesek lesznek kommunikálni, illetve virtuális soros port telepítése a fejlesztői gépre. Mivel mi nem rendelkezünk olyan géppel a fejlesztés során, amin lett volna fizikailag két csatlakozás a virtuális megoldás mellett döntöttünk.

6.3.6. A robot biztonsági megoldása

A biztonsági megoldások több szempont szerint vizsgálhatjuk meg. Fizikális védelem valamit preventív jellegű, megelőző berendezések. Fizikai védelem a rendszer robusztusságának növelésével érhető el, ilyen szempontból a Husky (3–3. ábra) hordozó egy igen masszív felépítmény. A preventív jellegű berendezések, eljárások lehetnek pl.: távolság érzékelők vagy összeköttetés figyelése a szerverrel.

Távolságérzékelésre több módszer is adódik: ultrahangos vagy infravörös tartományban működő érzékelők.

- Infra vörös szenzorok (6–6. ábra) működési elve a következő: van egy aktív infravörös fényt kibocsátó fényforrás valamint, egy az azonos hullámhosszra érzékeny passzív fényérzékeny dióda. Az objektum távolságát a fényimpulzus kibocsátása és annak visszaérkezése közt eltelt ΔT -idő, valamint a fény terjedési sebességének hányadosa határozza meg. A minimális érzékelhető ΔT idő (6–8. ábra) a kibocsátott impulzus szélességének (T_{on}) lefutó éle utáni első időpillanat, a maximum pedig a periódusok felfutó élei között eltelt idő (Off Time).
- Az ultrahangos szenzorok (6–7. ábra) működési elve megegyezik az infra szenzorokéval, azonban azok kevésbé érzékenyek az érzékelendő felület tulajdonságaira, pl. ablaküveg vagy fekete gumiabroncs, amelyek esetén az infra szenzorok hibás értékeket olvashatnának be. Ezért sok helyen az ipari környezetben használják fel ezeket, mert jobb tulajdonságokkal rendelkezik poros, koszos környezetben is. [11]

A szenzorok lehetnek analóg illetve digitális kimenetűek is. Mi egy analóg kimenetű infra szenzort használunk fel. A szenzor kimenetét rákötjük a μC analóg komparátor bemenetére. Így tudjuk figyelni, hogy ha az előre meghatározott érték alá csökkenne az érzékelő kimenet, akkor a robotot az adott irányba nem engedjük tovább haladni.

6.4. Feladatok megvalósítása

6.4.1. Kommunikáció soros porton

A soros port megvalósítása a szoftverkörnyezetben soros port osztály esetében a megfelelő sebesség beállítások után, a soros port használatra kész is. Az osztálynál használt átvitelre vonatkozó beállítása a fentebb említettek megegyezik. A modulnál a *DataReceived* metódus került megvalósításra. Ebben a metódusban kiolvasásra kerül a μ C-től érkező üzenet és küldésre kerül a hálózaton felcsatlakozott felhasználónak, és egy logolásra szolgáló lista elembe kerül eltárolásra. A VisualStudio fejlesztő környezetben speciálisan lehet csak a beérkezett adatokat kiolvasni, mert különben szálhivatkozási hibát dob. Ennek a problémának a megoldására egy *Invoke* metóduson belül egy új *delegate* osztály létrehozásával az olvasás máris lehetséges. Adatok Küldése a μ C felé az osztály *Write* metódusának meghívásával lehetséges.

6.4.2. Hálózati kommunikáció

A kommunikáció alapjául az operációs rendszer hálózat kezelését használtuk fel. A hálózatra azért esett a választás, mert a megfelelő osztályok példányosítása után TCP/IP protokollt használhatjuk. Ennek felhasználásával egy igen széleskörűen alkalmazott közegen leszünk képesek kommunikációs csatornát felépíteni. Akár mobil hálózatok is felhasználhatóvá válnak. Először vezeték hálózatokat használtunk, majd később áttértünk WIFI alapú közeg használatára.

Kezdetben a hálózati kommunikációs modul egy univerzális megoldás volt, amit a szerver és a robot oldalon is alkalmaztunk. Az itt bemutatott modul azonban már csak a szerver oldalon használatos mivel a fejlesztések előrehaladtával a mobil egységre egy iPhone típusú okostelefon került, aminek a hálózat kezelési metrikája eltér a .NET-es környezetben használtaktól.

A .NET-es környezetben a kommunikációs modul egy külön osztálykönyvtárként került megvalósításra, amit a használathoz a referenciák közé be kell állítani majd példányosítani. Az osztályok közti kommunikációért ez un. *EventHandler*ek felelnek, ezek azok az egyedi eljárások, amin keresztül az adott eseményre feliratkozott metódusoknak tudunk üzenetet küldeni és átadni paramétereket. Ilyenek például a *DataReceived*, ez akkor hívódik meg, ha a csatornán egy másik féltől érkező adatok kiolvasásra kerültek. A hálózati kapcsolat felépítése szempontjából fontos, hogy a háttérben folyamatosan figyeljük az adott kapcsolathoz beállított portot. Ehhez a szálak (threadek) alkalmazása elkerülhetetlen. Szálak alkalmazásával megvalósítható, hogy a fő program folyamattól függetlenül történjenek események vagy fussanak egyéb folyamatok a rendszerben, mint például egy port figyelése.

Az elkészített modul példányosításakor definiálni kell a csatornán átküldeni kívánt adatok típusát; képek küldésére akarjuk felhasználni vagy egyéb irányítási információk továbbítására. Ez után célszerű egyből feliratkozni a modul eseménykezelőire, mint például a *Connected* aminek a segítségével értesítést kapunk ha csatlakoztak a rendszerünkhöz. A létrehozott saját osztály példány, *StartServer()* metódusát meghívva indul el az hálózati portnak a figyelése. A háttérben ekkor egy szálon folyamatosan egy ciklus fut, ami figyeli, hogy van e várakozó kapcsolat, amíg nincs akkor a szálat alvó állapotba teszi 1 századmásodpercre. Azért szükséges ez az eljárás, mert a rendszerben felemésztené az erőforrásokat a folyamatos lekérdezése a *pending*⁶ állapotnak. Várakozó kapcsolat

⁶ Várakozó

észlelése esetén a kapcsolat adatai alapján egy *NetworkStream*⁷ példányosítására kerül sor. A meglévő *Stream* alapján már indítható az átvitelt figyelő szál, ami felel a beérkező adatok kiolvasásáért a *Stream*-ről és az adatok visszaadásáért. A szál figyel, hogy a *Stream*-en van-e kiolvasható adat, ha nincs, akkor a szálát alvó állapotba helyezzük 1 századmásodpercre, mint a beérkező kapcsolatok figyelését végző szál esetében.

A szálak esetében odafigyelést igényel, hogy a figyelést végző szálakat megfelelően le kell tudni állítani. Ezek a szálak olyan ciklust futtatnak, amiknek a leállításához egy osztályváltozó beállítása szükséges: *isStarted* logikai változó. Ennek a változónak a beállítása a *Stop* metódus meghívására átváltódik hamisra így a ciklusok kilépési feltételei teljesülnek és a szálak leállnak. Az is nagy odafigyelést igényel, hogy bezáráskor egy szál sem maradhat futásban, mert az megakadályozza a program bezárulását, ezért a program bezárásakor mindenképp meg kell hívni a szál *Join* metódusát. Így a fő programszál leállítása addig nem történik meg, amíg a háttérben futnak kapcsolódó szálak.

A modul tartalmaz egy *Connect* eljárást, amivel a már beállított szerverhez van lehetőség kapcsolódni az IP cím illetve a port megadása után. A *SendData* eljárás segítségével lehet az adott *Stream*-re elhelyezni az adatokat. A *NetworkStream*-re helyezett adatokat a rendszer automatikusan elküldi a másik félnek, ezzel a programozónak már nem kell foglalkoznia.

Az iPhone esetében a hálózat kezelés alapjaiban nem sokban különbözik a .NET-es féle megoldástól. *CFWriteStreamRef* objektum az, ami hálózati *Stream* referenciáját eltárolja. Amikor adatot akarunk küldeni akkor ez a referencia objektum, amire hivatkozni kell. Ezután egy *socket* kialakítása következik a hosttal, ezzel azt mondjuk meg, hogy a *Stream* kivel áll kapcsolatban, és hova kell csatlakoznia. Néhány kapcsolati tulajdonság beállítása után a *Stream* megnyitható és ezzel készen áll a kommunikációra. Adatok küldésére a *CFWriteStreamWrite* eljárás használható, aminek paraméterei közt az előbb létrehozott *NetworkStream*-et illetve a továbbítandó adatokat kell megadni. Az adatok küldése előtt érdemes lekérdezni a csatorna állapotát, hogy az képes-e fogadni a csomagokat. Ezt a *CFWriteStreamGetStatus* segítségével tehetjük meg, ami egy enum-ként adja vissza a kapcsolati státuszt.

6.4.3. Képkinyerés kamerából

A fejlesztések kezdetekor két számítógép hálózati kapcsolata jelentette a rendszer összetevőit, ezért egy olyan eljárást kellett találni, amivel a számítógépen sikeresen lehessen kiolvasni a képeket. Ehhez egy külön osztály készült, ami az *EmguCV wrapper* segítségével olvassa ki a webkamerától érkező képeket. A kiolvasott képek *Bitmap* osztály formájában továbbítja azoknak az osztályoknak, amelyek feliratkoztak az eseménykezelőre. A képkinyerő modul tartalmaz még egy megjelenítési részt is, ahol a kiolvasott képek látszanak, ezáltal könnyítve meg a későbbi tesztelést. A modult referenciaként megadva, majd példányosítva, bármilyen egyéb rendszerbe beilleszthetővé válik.

A képkioolvasást megvalósító osztályba került még elhelyezésre egy olyan funkció, aminek segítségével elmenthetővé válnak a kamera által készített és kiolvasott képek. Ezt később fel is használhatjuk például a publikus adatok közzétételénél, hiszen a modul által elmentett képeket le lehet kérni az integrált webservertől.

⁷ Stream: egy előre meg nem határozott hosszúságú adatfolyam

Mivel nem adódott lehetőségünk olyan méretű hordozó építésére, ami egy teljes funkcionális számítógépet vagy laptopot elbírna, ezért alternatív megoldások számbavételére volt szükség. Alternatívaként felmerült egy WIFI webkamera beszerzése, azonban a webkamerák általános képalkotási minőségét is felmérve, nem felelt volna meg a minőségi igényeknek, ezért döntöttünk egy okostelefon felhasználása mellett.

A rendszer fejlesztése végül iPhone használatával folytatódott. Azonban a képkinyerés a telefon esetében más megközelítést igényel, mint az *EmguCV* könyvtár használata esetén. Az iPhone 4-es telefon kamerája autófókuszos. Ez azért probléma, mert az képfeldolgozási eljárások nem képesek az olyan képsorozatot kezelni, aminek minden képe más fókuszponttal készült el. Ezért ezt a funkciót ki kellett kapcsolni. Az *Objective-C* esetében az *AVFoundation* programkönyvtár az, ami a telefon média bemeneteinek kezelésére szolgáló függvényeket és objektumokat leírja. Ahhoz hogy a kamerából kinyerjük a képeket egy folyamatirányító objektumot kell felhasználni. Az *AVCaptureSession* esetén az adott folyamatra vonatkozó bemeneti eszközöket kell megadni. Nekünk csak a kamera képe kell, ezért elég megadni a videó típusú eszközöket a rendszerből, majd beállítani, hogy a fényképező optika fókusza ne változzon a képek készítése folyamán. Kimenatként érdemes megadni valamilyen képmegjelenítő modult, hogy látható legyen az elkészített kép. Ezek elmentése után a *Sessiont* el kell indítani és már is kinyerhetőek a képek a kamerából. A kapott képeket ezután már továbbíthatjuk a hálózaton a szervernek modellépítés céljából.

6.4.4. Irányítás

Az irányítási feladat egy igen érdekes feladatkör volt a számunkra. Egy robot manuális irányítása esetén egy olyan rendszer használata, ami sokkal jobban közelíti az analóg irányítási metrikákat, jelentősen finomabb mozgatót tesz lehetővé a digitálisnál, billentyűzetes vezérlésnél. Ezért egy speciális osztály beépítését alkalmazva illesztettünk egy botkormányt a rendszerhez. A *SlimDX* modul egy nyílt forráskódú *DirecX wrapper*, amit a .NET es alkalmazásokhoz fejlesztenek. Ezzel a modullal sikerült elérni a *direct input API*-t ami felelős a játékvezérlők kezeléséért a Windowsos rendszer alatt. A botkormány illesztése azért is ajánlatos mert, mer így a gombok felprogramozásával sok funkciót kézre lehet szabni.

A játékvezérlők tengely állapotának leolvasása soros kóddal megtehető. Alapvetően a *polling* módszerét használva, bizonyos időközönként kiolvassuk a botkormány pozícióját és egy eseményen keresztül értesítjük a megfelelő metódusokat és átadjuk az állapot információt. Az értékek kiolvasása után azonban ezt az információt el kell tudni küldeni a robotnak is. A roboton egy 8 bites processzor helyezkedik el, ami nem minden esetben képes kezelni egy 16 bites számértéket, valamint a sebesség szabályozáshoz használt kitöltési tényező is 8 bit értéken állítható. Ezen megfontolásból egy kódoló dekódoló eljárás bevezetése ajánlott, ami a kiolvasott nyers adatokat a robot számára értelmezhetővé kell tenni, transzformáció segítségével.

A kódolási eljárásnál arra kellett hangsúlyt fektetni, hogy a robot processzorának ne kelljen sokat dekódolással töltenie, mert az számítási időbe kerülhet, és komoly memórafoglalással járhat, ami igen korlátozott méretű. A másik pont, amire figyelmet fordítottunk, hogy a robot oldalán kétféle motorvezérlésre készítettük fel az elektronikát; szervo motor illetve hagyományos egyenáramú motorok kezelésére. A különbség a két típusú motor között, hogy a szervo motor vezérléséhez PWM jeleket kell előállítani, amit 6.1.4.3. fejezetben már ismertettünk.

6.4.5. Adatküldési problémák a hálózaton

A hálózati kommunikáció esetében a képet átküldéséhez a *Bitmap.Save* metódusát használtuk, ami lokálisan jól működött. Amint éles hálózati környezetben próbáltuk ki csomag elhagyási, kiolvasási problémák jelentkeztek, mivel csak azt figyeltük van-e még várakozó csomag a hálózaton. Sok esetben előfordult, hogy bár az átvitel még nem fejeződött be, de a fogadó félén már nem volt mit kiolvasni, ezért az algoritmus abba hagyta az olvasást és visszaadott hibás adathalmazt. Ezért valamilyen felsőbb szintű *stream* kezelést kellett bevetni.

Szerializációs eljárások használata esetén, a visszaalakító függvény figyeli a streamen történt változásokat és csak akkor ad vissza adatot, ha a *stream* már nincs aktív használatban, ami nem egyenlő azzal, hogy nincs kiolvasható adat a pufferben. A legnagyobb probléma ezzel a módszerrel a megtöbbszöröződő adatforgalom, ami vezetékes hálózatok esetében elfogadható, azonban vezeték nélküli megoldásoknál már nem lesz elegendő sávszélesség az ilyen típusú kommunikációra.

Ezért valami megoldást kellett találni, hogy hibátlan átvitel mellett a csatorna adta korlátok is elegendők legyenek. A megoldást az jelentette, hogy a *Bitmap.Save* metódusával *JPEG* képként a streamre mentjük a képet, majd a fogadó félénél a *Bitmap* osztály speciális konstruktorát felhasználva (ami streamet fogad) kiolvassuk a képet. Ez abban az esetben működik, ha minden egyes képátküldés után a streamet lezárjuk, és ezzel jelezzük a fogadó félnek, hogy már nincs több adat.

6.4.6. Publikus adatok, képek szolgáltatása

A beintegrált http szerver, a képek hálózaton való elérésére ad lehetőséget. Egy webszerver univerzalitásából adódóan olyan eszközökre - amik a HTTP GET metódusát képesek kezelni - egy interface építhető fel, így mobil (IOS, Android, Windows Mobile) eszközök alkalmazására is lehetőség nyílik irányítás céljára, mert nem szükséges speciális programkódot írni a képek betöltésére, elég azokat letölteni. továbbá ezen eszközök gyakran rendelkeznek gyorsulás vagy helyzetérzékelővel, amiket felhasználva lehet a robotot számára irányítási adatokat kiolvasni. Egyes készülékeknél akár a Bluetooth-os kommunikációt bevetve, közvetlenül átadhatóak a kiolvasott értékek.

6.5. Modellépítés és megjelenítés

A szerver oldalon futó szoftver fő feladatát a robot irányításához szükséges adatok továbbítását, a robottól érkező kameraképek fogadását és feldolgozását jelentik.

A kameraképek feldolgozása egyrészt az érkező képek folyamatos megjelenítéséből áll, hogy a robotot megfelelően tudjuk irányítani, másrészt a környezet háromdimenziós modelljének elkészítéséből. Szintén a szerver oldalon történik a felépült modell megjelenítése a felhasználó számára. Mivel a probléma megoldása igen összetett, az ehhez szükséges módszereket, és a megvalósítás részleteit a következő fejezetben fejtjük ki bővebben.

7. A modellépítéshez használt képfeldolgozási módszerek

7.1. Az előfeldolgozás módszerei

A környezet alapján készülő modell felépítéséhez először a kamerából érkező folyamatos adás feldolgozása, a képeken jellegzetes pontok keresésre és a megtalált pontok hasonlóság alapján történő összepárosítása szükséges. A beazonosított pontpárok közötti távolság folyamatos számolásával és figyelésével meghatározhatjuk a további feldolgozásra alkalmas képkockákat.

A rendszer fejlesztésének kezdetén az architektúra, a robot oldalon is egy számítógép meglétét követelte, ami előfeldolgozással segítette volna a modellező szerver munkáját. A koncepció alapját az jelentette, hogy a hardveres feldolgozási sebességet figyelembe véve, minden beérkező képkockán le kell futtatni a jellegzetes pontdetektáló algoritmust. A modellező által feldolgozandó képek számának számítási igénye nagyban függ a robot mozgási sebességétől, hiszen a pontdetektálásból derül ki, hogy elég nagy-e már az elmozdulás az előző képhez képest ahhoz, hogy a feldolgozandó képet továbbítsa a szerver oldali modellezőnek. Ha az elmozdulás elegendő információt tartalmaz, ezeket az úgynevezett keyframe-eket küldjük tovább a bázisra, és ott épül a tényleges modell, ezzel spórolhatunk a robot-oldali feldolgozó teljesítményén, valamint az adatok átviteléhez szükséges sávszélességen. A modellező algoritmusunk gyorsasága abban rejlik, hogy nem elemzi a kamerából érkező összes képkockát, hanem csak a keyframe-eket. Figyelembe kell venni a keyframe-k kiválasztása során, a robot navigálásához minimálisan szükséges képkockák számát is.

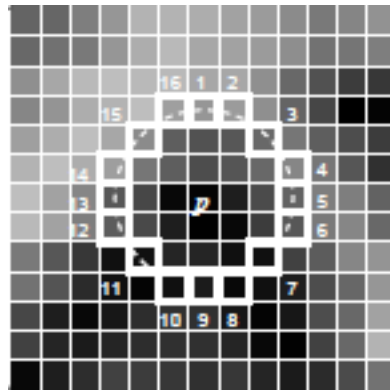
A feldolgozás ezen pontja a későbbiekben átkerült a szerver oldali feladatok közé, mivel a számítógép lecserélésre került egy mobiltelefonra, aminek számítási kapacitása jelentősen alatta marad egy notebook kapacitásának, cserébe a roboton nem szükséges egy teljes számítógépet elhelyeznünk, így az könnyebben mozgathatóvá vált.

7.1.1. Jellegzetes pontok detektálása

A sarokpontok jelentős szerepet játszanak a számítógépes képfeldolgozás során az elmozdulások követésében, képegyezések vizsgálatában, 3D modellezésben, vagy éppen képek összeillesztésében. Egy sarkot akár úgy is definiálhatunk, mint két él kereszteződése. Egy ilyen pont tekinthető szó szerint egy sarokpontnak, de akár jelenthet egy olyan pontot, ahol az adott környezetben a változásoknak lokális maximuma van, vagy akár egy vonal végpontját is jelölheti. A gyakorlatban sarokpont detektálásnak nevezik, de általánosságban minden olyan pontra alkalmazzuk a kifejezést, ami a feldolgozás során érdekes részleteket emelnek ki. A következőkben részletezünk néhány jellegzetes pontok detektálására használható módszert, amelyekből a feladat megoldása során kiválaszthatjuk a számunkra legmegfelelőbbet.

7.1.2. FAST: Features from Accelerated Segment Test

Gyors, megbízható módszer jellegzetes pontok keresésére [12]. Az algoritmus a vizsgálandó p képpont r sugarú Bresenham körére (7–1. ábra) eső képpontok tulajdonságait veszi figyelembe. Ha e pontok közül N -nél több darabszámú pont fényesebb p -nél egy megadott küszöbértéknyivel, akkor p -t jellegzetes pontnak tekinthetjük.



7-1. ábra: Bresenham-kör
[11 p. 5]

Először az 1. és 9. képpontokat vizsgálja: ha hasonló intenzitásúak mint p , akkor p nem lehet jellegzetes pont. Ezután az 5. és 13. képpontok következnek, majd ezzel a logikával elemzi a többi képpont párt. Ha $N = 12$, akkor legalább 12 képpontot meg kell vizsgálnunk, hogy biztosan állíthassuk p -ről, hogy jellegzetes pont, viszont két teszt alapján már megállapíthatjuk, ha nem az. Tesztekkel igazolható, hogy $r = 3$ és $N = 9$ esetén a FAST módszer jelentősen hatékonyabb más élkereső módszereknél, egyetlen hátránya az igen nagy zajérzékenység

7.1.3. SURF: Speeded Up Robust Features

Az először 2006-ban bemutatott jellegzetes pontokat kereső és leíró algoritmust eredetileg a SIFT (*Scale-invariant feature transform*) sarokdetektáló alapötlete ihlette [13], készítői szerint azonban robusztusabb a képen történő változásokkal szemben és gyorsabb. A módszer számításait integrál képekre támaszkodva végzi. Az integrál képeket az eredeti képből úgy építjük fel, hogy minden képpont a felette és tőle balra lévő képpontok összege lesz. Az így felépített képek meggyorsítják a későbbi számításokat. Az integrál képből négyzeteket kivágva, a négyzet belsejében lévő képpontok különbségeiből állapítható meg, hogy az adott területet jellegzetes pontnak tekintjük-e a későbbiekben.

7.1.4. Jellegzetes pontok követésének problémája

Feladatunk sikeres végrehajtásának érdekében nem elegendő megtalálnunk minden képkockán az adott jeleneten jellegzetesnek ítélt képpontokat, fel kell tudnunk ismerni őket a jelenetről készített többi képen is, majd az azonos pontokat egymáshoz kell párosítanunk. Más megfogalmazással a pontok „mozgását” kell követnünk a folyamat során. A mozgás leírására kétféle modellt alkalmazhatunk [14 p. 57]:

Tisztán egyenes vonalú mozgással leírható képdeformációs modell: Feltételezzük, hogy minden képpontunk azonos mozgást végez. Ez a modell csak akkor tekinthető helyesnek, ha a fotózott jelenet elég „lapos” és elég messze van a kameránktól, valamint a képeket készítő kamera lassan mozog, párhuzamosan a fotózott jelenettel. A követés során viszont nem a teljes képet vizsgálunk, hanem egy kis ablakot a követendő képpontunk körül. Ezzel a modellel végzett követések jelentik az alapját a legtöbb optika folyam és pontmegfeleltetési algoritmusnak.

Affin deformációs képmodell: A képpontok mozgását több paraméterrel írja le. A modell jó megközelítése az olyan mozgásoknak, amikor a megfigyelt kép valamilyen elmozdulást és forgást végez a kamera optikai tengelyéhez viszonyítva.

Mivel csak egyetlen képpont követése lehetetlen feladat a képkockákon létrejövő torzítások, megvilágítási változások miatt, ezért minden esetben a képpont körüli környezetet, valamilyen ablakot kell vizsgálnunk.

Két ilyen I_1 , I_2 régió, vagy ablak akkor felel meg egymásnak, ha teljesítik a következő egyenlőséget:

$$I_1(x) = I_2(h(x, \alpha)) \forall x \in W(x)$$

Ahol $(h(x, \alpha))$ jelenti az x pont mozgási paramétereit a kétféle modell alapján. A tisztán egyenes vonalú mozgást leíró esetben α csak az elmozdulás nagyságát írja le, affin modellt választva pedig forgást és elmozdulást is reprezentál. $W(x)$ jelenti a vizsgált x képpont körüli szomszédságot. A gyakorlatban azt az x pontot tekintjük megfelelő jellegzetes pontnak, amely a mozgás α paraméterét a legegységesebben meghatározza. A fellépő zaj miatt elképzelhető, hogy a fenti egyenletnek sosem lesz megoldása, tehát ha megfogalmazzunk egy $\phi(I_1, I_2)$ eltérési függvényt, gyakorlatilag a következő minimalizációs problémát kell megoldanunk:

$$\alpha = \arg \min_{\alpha} \phi(I_1(x), I_2(h(x, \alpha)))$$

Természetesen a módszer nem csak jellegzetes pontokra, ha nem jellegzetes vonalakra is alkalmazható.

Pontkövetés az affin képdeformációs modell

Feladatunk során a két modell közül ezt tudjuk hasznosítani, így egy kicsit részletesebben is foglalkozunk a működésével. Ha feltételezzük, hogy a képpontok egymástól függetlenül mozognak, akkor az egyes régiók mozgását a következőképpen írhatjuk le: $I_1(x) = I_2(h(x))$, ahol h a következő alakú:

$$h(x) = Ax + d = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} x + \begin{bmatrix} \alpha_5 \\ \alpha_6 \end{bmatrix}$$

Továbbá meg kell fogalmaznunk a fényerő állandóságának feltételét:

$$I(x, t) = I(Ax + d, t + dt)$$

A fenti egyenletet alkalmazva a kép egy régióján, meghatározhatjuk a modell mozgási paramétereit (A, d), ha integrálunk az alábbi egyenlettel, minden $W(x, y)$ régióban lévő pontra.

$$E_a(A, d) = \sum_{W(x)} (I(x, t) - I(Ax + d, t + dt))^2$$

A fentebb említett minimalizációt megoldva jó becslést kapunk a mozgási paraméterekre, melyek alapján meghatározhatjuk a vizsgált képpontunk helyét egy következő képkockán.

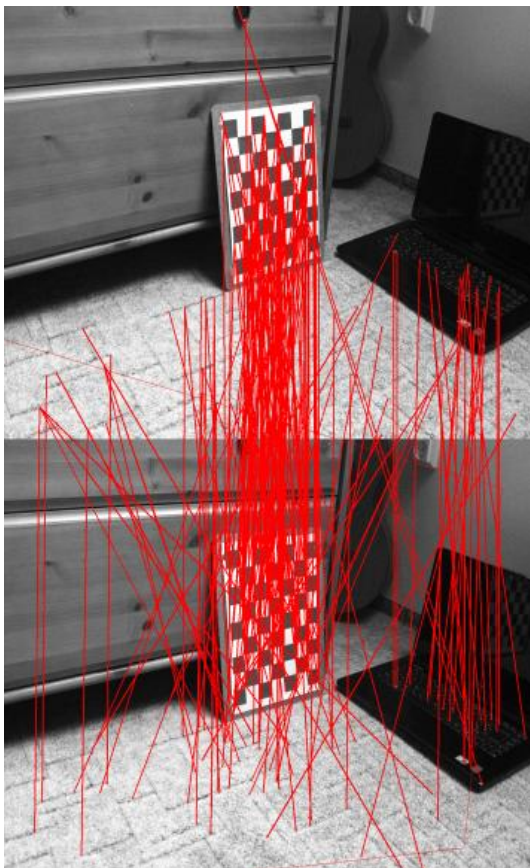
7.1.5. A pontkeresés megvalósítása

A jellegzetes pontok kinyerésére és követésére a tesztek során többféle módszert is kipróbáltunk. A már ismertetett FAST módszer mellett szól, hogy futása igen gyors, képenként nagyjából 300 megtalált pont esetén is kevesebb, mint egy másodperc alatt ad eredményt. Hátránya, hogy sajnos eléggé érzékeny a zajra, így, ha nagyobb az elmozdulás a két vizsgált kép között, elég sok hibás párosítást produkál. Ennek kiküszöbölésére megpróbáltuk egy egyszerű, pontpárok távolságán

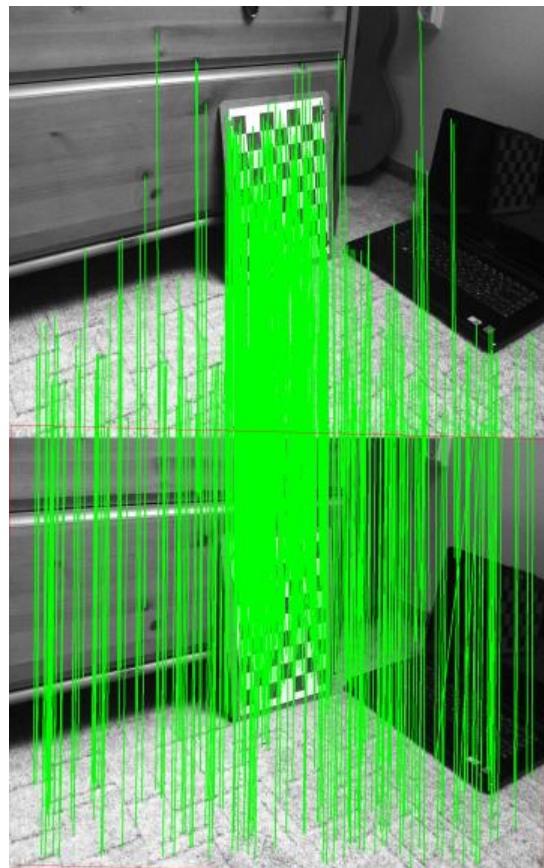
alapuló összehasonlítással kiszűrni a hibás párokat. A módszer lényege, hogy ha egy bizonyos küszöbnél nagyobb a távolság a pontpár tagjai között, az adott párt eldobjuk, ugyanis nem valószínű, hogy a pontunk az egyik képkockán például a kép bal felső sarkában van, a következőben pedig a jobb alsóban. Ezek után az eredmények szemmel láthatóan javultak, bár a megfelelő küszöbérték megválasztása meglehetősen nehéz, és jelenetfüggő kérdésnek bizonyult. Rossz küszöbérték esetén pedig sok jó megoldást is eldobhatunk.

A szintén ismertetett SURF módszerrel kapcsolatban hasonló problémákba ütköztünk. A hibás párok terén valamivel jobb eredményt produkált, de az utófeldolgozás itt is szükséges lépés maradt. Mivel a FAST módszernél hosszabb futási idővel rendelkezik, így végül ennek a módszernek a használatát is elvetettük.

A legjobban használható módszernek az affin pontkövetéses algoritmus bizonyult. Bár a pontkeresés futási ideje az előző két módszernél valamivel lassabb, a párosítás során született eredmények sokkal megbízhatóbbnak bizonyultak, így az utófeldolgozást megspórolhatjuk. Ha a pontok közötti átlagos távolság elér egy bizonyos nagyságot, megállapíthatjuk, hogy a kamera már eleget mozdult el ahhoz, hogy a képek erőforrás takarékosan felhasználhatóak legyenek a végleges modell építéséhez. Az így megtalált úgynevezett kulcs-képkockákat felhasználva kezdődik meg a pontok térbeli pozíciójának meghatározása. Az alábbi ábrák (7-2, 7-3) a két módszer alkalmazásával kapott szűretlen pontkövetési eredményeket mutatják. Az ábrákon ferdén fekvő vonalak nagy valószínűséggel téves párosításokat jelölnek. Bár SURF módszernél kevesebb hiba figyelhető meg, a gyakorlati alkalmazás során az affin követés megbízhatóbbnak bizonyult, egyszerűbb paraméterezése miatt.



7-2. ábra: FAST módszer eredményei



7-3. ábra: SURF módszer eredményei

7.2. Pontpárok elhelyezése a térben

A pontpárok megtalálása után következik a környezet háromdimenziós modelljének elkészítése. A különböző képeken megtalált azonos pontok alapján először ki kell számítanunk azok térbeli pozícióját. Hogy ezt megtehesük meg kell értenünk néhány geometriai fogalmat. A következőkben a probléma megoldására alkalmas módszereket részletezzük.

7.2.1. A projektív kamera

A térbeli pontokat képpontokká leképező kameránkat jól leírhatjuk egy P kameramátrixszal, amely ezen 3D-2D transzformáció adatait tárolja. A művelet természetesen információvesztéssel jár, a jelenet minden tulajdonságát nem tudjuk megőrizni, a kapott kameramátrix segítségével alapesetben csak egy projektív modell elkészítésére van lehetőségünk. Ezt a projektív transzformációt végrehajtó kamerát nevezzük projektív kamerának [15 p. 153], a transzformációt pedig P mátrixszal írjuk le.

Ez a P a térbeli X pontokat x képpontokká képezi le a következő alapján: $x = PX$. A mátrix felbontható a következő formában: $P = [M \mid p_4]$, ahol M egy 3×3 -as mátrix, p_4 pedig eltolás. P -nek létezik egy egydimenziós jobboldali nulltere, mivel rangja 3, és oszlopainak száma 4. Ezt a nullteret jelöljük C -vel, amely nem más, mint a kamera középpontjának pozíciója, és teljesíti a következő összefüggést: $PC = 0$. Ha vizsgálunk egy térbeli vonalat, amely tartalmazza C -t és egy tetszőleges A pontot, a további pontok ezen a vonalon a következőképpen írhatók le:

$$X(\lambda) = \lambda A + (1 - \lambda)C$$

Alkalmazva az $x = PX$ leképezést a pontokra, mivel $PC = 0$, ezért:

$$x = PX(\lambda) = \lambda PA + (1 - \lambda)PC = \lambda PA$$

Vagyis minden pont ezen a vonalon, ugyan abba a PA képpontba fog leképeződni, tehát a vonalunknak egy, a kamera középpontján áthaladó sugárnak kell lennie. Ennek következményeképpen C a kamera középpontjának homogén reprezentációja, mert bármilyen A -t választunk, az $X(\lambda)$ vonal egy C -n áthaladó sugár lesz.

A használt P mátrix nem csak egészsként alkalmazható számítások végrehajtására, hiszen egyes elemei külön-külön is fontos geometriai jelentőséggel bírnak:

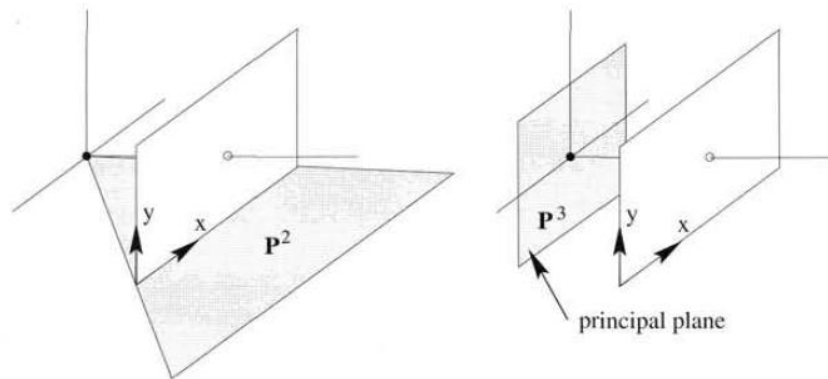
P oszlopvektorai: hármas vektorok, melyek geometriailag speciális pontokat jelölnek. Ha P oszlopait p_i -vel jelöljük ($i = 1 \dots 4$), akkor p_1, p_2, p_3 jelöli a világkoordináta-rendszer X, Y és Z tengelyeinek távlatpontjait. Gyakorlatban ezek a pontok jelentik a tengelyek irányának leképezését, és p_4 jelenti a világközéppont képét.

P sorvektorai: ezek a négyes vektorok geometriailag speciális síkokat jelképeznek. Jelöljük P sorait P^{iT} -vel, a következők alapján:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} P^{1T} \\ P^{2T} \\ P^{3T} \end{bmatrix}$$

Elsődleges sík: az a sík, amelyen a kamera középpontja fekszik, és merőleges a képsíkra. Azokat a térbeli X pontokat tartalmazza, amelyek a képen a végtelen vonalára képeződnek le. Tehát felírható a

következő összefüggés: $PX = (x, y, 0)^T$, vagyis egy pont akkor, és csak akkor van ezen a síkon, ha $P^{3T}X = 0$. A definícióból következik, hogy $P^{3T}C = 0$, vagyis a kamera középpontja a síkon van.



7-4. ábra: A kamera síkjai [15 p. 178]

Tengelyek síkjai: vizsgáljuk azokat az X pontokat, amelyek a P^1 síkon fekszenek. Ezek a pontok eleget tesznek a $P^{1T}X = 0$ feltételnek és $PX = (0, y, w)^T$ formában képeződnek le, vagyis a képen az y tengely pontjait alkotják (7-4. ábra). $PC = 0$ egyenlőségből következik, hogy $P^{1T}C = 0$, vagyis a kamera középpontja a P^1 síkon fekszik. A P^1 síkot tehát C és a képen lévő $x = 0$ egyenes egyértelműen meghatározza. Ugyan ez bebizonyítható P^2 -vel és az $y = 0$ egyenessel.

Elsődleges pont: az elsődleges tengely az az egyenes, amely a kamera középpontján áthalad, és merőleges az elsődleges síkra, P^3 -ra. Ez a tengely a képsíkot a főpontban (*principal point*) metszi. Ez a pont a következőképpen határozható meg: egy $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T$ sík normálisát jelölje $(\pi_1, \pi_2, \pi_3)^T$ vektor. Ezt akár egy $(\pi_1, \pi_2, \pi_3, 0)^T$ ponttal is jelölhetjük, amely a végtelen síkján fekszik. Ha a kamera P^3 síkját vizsgáljuk, ez a pont $(p_{31}, p_{32}, p_{33}, 0)^T$, melyet $\hat{P}^3$ -mal jelölünk. Ha ezt a pontot P kamera használatával kivetítjük, megkapjuk $P\hat{P}^3$ kamera elsődleges pontját. Ilyenkor a $P = [M \mid p_4]$ kameramátrixnak csak baloldali 3×3 -as részét használtuk fel. Gyakorlatban az elsődleges pont a következő egyenlettel számolható: $x_0 = Mm^3$, ahol m^{3T} jelöli M harmadik sorát.

Elsődleges tengely vektora: Bár minden térbeli X pont, amely nem az elsődleges síkon helyezkedik el, leképezhető az $x = PX$ egyenlet segítségével képponttá, ezen pontoknak csak a fele látszik a képen, még hozzá azok, amelyek a kamera előtt helyezkednek el. Írjuk fel P -t $P = [M \mid p_4]$ alakban. Láttuk, hogy az m^3 vektor az elsődleges tengely irányába mutat. Az elsődleges vektort úgy szeretnénk definiálni, hogy a kamera elé mutasson (vagyis pozitív irányba). Mivel P előjelét tudjuk meghatározni, ezért azt sem tudjuk, hogy m^3 vagy $-m^3$ mutat-e pozitív irányba. Ezt a bizonytalanságot kell megszüntetnünk. Először a kamera koordináta-rendszeréhez képest vizsgáljuk a pontokat. A térbeli X pont képe $x = P_{cam}X_{cam} = K [I \mid 0] X_{cam}$, ahol X_{cam} jelöli a térbeli pont koordinátáit a kamera koordináta-rendszerében. Így a $v = \det(M)m^3 = (0, 0, 1)^T$ vektor pontosan a kamera elé fog mutatni, az elsődleges tengely irányában, P_{cam} skálázásától függetlenül.

7.2.2. A kameramátrix felbontása

Kamera középpontjának megtalálása: a kamera középpontja az a C pont, amelyre igaz, hogy $PC = 0$. Numerikusan a P mátrix SVD-je során előálló jobboldali nullvektorként állítható elő. Algebrai módon, ha $C = (X, Y, Z, T)^T$, akkor:

$$X = \det([p_2, p_3, p_4])$$

$$Y = -\det([p_1, p_3, p_4])$$

$$Z = \det([p_1, p_2, p_4])$$

$$T = -\det([p_1, p_2, p_3])$$

Kamera orientációnak és belső paramétereinek megtalálása: amennyiben kameránkat felbontjuk a következő módon: $P = [M \mid -M \tilde{C}] = K [R \mid -R \tilde{C}]$, kiszámíthatjuk K -t és R -t, ha M -et a felírjuk $M = KR$ formában. Ezt az RQ-dekompozíciós módszer segítségével tehetjük meg. Az R mátrix adja meg a kamera orientációját, K pedig a kalibrációs mátrix, amely a következő formájú:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ahol:

- α_x jelöli a skálázás nagyságát az x koordináta irányában,
- α_y jelöli a skálázás nagyságát az y koordináta irányában,
- s a ferdeségi paraméter,
- $(x_0, y_0)^T$ jelöli az elsődleges pont koordinátáit,
- a képarány pedig α_y / α_x .

7.2.3. Pontok elhelyezése a térben

Feladatunk a megtalált x_{1i} , x_{2i} pontpárok alapján kiszámítani a hozzájuk tartozó térbeli X_i pontot, valamint a képekhez tartozó P_1 , P_2 kameramátrixokat úgy, hogy a következő összefüggés minden i -re teljesüljön [15 p. 262]:

$$x_{1i} = P_1 X_i \text{ és } x_{2i} = P_2 X_i$$

Ha túl kevés ilyen pontpárral rendelkezünk, a feladatot lehetetlen megoldani. Elegendő pontpár esetén a pontokat helyre tudjuk állítani, de csak projektív bizonytalansággal. A bizonytalan modell pontossága növelhető, ha további információval rendelkezünk a kamerákról, vagy azok környezetéről.

A módszer alapvető lépései a következők:

1. Számítsuk ki az F alpmátrixot a megtalált pontpárok alapján.
2. Számítsuk ki a kameramátrixokat F -ből.
3. Minden megtalált pontpárra számítsuk ki a hozzájuk tartozó térbeli pont pozícióját.

Az egyes lépésekkel természetesen a későbbiekben részletesen is foglalkozunk.

7.2.3.1. A rekonstrukció bizonytalansága

Amennyiben nem rendelkezünk háttérinformációval a vizsgált jelenetről, annak pontos modellje általában nem építhető fel. Ez akkor is igaz, ha rendelkezünk például a kamerák belső paramétereit leíró mátrixszal, illetve a relatív kamerapozíciókkal. Ilyenkor legjobb esetben is csak a jelenet Euklideszi-transzformációval (eltolás, forgatás) megváltoztatott modelljét tudjuk leírni egy referenciaponthoz képest. Másik probléma, hogy a jelenet skálázása sem nyerhető vissza teljesen, tehát a végső modellen vizsgált tárgyakról nem tudjuk pontosan megmondani hány méter, vagy centiméter hosszúságúak. Ha rendelkezünk egy referenciával az adott jelenetből (például tudjuk, hogy egy ajtó pontosan 90 centiméter széles), akkor már képesek vagyunk a pontosabb modell előállítására, ilyen extra információk nélkül viszont a modellünk az eredeti jelenetnek csak egy hasonlósági transzformáltja lesz (forgatás, eltolás, skálázás).

Ha X_i -vel jelöljük a térbeli pontjaink halmazát, P_1, P_2 -vel pedig azokat a kamerákat, amelyek x_{1i}, x_{2i} pontokba vetítik a térbeli pontokat akkor megvizsgálhatjuk a következő hasonlósági transzformációt:

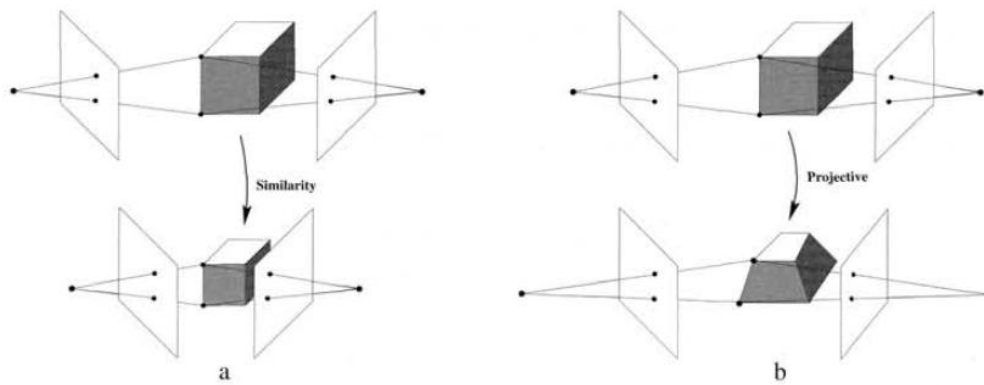
$$H_S = \begin{bmatrix} R & t \\ 0^T & \lambda \end{bmatrix}$$

Ahol R a forgatásnak, t az eltolásnak és λ^{-1} az átfogó skálázásnak felel meg. Ha minden X_i pontot $H_S X_i$ -vel és minden P_1, P_2 kamerát $P_1 H_S^{-1}$ és $P_2 H_S^{-1}$ -el helyettesítünk, az nem változtatja meg a megfigyelt képpontokat, mivel $P X_i = (P H_S^{-1})(H_S X_i)$. Továbbá ha felbontjuk P -t $P = K[R_P R^{-1} \mid t']$ formában, a következő összefüggés számítható ki:

$$P H_S^{-1} = K[R_P R^{-1} \mid t']$$

Ahol t' -t nem kell pontosabban kiszámítanunk. Ennek eredményeképpen belátható, hogy H_S^{-1} -el szorozva P kalibrációs mátrixai nem változnak meg. Ez a bizonytalanság megmarad kalibrált kamerák használata esetén is, viszont ebben az esetben bizonyíthatóan ez az egyetlen ilyen bizonytalansági tényező. Tehát kalibrált kamerák használatával a jelenet modellje és az eredeti jelenet között csak egy hasonlósági transzformáció a különbség.

Projektív bizonytalanság: Ha semmit sem tudunk a kamerák pozíciójáról, illetve azok kalibrációjáról, az elkészített modell az eredeti jelenetnek csak egy projektív transzformáltja lehet. Ha ezt a transzformációt egy 4x4-es invertálható mátrixszal fejezzük ki, és a térbeli pontokat ezzel, valamint a kamerákat ennek inverzével szorozzuk, az szintén nem változtatja meg a kapott képpontokat. Ez megmutatja, hogy kalibrálatlan kamerákkal az eredeti jelenetnek csak egy projektív transzformáltját vagyunk képesek előállítani.



7-5. ábra: Hasonlósági és projektív rekonstrukció [15 p. 265]

7.2.3.2. Projektív rekonstrukció

Ha rendelkezünk azonosított pontpárokkal két kameraképen, amelyek egyértelműen meghatároznak egy F alaplátrixot (*fundamental matrix*), a pontok térbeli pozíciója és a kamerák helyzete kinyerhető ezekből a pontpárokból, és bármely két ilyen modell, amely ezekre a pontpárookra épít, projektíven ekvivalens. A két kamerát összekötő vonalon fekvő pontok ez alól kivételt képeznek, mert azok nem nyerhetők vissza egyértelműen. Az állítás jelentősége, hogy a jelenet projektív modellje csupán a pontpárok ismeretéből kiszámítható, a kamera kalibrálása és helyzetének ismerete nélkül. Gyakorlatban a valós modell ebből a projektív modellből egy projektív transzformációval előállítható.

Tehát, ha a valós rekonstrukció adatai $(P_{E1}, P_{E2}, \{X_{Ei}\})$, a projektív pedig $(P_1, P_2, \{X_i\})$ akkor létezik olyan 4×4 -es H hasonlósági mátrix, melyre igazak a következő egyenlőségek:

$$P_{E1} = P_1 H^{-1} \text{ és } P_{E2} = P_2 H^{-1} \text{ valamint } X_{Ei} = H X_i$$

Néhány helyzetben elegendő ezt a projektív modellt előállítanunk (például ha egy pont és egy sík metszéspontját keressük), de a valós modell előállításához is felhasználhatjuk kiindulási pontként.

7.2.3.3. Affin rekonstrukció

A modell erre a szintre fejlesztéséhez meg kell találnunk a síkot a végtelenben. Tegyük fel, hogy a projektív modellünk a következő hármastól áll: $(P_1, P_2, \{X_i\})$. Továbbá létezik egy π sík, amely a végtelenben helyezkedik el. A valódi modellben a π sík koordinátái: $(0,0,0,1)^T$, tehát célunk megtalálni azt a projektív transzformációt, amely π -t ezekre a koordinátákra képezi le, vagyis $H^T \pi = (0,0,0,1)^T$. A transzformáció a következő alakú:

$$H = \begin{bmatrix} I & 0 \\ \pi^T \end{bmatrix}$$

Ezt kell alkalmaznunk minden térbeli pontra, és a kameramátrixokra. A módszer nem alkalmazható, ha π^T utolsó koordinátája nulla. Ebben az esetben azt a H^T mátrixot kell megtalálnunk, melyre igaz:

$$H^{-T} \pi = (0,0,0,1)^T$$

Ezek után a jelenlegi modellünk egy affin transzformációval tér el a valóstól. Néhány felhasználásra ez is elegendő lehet, például ha egy ponthalmaz középpontját keressük, vagy meglévő egyenesekkel szeretnénk párhuzamos egyeneseket előállítani. Ilyen számítások nem végrehajthatóak a projektív modellel.

Ahhoz hogy megtalálhassuk a végtelen síkját, szükségünk van néhány extra információra a vizsgált jelenetről.

Tiszta egyenes mozgású kamera: Ha a kameránk nem végez forgást, csupán csak egy irányba mozdul el, a végtelen síkja akár két kép alapján is megtalálható. Ilyenkor egy térbeli X pont mind a két képen ugyanarra a pontra képeződik le. Ez könnyen észrevehető a mindennapi életben is: ha egy egyenes vonal mentén mozgunk, a nagy távolságban lévő tárgyak mozdulatlanak tűnnek, míg a közeleiek gyorsan mozognak. Az elkészített projektív modellben, ha találunk három olyan térbeli X pontot, melyekhez tartozó x_1, x_2 koordináták egyenlők, ezekkel egyértelműen meghatározhatunk egy síkot, amely a végtelen síkja lesz. Ha már előre rendelkezünk azzal az információval, hogy a kameránk csak ilyen elmozdulást fog végezni, az F mátrix számítása leegyszerűsödik, a következőképpen:

$$F = [e_1]_x = [e_2]_x$$

A két kameramátrix pedig ebben a formában írható fel:

$$P_1 = [I \mid 0] \text{ és } P_2 = [I \mid e_2]$$

Ismert pontok: Amennyiben előre ismerünk három olyan térbeli pontot a jelenetből, amelyek biztosan a végtelen síkján fekszenek, a sík ezek alapján egyszerűen meghatározható.

Párhuzamos vonalak: A legfontosabb kiindulópontunk az, hogy a térben párhuzamos vonalak a képeinken nem párhuzamosak, ezért egy ponton metszeniük kell egymást. Ezek az úgynevezett távlatpontok a végtelen síkján helyezkednek el. Tehát ha találunk a képeinken három pár párhuzamos egyenest, a metszéspontjaikból meghatározható három ponttal már kiszámíthatjuk a végtelen síkját.

7.2.3.4. Metrikus rekonstrukció

Ahhoz, hogy a modellünket metrikussá fejleszthessük, meg kell találnunk annak a térbeli kúpnek a végtelen síkjával való metszetét, amely a kamera által végrehajtott projektív transzformáció további paramétereit adja meg. Ez a kúpszelet („*absolute conic*”), melyet Ω_∞ -nel jelölünk, fejezi ki az affin koordináta rendszer metrikus tulajdonságainak helyreállításához szükséges további 5 szabadsági fokot. Fontos tulajdonsága, hogy értéke hasonlósági transzformációk által nem változik. Megtalálásához előbb meg kell keresnünk képét az egyik rendelkezésre álló kameraképen, amelyet visszavetítve a térbe már kiszámíthatjuk metszéspontját a végtelen síkjával.

Tegyük fel, hogy az affin modellünkben ennek az Ω_∞ -nek egy $P = [M \mid m]$ kamera által látott képe egyenlő ω -val („*image of the absolute conic*”, a továbbiakban *IAC*). Ekkor a modellünk a következő hasonlósági transzformációval fejleszthető tovább metrikussá:

$$H = \begin{bmatrix} A^{-1} & \\ & 1 \end{bmatrix}$$

Ahol A Cholesky-faktorizációval állítható elő a következő egyenlet alapján: $AA^T = (M^T \omega M)^{-1}$.

Ω_∞ megtalálására három módszer ismert, a gyakorlatban pedig általában ezek kombinációját alkalmazzuk:

A jelenet ortogonalitásából fakadó információk: két távlatpont, v_1 és v_2 használatával felállítható egy feltétel ω -ra:

$$v_1^T \omega v_2 = 0$$

Ehhez hasonlóan, egy távlatvonal és egy távlatpont szintén megad egy ilyen összefüggést:

$$l = \omega v$$

Gyakran használjuk a kettőt párhuzamosan: egy távlatpontot vizsgálunk függőleges irányban, és egy távlatvonalat vízszintesen. Ennek következményeképpen, egy jelenet segítségével, amely tartalmaz egy olyan síkot, amelyről pontos metrikus információink vannak (például egy sakktábla), mindkét egyenlőség felállítható.

Ismert belső paraméterek: ha ismerjük a felhasznált kamera K kalibrációs mátrixát, ω meghatározható a következő módon:

$$\omega = K^{-T} K^{-1}$$

Így a belső paraméterek ismeretével szűkíthetjük, vagy meghatározhatjuk ω -t. Abban az esetben, ha a kameránknál a tengelyek közötti dőlésszög 0° , további összefüggéseket írhatunk fel:

$$\omega_{12} = \omega_{21} = 0$$

Továbbá, ha a kamerapixelek négyzet alakúak:

$$\omega_{11} = \omega_{22}$$

Ugyanazt a kamerát használjuk minden képhez: az Ω_∞ egyik legfontosabb tulajdonsága, hogy képe egy adott kameraképen csak a kalibrációs mátrixtól függ. Ha egy P_1 és P_2 kameramátrixok kalibrációs mátrixa azonos (ami gyakran azt jelenti, hogy a két kép ugyan azzal a kamerával készült, különböző helyzetekből), akkor $\omega_1 = \omega_2$. Ha elég kamerakép áll rendelkezésünkre egyetlen kamerából, felhasználhatjuk ezt a tulajdonságot Ω_∞ megtalálására. Mivel Ω_∞ a végtelen síkján fekszik, képe az egyik kameraképről a másikra a következő egyenlőséggel számítható:

$$\omega' = H_\infty^{-T} \omega H_\infty^{-1}$$

Ahol ω és ω' jelentik Ω_∞ képeit a képkockákon. A módszer alkalmazásához rendelkezniünk kell a jelenet affin modelljével, hogy ki tudjuk számítani H_∞ -t. Ha $\omega = \omega'$, lineáris egyenletrendszer kapunk ω értékeire. Gyakorlatban ezzel az egyenletrendszerrel négy változó állapítható meg ω öt értéke közül. Ahhoz hogy ω -t egyértelműen meghatározhassuk, fel kell használnunk a fentiekben említett összefüggések valamelyikét, például a kamerák belső paramétereit.

7.2.3.5. Közvetlen metrikus rekonstrukció

Amennyiben ismerjük ω -t, a projektív modellt közvetlenül is továbbfejleszthetjük metrikussá, ha legalább két képpel rendelkezünk az adott jelenetről. A problémára két megoldás is ismert:

IAC módszer: a módszer ω és a K kalibrációs mátrix közötti kapcsolaton alapszik, vagyis: $\omega = (KK^T)^{-1}$. Így akár K -t is kiszámíthatjuk ω -ból, ha invertáljuk és Cholesky-faktorizációt alkalmazunk rajta. Ha

mindegyik képhez ismerjük ω -t, kiszámíthatjuk a kamerák kalibrációs mátrixát. Ezután a metrikus modell előállítható az E mátrix segítségével.

A másik módszert követve, ha ismerjük ω -t, közvetlenül meghatározhatjuk belőle a végtelen síkját és Ω_∞ -t. Ehhez szükséges a projektív modell kameramátrixainak (P_1, P_2) ismerete is. Ha ω -t a kameramátrixok segítségével visszavetítjük mindkét képen, pontosan Ω_∞ -ban fogják egymást metszeni a térben, ezzel a hozzá tartozó π_∞ is meghatározható. Bár ezzel valójában két megoldást is kapunk Ω_∞ -re, de a belőlük nyert modell csak egy 180° forgatásban különbözik.

7.2.4. Az epipoláris geometria

Hogy megérthessük a módszer működését, először is meg kell ismerkednünk az epipoláris geometria fogalmával. Ha a korábban ismertetett módszerekkel sikerül egy mozdulatlan jelenetről, két különböző helyről készített képeken megtalálnunk, és összepárosítanunk a jellegzetes pontokat, akkor, ha kiküszöböltük a képkockákon létrejövő torzítást, elmondhatjuk, hogy pontpár egy térbeli pont képe. A pontpárt alkotó pontok között fontos geometriai összefüggéseket fedezhetünk fel [14 p. 80].

Ha a térbeli pontunk háromdimenziós koordinátáit a két kamera pozíciójához képest $X_1 \in \mathbb{R}^3$ és $X_2 \in \mathbb{R}^3$ jelölésekkel látjuk el, a következő egyenletnek kell teljesülnie:

$$X_2 = RX_1 + T$$

Amely a képpontokként (x_i) és azok mélységeként (λ_i) is felírható:

$$\lambda_2 x_2 = R\lambda_1 x_1 + T$$

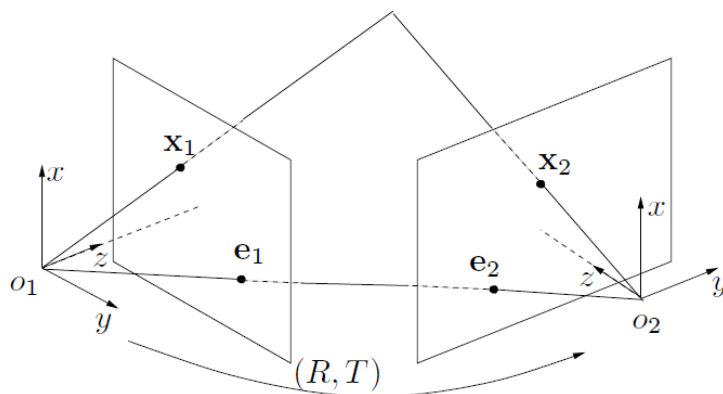
Hogy a mélységet eltüntethessük az egyenletből, mindkét oldalt megszorozzuk $\hat{T}$ - vel:

$$\lambda_2 \hat{T} x_2 = \hat{T} R\lambda_1 x_1$$

Ahol $\hat{T} x_2 = T \times x_2$. Mivel $\lambda_1 > 0$, x_1 és x_2 eleget tesz a következő megszorításnak:

$$x_2^T \hat{T} R x_1 = 0$$

Ez az úgynevezett epipoláris megszorítás. A következő ábra (7-6.) részletesen bemutatja az epipoláris geometriát:



7-6. ábra: Az epipoláris geometria [14 p. 81]

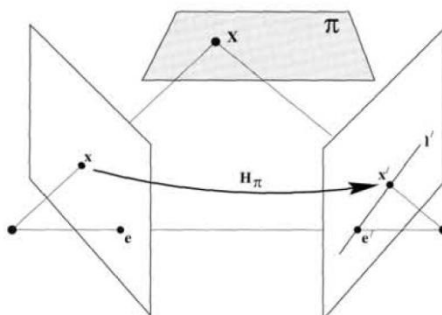
A két kamera helyzete (o_1 és o_2) és a térbeli p pont által meghatározott síkot epipoláris síknak, a két kamerát összekötő vonal és a képsík metszéspontját epipoláris pontnak (e_1 és e_2) és az epipoláris sík és a képsík metszeti vonalát epipoláris vonalnak nevezzük.

7.2.4.1. Az F alapmátrix

Az F mátrix jelenti az epipoláris geometria algebrai reprezentációját. Ha egy térbeli X pont két képen található képét vesszük, az első képen lévő x_1 -hez tartozó x_2 pontnak az epipoláris vonalra (l_2) kell esnie. Ezzel kapunk egy $x_1 \mapsto l_2$ megfeleltetést, amely gyakorlatilag egy projektív leképezés pontok és vonalak között. Ezt a leképezést írja le számunkra az F mátrix [15 p. 241].

A leképezés két lépésből áll: először meg kell találnunk x_1 olyan x_2 megfelelőjét a második képen, amely az epipoláris vonalon (l_2) fekszik, majd kiszámítjuk l_2 -t x_2 és e_2 összekötésével.

1. lépés: Ponttranszformáció síkon keresztül: Vegyünk egy π síkot a térben, amely nem metszi egyik kameránk pozícióját sem. A kamerát és x_1 -et összekötő vonal pontosan a térbeli pontban, X -ben metszi a π síkot. Ezután X -et kivetítjük x_2 -be a második képen. Mivel X az x_1 -el azonos vonalon fekszik, a kivetített x_2 pontnak az epipoláris vonalon (l_2) kell lennie. Ez minden megtalált képpontpárra elmondható, tehát létezik olyan H_π kétdimenziós hasonlósági transzformáció, amellyel minden x_1 leképezhető x_2 -be.



7-7. ábra: Transzformáció síkon keresztül [15 p. 243]

2. lépés: Az epipoláris vonal kiszámítása: az x_2 -höz tartozó e_2 -n átmenő epipoláris vonal (l_2) leírható a következő formában: $l_2 = e_2 \times x_2 = [e_2]_\times x_2$. Mivel minden x_2 leírható a

$x_2 = H_\pi x_1$ egyenlettel, ezért definiálhatjuk a következőt: $l_2 = [e_2]_\times H_\pi x_1 = F x_1$, ahol F jelöli az alapmátrixot (*fundamental matrix*).

Gyakorlatban az F mátrix kétdimenziós térből egydimenziós térbe való leképezést ír le, és rangja mindig 2.

A két kamera relatív pozícióját leíró P_1 és P_2 mátrixok F ismeretével számíthatók ki. A térbeli pont helyzete a kameramátrixok alapján a következő formában írható le:

$$X(\lambda) = P^+ x + \lambda C$$

Ahol P^+ jelöli P pszeudo-inverzét (vagyis $PP^+ = I$), és C ennek nullvektora, vagyis a kamera középpontja (tehát $PC = 0$). Az előzőekben felírt H_π hasonlóság immár felírható $H_\pi = P_2 P_1^+$ formában. Tehát F felfogható úgy is, mint: $F = [e_2]_\times P_2 P_1^+$.

Ha az első kamera pozícióját vesszük kiindulási pontnak, akkor a kameramátrixok a következő alakúak:

$$P_1 = K_1 [I \mid 0]$$

$$P_2 = K_2 [R \mid t]$$

Ahol K_1 és K_2 jelölik a kamerákhoz tartozó kalibrációs mátrixokat, R és t pedig a két pozíció közötti forgatást és elmozdulást.

Két, nem egybeeső középpontú kamera által meghatározott F mátrix fontosabb tulajdonságai:

- egyedi, 3x3-as homogén mátrix
- rangja 2
- minden x_1, x_2 pontpárra igaz, hogy: $x_2^T F x_1 = 0$. A megállapítás jelentősége, hogy F nem csak a két kameramátrix alapján számítható ki, hanem elég hozzá, ha elegendő pontpárt meg tudunk találni a vizsgált képeken.
- determinánsa 0

7.2.4.2. Az esszenciális (E) mátrix

Az F mátrix speciális esete, amely normalizált koordináták alapján számítható ki. Gyakorlatilag az E mátrixot fedezték fel hamarabb, az F csak ennek általánosítása, ahol a kamerákat nem tekintjük kalibrálnak.

Normalizált koordináták: ha felbontjuk a kameramátrixot a következőképpen: $P = K [R \mid t]$ és $x = PX$ egy pont a képünkön, akkor, ha rendelkezünk a K kameramátrixszal, alkalmazhatjuk annak inverzét a ponton: $\hat{x} = K^{-1} x$. Ekkor $\hat{x} = [R \mid t] X$, ahol $\hat{x}$ jelöli a képpontot normalizált koordinátákkal. Normalizált kamera mátrixnak nevezzük a következőt: $K^{-1}P = [R \mid t]$. Ezzel az ismert kalibrációs mátrix hatását tüntetjük el a kameramátrixból. A normalizált kameramátrixokból számított F mátrixot nevezzük E mátrixnak, amely a következő alakú: $E = [t]_\times R = R [R^T t]_\times$

Az E mátrix definíciója normalizált képkoordináták esetén a következő:

$$x_2^T E x_1 = 0$$

Az F és E mátrixok közötti összefüggés pedig a következő formában írható fel:

$$E = K_2^T F K_1$$

Az E mátrix tulajdonságai:

- 5 szabadsági foka van,
- homogén mátrix,
- SVD-je során előálló értékek közül kettő egyenlő, a harmadik pedig 0.

7.2.5. Az F mátrix számítása

A sikeres rekonstrukcióhoz szükséges F mátrix előállítására több módszer is kínálkozik. A következőkben ezeket részletezzük, hogy végül eldönthessük, melyik a legalkalmasabb feladatunk megoldására [15 p. 279].

Ha legalább 7 pontpárt megtaláltunk a vizsgált kameraképeken, az epipoláris geometriát leíró F mátrix ezek alapján kiszámítható. Amennyiben a pontokat $x_1 = (x_1, y_1, 1)^T$ és $x_2 = (x_2, y_2, 1)^T$ formában írjuk fel, minden pontpár megad egy lineáris egyenletet F egy ismeretlenéhez. A mátrix egy pontpárhoz tartozó egyenlete tehát:

$$x_2 x_1 f_{11} + x_2 y_1 f_{12} + x_2 f_{13} + y_2 x_1 f_{21} + y_2 y_1 f_{22} + y_2 f_{23} + x_1 f_{31} + y_1 f_{32} + f_{33} = 0$$

Ahol az f vektor jelöli F elemeit. Az egyenlet kifejezhető a következő formában:

$$(x_2 x_1, x_2 y_1, x_2, y_2 x_1, y_2 y_1, y_2, x_1, y_1, 1) f = 0$$

A következő egyenletrendszert kapjuk n darab pontpár esetén:

$$A f = \begin{bmatrix} x_{21} x_{11} & x_{21} y_{11} & x_{21} & y_{21} x_{11} & y_{21} y_{11} & y_{21} & x_{11} & y_{11} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{2n} x_{1n} & x_{2n} y_{1n} & x_{2n} & y_{2n} x_{1n} & y_{2n} y_{1n} & y_{2n} & x_{1n} & y_{1n} & 1 \end{bmatrix} f = 0$$

Ahhoz, hogy az egyenletnek legyen megoldása, az A mátrix rangja legfeljebb 8 lehet. Ha rangja pontosan 8, az egyenletnek egyetlen megoldása van, és lineáris módszerekkel megoldható. Abban az esetben, ha nem rendelkezünk pontos adatokkal a pontpárokról, például valamilyen, a képeken fellépő torzítás miatt, a mátrix rangja több lehet nyolcnál. Ilyenkor minimalizációs problémát kell megoldanunk, hogy megtalálhassuk azt a mátrixot, amellyel már valós eredményt kaphatunk.

Szingularitási feltétel: Az F mátrix egyik fontos tulajdonsága, hogy rangja egyenlő kettővel, továbbá a bal- és jobboldali nulltere jelenti a két kép két epipoláris pontját. Amennyiben rangja nem kettő, a belőle számított epipoláris vonalak nem egy pontba tartanak, ami a továbbiakban számítási hibákhoz vezethet. Vagyis a kiszámított F mátrixunkat le kell cserélnünk egy olyan F' mátrixra, amely teljesíti a szingularitási megszorítást, és a legközelebb áll F -hez. Ezt az SVD módszer alkalmazásával tehetjük meg: Ha $F = U D V^T$ jelöli F SVD-jét, ahol D egy diagonális mátrix ($D = \text{diag}(r, s, t)$), amely teljesíti, hogy $r \geq s \geq t$, akkor $F' = U \text{diag}(r, s, 0) V^T$ lesz az a mátrix, amely minimalizálja az $F - F'$ Frobenius-normát.

7.2.5.1. A 7 pontos algoritmus

Minimum 7 megtalált pontpárral kell rendelkezünk, hogy megoldhassuk az $A f = 0$ egyenletet. Ha az A mátrix rangja 8, akkor az egyenlet bizonyítottan megoldható. Abban az esetben, ha az A mátrix

rangja csak 7, még mindig találhatunk megoldást, ha figyelembe vesszük a szingularitási megszorítást. Ha csak hét pontpárral rendelkezünk, az A mátrix egy 7×9 -es mátrix lesz, amelynek rangja általában 7.

$Af = 0$ egyenlet megoldása ebben az esetben egy $\alpha F_1 + (1-\alpha)F_2$ alakú kétdimenziós tér, ahol α egy skalár. Az F_1 és F_2 mátrixokat A jobb oldali nullterei alapján kapjuk meg. Ha felhasználjuk azt a feltételt, hogy F determinánsa mindig nulla, az egyenlet a következő formára hozható:

$$\det(\alpha F_1 + (1 - \alpha)F_2) = 0$$

Mivel F_1 és F_2 értékei ismertek, az egyenlet alapján α kiszámítható, melyre egy vagy három megoldást kapunk. Ezután α visszahelyettesítésével végeredményül szintén egy vagy három megoldásunk lehet az F mátrixra.

7.2.5.2. *A normalizált 8 pontos algoritmus*

A legegyszerűbb módszer az F mátrix számítására. Nem kell hozzá más, mint egy lineáris egyenletrendszer megalkotása és megoldása. Megfelelően alkalmazva kiemelkedő eredmények érhetőek el vele. A siker kulcsa a bemenő adatok megfelelő módon történő normalizálásában rejlik. A javasolt normalizálás egy eltolás és skálázás minden képen úgy, hogy a referenciapontok középpontja a koordináta-rendszer középpontjában legyen, és a pontok távolsága a középponttól egyenlő legyen $\sqrt{2}$ -vel. A pontosság érdekében a szingularitási megszorítást az F mátrixra az előtt kell alkalmaznunk, mielőtt denormalizálnánk az eredményt.

Az algoritmus lépései:

1. Normalizálás: alakítsuk át a képpontokat a következők alapján:

$$\hat{x}_{1i} = T_1 x_{1i} \text{ és } \hat{x}_{2i} = T_2 x_{2i}$$

Ahol T_1 és T_2 eltolásból és skálázásból álló normalizáló transzformáció.

2. $\hat{F}$ megtalálása a következő módon:

(a) Lineáris megoldás: a pontpárok alapján elkészített $\hat{A}$ mátrix legkisebb szinguláris értékéből.

(b) Megszorítás kikényszerítése: SVD segítségével cseréljük ki $\hat{F}$ -et $\hat{F}'$ -vel, úgy, hogy $\hat{F}'$ determinánsa nulla legyen

3. Denormalizálás: Legyen $F = T_2^T \hat{F}' T_1$. Az F mátrix az eredeti pontoknak megfelelő alpmátrix.

7.2.5.3. *„Arany középút” metódus*

Az F mátrix minél pontosabb megtalálása a feltételezett zaj modellén múlik. Ebben az esetben azt feltételezzük, hogy a képpontok koordinátáit érő zaj Gauss eloszlású. Ilyenkor azt az F -et kell megtalálnunk, amelyik minimalizálja a visszavetítési hibát (geometriai távolságot):

$$\sum_i d(x_{1i}, \hat{x}_{1i})^2 + d(x_{2i}, \hat{x}_{2i})^2$$

Ahol x_{1i} és x_{2i} a mért képpontok koordinátái a képpárnak, $\hat{x}_{1i}$ és $\hat{x}_{2i}$ pedig a valósak, melyek pontosan teljesítik az $\hat{x}_{2i}^T F \hat{x}_{1i} = 0$ egyenlőséget.

Az algoritmus lépései:

1. Kezdeti $\hat{F}$ számítása lineáris módszer alkalmazásával.
2. Kezdeti $\{\hat{x}_{1i}, \hat{x}_{2i}\}$ számítása a következőképpen:
 - (a) Kameramatrixok választása $P_1 = [I \mid 0]$ és $P_2 = [[e_2]_x \mid \hat{F} \mid e_2]$ alapján, ahol e_2 -t $\hat{F}$ alapján számítjuk ki.
 - (b) A pontpárok és $\hat{F}$ alapján háromszögeléssel becsüljük meg $\hat{X}_i$ -t.
 - (c) $\hat{F}$ -el konzisztens pontpárok találhatók a következő összefüggésekkel:

$$\hat{x}_{1i} = P_1 \hat{X}_i \text{ és } \hat{x}_{2i} = P_2 \hat{X}_i$$
3. Levenberg-Marquardt algoritmus használatával minimalizáljuk a fellépő visszavetítési hibafüggvény értéket.

Bár a minimalizálási módszer számításigényesnek tűnhet, a gyakorlatban nem lassabb más iteratív módszereknél.

7.2.5.4. RANSAC

A mátrix számítása történhet RANSAC algoritmussal („Random Sample Consensus”). Az algoritmus feladata, megállapítani egy adott adathalmaz paramétereit, miközben kiszűri az adatokból a valamilyen okból nem oda tartozó, kirívó értékeket. A számításhoz 7 pontpár adatait használja fel egyszerre, ennek előnye, hogy a kapott mátrix rangja biztosan 2 lesz, valamint, mivel az adathalmazba nem illő adatok kiszűréséhez szükséges mintavételezések száma exponenciálisan növekszik az egyszerre vizsgált pontpárok számával, így csökkenthető a számítási igény. A 8 pontot vizsgáló algoritmussal szemben annyi a hátránya, hogy így akár három lehetséges megoldást is kaphatunk az F mátrixra, így mindhármat tesztelnünk kell, hogy kiválaszthassuk a jó megoldást.

Az algoritmus működése:

1. Jellegzetes pontok keresése mindkét képen.
2. Megtalált pontok párosítása.
3. RANSAC robosztus becslés: ismételjük N -szer a következőt: (N a kívülálló adatok várható aránya alapján határozandó meg)
 - Válasszunk ki véletlenszerűen 7 pontpárt és számítsuk ki az F mátrixot. Eredményként egy vagy három megoldást kapunk.
 - Számítsunk ki a d távolságot minden feltételezett pontpárra.
 - Számítsuk ki a nem kívülállók számát F alapján, ahol a számított d kisebb egy bizonyos küszöbnél
 - Ha három megoldásunk van F -re, számítsuk ki a kívülállók számát mindegyikre, és tartsuk meg azt, amelyiknél a legkevesebb a kívülálló, egyenlő helyzetben azt, amelyiknél a legkisebb a nem kívülállók normális eloszlása.
4. Számítsuk ki újra F -et csak a nem kívülállónak megítélt pontpárok felhasználásával, minimalizálva a hibát (Levenberg-Marquardt algoritmust alkalmazva).
5. Irányított párosítás: határozzunk meg további jellegzetes pontokat a megtalált F segítségével, az epipoláris vonal környezetében.

Az utolsó két lépés addig ismételhető, amíg a pontpárok száma stabilá nem válik.

Távolság számítása: az F mátrix aktuális becslése alapján a d távolság azt mutatja meg, hogy a vizsgált pontpár mekkora hibával közelíti az epipoláris geometriát. Mérésére két lehetőségünk van: választhatjuk mértékegységként a reprojekciós hibát, vagyis a pontok F mátrix alapján történő háromszögelésével előálló eltérés nagyságát, vagy a hiba Sampson-becslésével történő közelítését. Természetesen, ha valamelyik mellett döntöttünk, az algoritmus során végig azt kell használnunk, hogy eredményeink pontosak maradjanak.

Geometriai távolság (vagyis a reprojekciós hiba) számítása:

$$\sum_i d(x_{1i}, \hat{x}_{1i})^2 + d(x_{2i}, \hat{x}_{2i})^2$$

A hiba Sampson becslése:

$$\sum_i \frac{(x_{2i}^T F x_{1i})^2}{(F x_{1i})_1^2 + (F x_{1i})_2^2 + (F^T x_{2i})_1^2 + (F^T x_{2i})_2^2}$$

Irányított párosítás: Az F mátrixunk aktuális becslésével meghatározhatunk egy sávot a második kép epipoláris vonala körül, amelyről tudjuk, hogy a pontok párja az első képen is az epipoláris vonal körül helyezkedik el. Így a pontok párosítására már gyengébb küszöbértéket is használhatunk, ezáltal több pontpár megtalálására vagyunk képesek.

7.2.5.5. F mátrix számítása speciális esetekben

Ha rendelkezünk a kamerák belső paramétereit leíró mátrixokkal, normalizálhatjuk a képek koordinátáit, ezáltal az F mátrix helyett rögtön az E mátrixot számíthatjuk ki. Mivel az E mátrix teljesíti a következő összefüggést, ezért lineáris módszerekkel, akár 8 pontpár alapján is kiszámítható:

$$x_2^T E x_1 = 0$$

A számítások közti különbség csak a végeredmény szükséges tulajdonságainak kikényszerítésében tér el. Az E mátrixnak azon kívül, hogy determinánsa nulla, még azt a feltételt is teljesítenie kell, hogy SVD-je során előálló értékek közül kettő egyenlő. Ez a következő szabály alkalmazásával érhető el: Ha az E mátrix SVD-je a következő alakú: $E = UDV^T$, ahol $D = \text{diag}(a, b, c)$ és $a \geq b \geq c$, akkor a valódi E -hez legközelebb eső $\hat{E} = U \hat{D} V^T$, ahol $\hat{D} = \text{diag}(\frac{(a+b)}{2}, \frac{(a+b)}{2}, 0)$.

Amennyiben célunk csupán a két kameramátrix kiszámítása, nem feltétlenül szükséges magát az E mátrixot is előállítanunk, ugyanis a második kameramátrix közvetlenül is előállítható az SVD módszer segítségével.

7.2.5.6. Kameramátrixok számítása

Ha már rendelkezünk a kameraképek közötti átmenetet leíró mátrixszal, segítségével kiszámíthatóak a két kamera egymáshoz viszonyított pozícióját leíró kameramátrixok, P_1 és P_2 , a következő képletek segítségével:

$$P_1 = [I \mid 0] \text{ és } P_2 = [[e_2]_X \ F \mid e_2]$$

Ezek alapján a P_1 mátrix egy egyszerű 3×4 -es mátrix, melynek főátlója csupa egyesből áll, a többi érték pedig 0 benne. P_2 bal oldali 3×3 -as almátrixát $[e_2]_X \ F$ alkotja, ahol e_2 a második kameraképen lévő epipoláris pont (ahol a kamerák térbeli pozícióját összekötő képzeletbeli vonal metszi a második kamerakép síkját), F pedig a már kiszámított alapmátrix.

A képletek alapján is megállapítható, hogy valójában csak a második kamera pozícióját számítjuk ki az elsőhöz képest.

7.2.5.7. Kameramátrixok az E mátrixból

Mint láttuk az E mátrix kiszámítható közvetlenül a párosított képpontokból, illetve egy, már meglévő F mátrixból, amennyiben ismerjük a kamerák belső paramétereit leíró mátrixokat. Az E mátrixból lehetőségünk van kinyerni a kamerák relatív pozícióját leíró mátrixokat. Ha az első kamera pozícióját tekintjük a kiindulási pontnak, csak a második kamera helyzetét kell leírnunk ehhez képest. A folyamat végén négy lehetséges megoldást kapunk a keresett mátrixra.

Ehhez fel kell bontanunk E -t S és R szorzataként, ahol S egy antiszimmetrikus mátrix, R pedig egy rotációs mátrix. Ha az E mátrixunk SVD-je során előálló érték: $U \text{ diag}(1,1,0) V^T$, akkor két lehetséges esetet kapunk E felbontására:

$$S = UZU^T$$

$$R = UWV^T \text{ vagy } R = UW^T V^T$$

Ahol a következő segédmátrixokat használtuk fel:

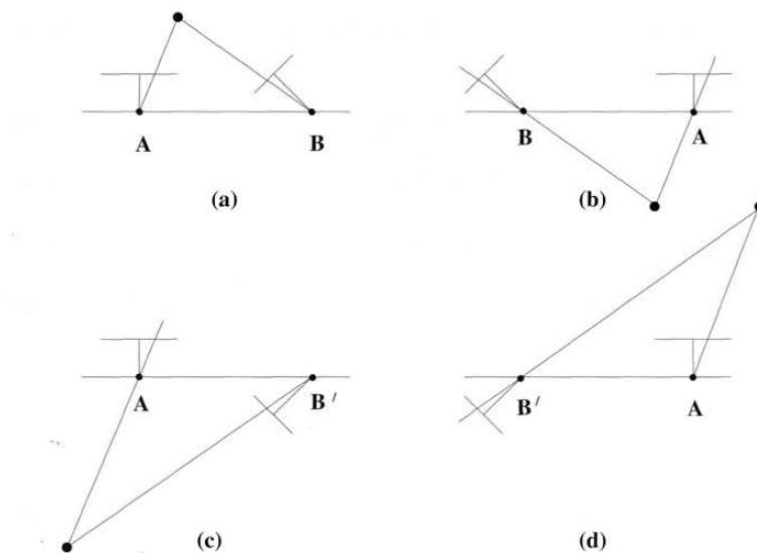
$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Bizonyítható, hogy W ortogonális, Z pedig antiszimmetrikus mátrix.

A kameramátrix leírására tehát négy lehetséges megoldást kapunk:

$$P = [UWV^T \mid +u_3] \text{ vagy } [UWV^T \mid -u_3] \text{ vagy } [UW^T V^T \mid +u_3] \text{ vagy } [UW^T V^T \mid -u_3]$$

A négy megoldás geometriai értelmezése: Az első és második megoldás mindössze az elmozdulási vektor irányában különbözik, tehát gyakorlatilag a két kamera helyzetét cseréli meg. A harmadik és negyedik megoldás közti különbség pedig egy 180° -os forgatást jelent a kamerák középpontját összekötő vonal körül.



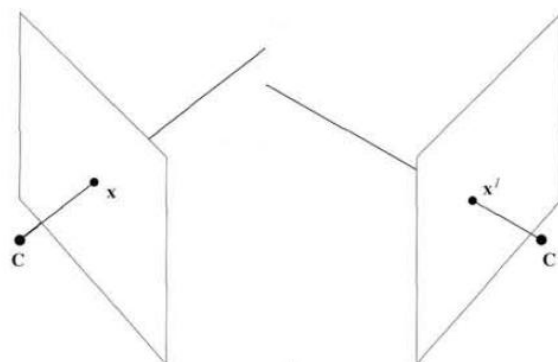
7-8. ábra: A négy lehetséges kameramátrix [15 p. 260]

A négy lehetséges megoldás közül a vizsgált térbeli pont, csak egy esetben lesz mindkét kamera látószögében, így az igazi kameramátrix egyértelműen meghatározható.

7.2.5.8. Háromszögelés

Ha már rendelkezünk az F mátrixszal, illetve a kamerák relatív pozícióit leíró P_1 , P_2 mátrixokkal, a térbeli X pont pozíciója kiszámítható a képeken lévő pontpárokból visszavetített sugarak segítségével. Mivel a kameralencsék miatt biztosan lesznek torzítások a képeken, ezért az így visszavetített sugarak gyakran nem egy térbeli pontban metszik egymást. Feladatunk tehát az, hogy a rendelkezésre álló adatok alapján a legjobb becslést adjuk a keresett pont térbeli pozíciójára [15 p. 310].

Elsődleges szempont, hogy a mérési pontatlanságot próbáljuk csak a pontpárokra korlátozni, és lehetőleg minél pontosabb adatokat kapjunk a kameramátrixokra. A pontatlanság miatt, nagy valószínűséggel nem találunk olyan X térbeli pontot a visszavetítés során, amely teljesíti az $x_1 = P_1 X$, $x_2 = P_2 X$ egyenletrendszert, és a pontpárok nem fognak megfelelni az epipoláris megszorításnak: $x_2^T F x_1 = 0$. A visszavetített sugarak csak akkor metszik egy pontban egymást, ha a pontpár teljesíti az epipoláris megszorítást.



7-9. ábra: A reprojekciós hiba [15 p. 311]

Lineáris háromszögelés: Ha tudjuk, hogy minden képen a megtalált x pontok helye: $x = PX$ és $x' = P'X$, ahol X jelöli a pont háromdimenziós helyét, P és P' pedig a kameramátrixokat, akkor az egyenletek felírhatóak $AX = 0$ formában is. Koordinátákra lebontva, a következő egyenleteket kapjuk:

$$x(p^{3T}X) - (p^{1T}X) = 0$$

$$y(p^{3T}X) - (p^{2T}X) = 0$$

$$x(p^{2T}X) - y(p^{1T}X) = 0$$

Ahol p^{iT} jelöli P mátrix sorait. Az $AX = 0$ egyenlet segítségével a következő egyenletrendszer alkotható:

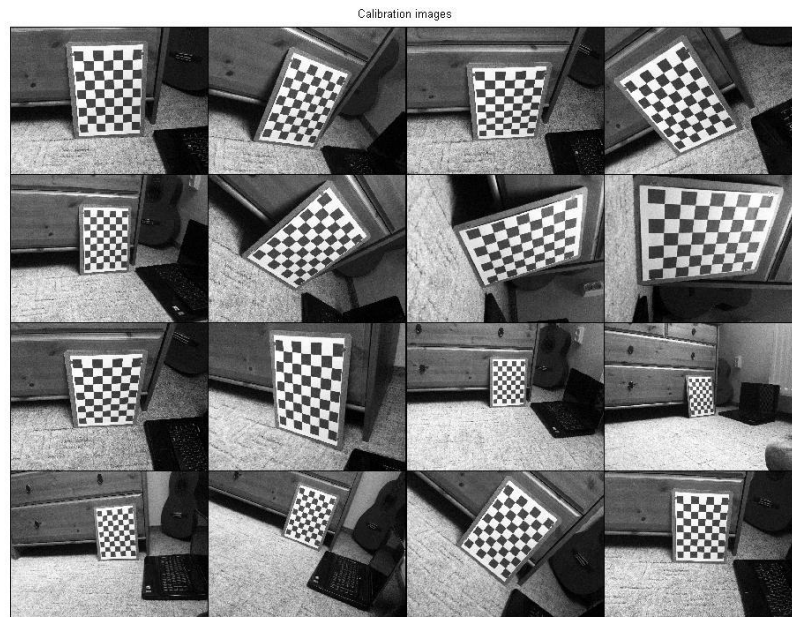
$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p^{3T} - p'^{1T} \\ y'p^{3T} - p'^{2T} \end{bmatrix}$$

Homogén módszer (DLT): A megoldást az A mátrix legkisebb szinguláris értéke alapján találja meg.

Inhomogén módszer: A megoldást során csak inhomogén egyenleteket használunk. Ha X -et homogén egyenletek rendszereként ábrázoljuk: $X = (X, Y, Z, 1)^T$, akkor $AX = 0$ megoldása négy inhomogén egyenlet és három ismeretlen rendszerévé redukálódik. Az eredmény ezután a legkisebb négyzetek módszerét felhasználva megtalálható. A módszer sajnos gyengén teljesít, ha X utolsó koordinátája egyenlő, vagy közel egyenlő nullával. Ebben az esetben ezt a koordinátát nem állíthatjuk 1-re, és az eredményeink instabillá válnak.

7.2.6. Kamera kalibráció

A modellépítéshez az előre kalibrált kamerát használó és az önkalibrációval működő algoritmust is kipróbáltuk, mivel mindkettő csak bizonyos esetekben használható. Az első módszernél a kamera belső paramétereit tartalmazó mátrixot egy előre elkészített program segítségével számítjuk ki. Ez a „Camera calibration toolbox for MATLAB” nevű algoritmusgyűjtemény bárki számára ingyenesen elérhető az interneten. A kalibrációhoz szükségünk van egy kalibrációs mintára, melynek paramétereit pontosan ismerjük. A gyakorlatban ezt egy sakktáblával tudjuk megoldani. Csupán azt kell tudnunk, hogy az egyes mezők oldalai milyen hosszúak. Első lépésként különböző szögekből kell képeket készítenünk a sakktábláról. Ezután a programba betöltjük az elkészült képeket, majd a felhasználót bevonva a műveletbe, ki kell jelölnünk a sakktábla négy sarkát minden egyes képen. Az algoritmus a többi csúcspontot automatikusan megtalálja, és ezek alapján kiszámítja a kalibrációs mátrixot. A kapott eredményt később addig finomíthatjuk, amíg a mátrix alapján visszavetített pontok hibája egy bizonyos határ alá nem kerül. A kinyert mátrix az előre kalibrált módszernél elég információt szolgáltat a metrikus modell felépítéséhez, az önkalibrációs esetben pedig jó kezdeti becslést adhat a változó fókusztávolság kiszámításához.



7-10. ábra: A kalibrációhoz használt képek

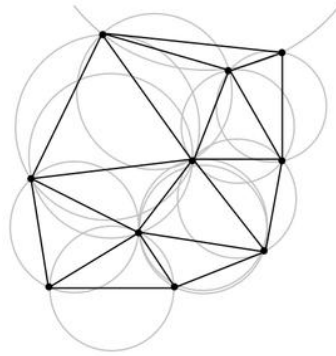
7.2.7. Pontelhelyezés a gyakorlatban

A modell megalkotásához többféle módszert is felhasználtunk. Elsőként az előre kalibrált kamerával készült képek alapján helyeztük el a pontokat a térben. Ebben az esetben a probléma megoldása a következő lépésekből áll: a megtalált pontpárok alapján a 7 pontos algoritmus segítségével RANSAC metódust alkalmazva kiszámítjuk az F mátrixot, majd ezt az „Arany középút” módszerrel tovább finomítjuk. Az eredményből a kamerák belső paramétereit leíró mátrixszal előállítjuk az E mátrixot, amelyből kinyerhetjük a kamerák pozícióit leíró mátrixokat. Ezeket a kameramátrixokat használjuk fel a pontok térbeli pozícióját visszaadó háromszögeléshez. A módszer szemmel láthatóan szép eredményeket produkál, egyetlen hátránya, hogy előre kell ismernünk a kamera belső paramétereit valamint, hogy ezek a paraméterek változatlanok maradjanak, fix fókusztávolsággal kell dolgoznunk.

A másik módszer önkalibrációra épül, és a metrikus modellt lépésenként építi fel. Először elkészíti a jelenet projektív modelljét, majd egy kezdeti hipotetikus kalibrációs mátrixot pontosít, minden képkockához megkeresve az aktuális fókusztávolságot. Az elkészült mátrix segítségével előállítja azt a hasonlósági transzformációt, amelyet a projektív modellen alkalmazva végül megkapjuk a jelenet metrikus modelljét. Az algoritmus hátránya, hogy az önkalibráció nem ad olyan pontos eredményt, mint az előre kalibrált módszer, így gyakran kapunk elfajuló eredményeket.

7.3. Vizualizáció javítása: testháló és textúrázás

A jobb vizualizáció érdekében a kiszámított pontokból felépítünk egy olyan testhálót, amely jobban visszaadja az eredeti jelenet struktúráját. Kiinduló ötletünk a pontok térbeli *Delaunay-háromszögelésén* alapszik. Az algoritmus előállítja a pontok által alkotott test konvex testhálóját, a későbbiekben pedig ezt kell tovább finomítanunk, ha nem szeretnénk, hogy a konkáv testek részletei elveszenek. A felesleges részek eltávolítása az *alfa-háló* segítségével történik. A használt küszöbértékkel adhatjuk meg, hogy egymástól milyen távol eső pontokat szeretnénk összekötni.



7-11. ábra: Delaunay-háromszögek [15]

7.3.1. Delaunay-háromszögek

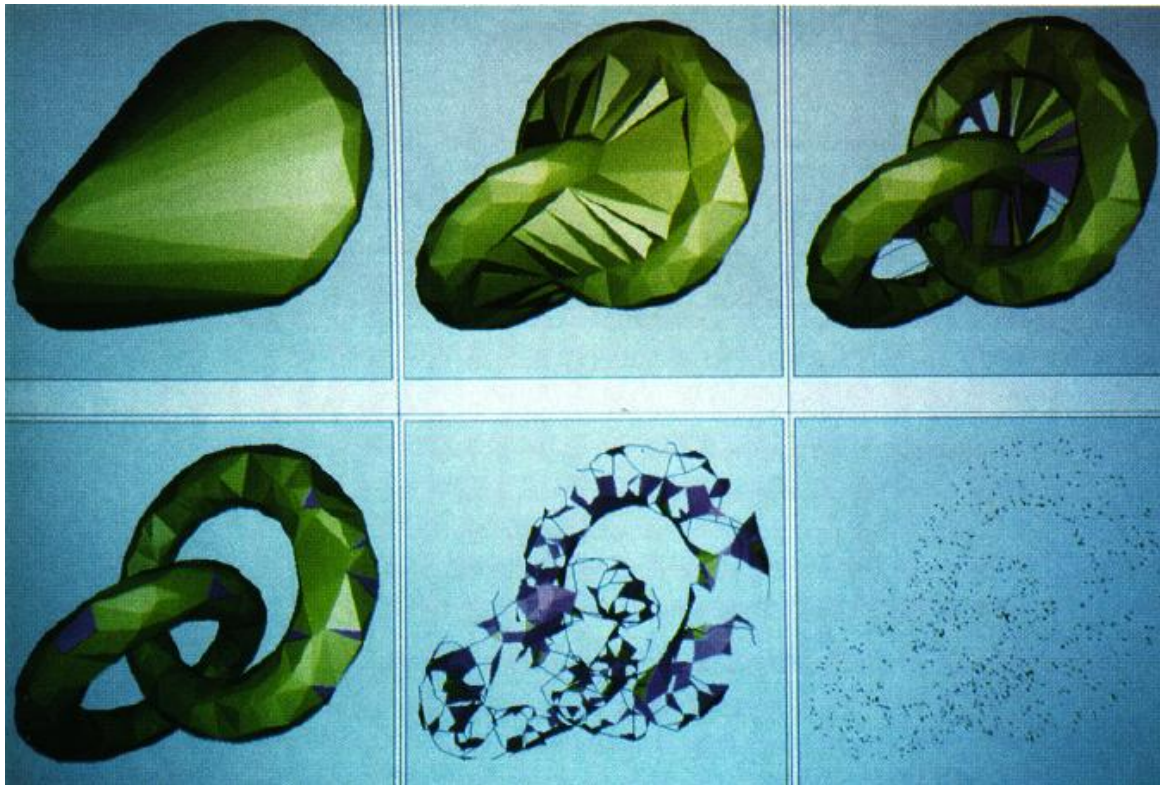
Egy P pont halmaz Delaunay-háromszögelése [16 p. 225], egy olyan egyedi háromszögelés, amelynek elemei köré írt körei a pont halmaz pontjait legfeljebb a határukon tartalmazzák, tehát egy pont sincs ezeken a körökön belül. Fontos tulajdonsága, hogy maximalizálja a háromszögek legkisebb szögét, így kerül a keskeny, vékony háromszögeket. A háromszögelés a pont halmaz konvex burkát állítja elő. A megvalósítás során a fejlesztőkörnyezett beépített funkcióját használtuk fel a háromszögek számítására, amely a kétdimenziós háromszögelést terjeszti ki háromdimenziósra, így kapjuk meg a térbeli pontok által kifeszített test konvex hálóját.

7.3.2. Alfa-háló

Mivel a Delaunay-háromszögeléses módszerrel előállított háló minden esetben konvex, a térbeli pontjaink által határolt objektum pedig nem feltétlenül az, így a kapott testhálót érdemes tovább finomítanunk, hogy ne veszítsünk el fontos részleteket. Erre nyújt jó lehetőséget az alfa-háló, amely egy, már meglévő háromszögelésből távolít el éleket, egy α paraméter alapján [17 p. 43].

Az algoritmus lényege, hogy az eredeti hálóból azokat az éleket távolítja el, amelyek végpontjai közé egy α sugarú kört (a térben gömböt) el tudunk helyezni. Ezt akár úgy is elképzelhetjük, mintha egy α sugarú labdát görgetnénk végig a pontokon. A modell részletességét az α paraméter változtatásával tudjuk növelni. Ha $\alpha=0$, akkor eredményül az eredeti pont halmazt kapjuk vissza, $\alpha=\infty$ esetén pedig a pont halmaz konvex burkát. Az alábbi képeken a paraméter változtatásának hatása figyelhető meg: az első képen értéke végtelen, az utolsón pedig nulla. A köztes állapotokban jól megfigyelhető, hogyan körvonalazódnak az objektum konvex háló által elrejtett részletei.

A végeredményül kapott háló, mivel nem ragaszkodik a konvexitáshoz, jobban kiemeli a részleteket, akár olyan helyeken is kialakulhatnak lyukak a modellben, amelyek kívülről nem érhetőek el. A testháló gyakorlati felépítése során a legnagyobb problémát az α paraméter megválasztása okozta. Mivel a Delaunay-háló automatikusan generálódik, csak az alfa-hálónál van szükség külső beavatkozásra. A paramétert úgy kell megválasztanunk, hogy a kapott modell minél jobban tükrözze az eredeti jelenetet. Ha túl nagyra választjuk, egyes konkáv részleteket elveszíthetünk, ha túl kicsire, nem kapunk összefüggő hálót.



7-12. ábra: Az α paraméter változása [17 p. 45]

7.3.3. Textúrázás

Bár a pontok alapján kiszámított testháló már jól jellemzi az eredeti környezetet, a szemlélő felhasználó számára tovább gazdagíthatjuk a modellt, ha a készült képek alapján „kiszínezzük” a hálónkat. Mivel a testháló háromszögekből áll, és tudjuk ezek térbeli pontjairól, hogy melyik képpont alapján kerültek kiszámításra, nincs más dolgunk, mint az eredeti képből ezek alapján kivágni egy háromszöget, és a térbeli hálóra illeszteni azt. A problémát az okozza, hogy mivel a modell több kamerakép alapján készült, meg kell találnunk azt a képet, amelyiket a hálóra kivetítve, az eredeti jelenetet a legélethűbben vizualizálja. Megoldásként többféle módszer közül is választhatunk: [18 p. 36]

Az első és legkézenfekvőbb, hogy fogjuk meg a legelső kameraképet, amelyikkel rendelkezünk, és minden háromszöget ez alapján textúrázzunk. A fő probléma ezzel a módszer, hogy a hálót alkotó háromszögek közül nagy valószínűséggel nem mindegyik látszik ideális szögből az első képkockán, legfőképp, ha a képek közötti elmozdulás nagy. Ebből rögtön adódik is a második módszer alapfelvetése, miszerint a testháló minden háromszögét megvizsgáljuk és kiszámítjuk a képeket készítő kamerákkal bezárt szögüket. A textúrát az a kamerakép alapján választjuk ki, amelyikre az adott oldal a legmerőlegesebb. Ez elméletben biztosítja, hogy a legmegfelelőbb képrészlet kerüljön egy adott oldalra, gyakorlatban viszont a háló pontatlanságai miatt nem feltétlenül az lesz a legrészletesebb textúrát adó kamera, amelyik a legmerőlegesebb a háromszögre. A harmadik módszer abból indul ki, hogy mivel szeretnénk a legrészletesebb képrészletet ráfeszíteni a hálóra, vizsgáljuk meg a háromszögek méretét minden kameraképen. Tekintve, hogy a térbeli háromszögünk fix méretű, a képkockákon pedig a hozzá tartozó képrészlet változó lehet a kamera orientációjától függően, az a kép rejti a legtöbb részletet, amelyiken a kivágott háromszög mérete a legnagyobb. Az

ideális megoldáshoz tehát minden kameraképre ki kell számolnunk a térbeli háromszög képének területét, és azt a képkockát használjuk fel, amelyiken ez a terület a legnagyobb.

A megvalósítás során a három módszer közül a legelsőt választottuk, alacsony számításigénye miatt. A modellező rendszer ezen részét is a MATLAB programozási nyelv segítségével sikerült megvalósítani, amely a háromdimenziós műveletekhez az OpenGL algoritmusgyűjtemény funkcióit használja.

7.4. Kiegészítések

A számítások során sokszor hivatkozunk egyes bonyolultabb matematikai módszerekre, ezek megértéséhez szolgál segítségül a következő néhány kiegészítés.

7.4.1. DLT (Direct Linear Transformation) algoritmus

A DLT algoritmus lineáris megoldást nyújt két kétdimenziós pontthalmaz közötti H hasonlósági mátrix felírására [15 p. 88]. Ha a pontpárokat x_i és x'_i -vel jelöljük, a köztük lévő összefüggés a következő formában írható fel: $x'_i = Hx_i$. Fontos, hogy az egyenlet homogén vektorokból áll, így Hx_i és x'_i harmadik vektorai nem feltétlenül egyenlők. Irányuk megegyezik, de nagyságuk valamilyen skálázási faktoral eltérhet. Az egyenlet így felírható a vektorok keresztszorzataként is:

$$x'_i \times Hx_i = 0$$

Az a forma lehetővé teszi, hogy H -t lineáris módszerekkel megtalálhassuk. Ha H j -edik sorát h^j -vel jelöljük, az összefüggést felírhatjuk a következő formában:

$$Hx_i = \begin{pmatrix} h^{1T}x_i \\ h^{2T}x_i \\ h^{3T}x_i \end{pmatrix}$$

Alkalmazva az $x'_i = (x'_i, y'_i, w'_i)$ felbontást, a keresztszorzat megadható más formában is:

$$x'_i \times Hx_i = \begin{pmatrix} y'_i h^{3T}x_i - w'_i h^{2T}x_i \\ w'_i h^{1T}x_i - x'_i h^{3T}x_i \\ x'_i h^{2T}x_i - y'_i h^{1T}x_i \end{pmatrix}$$

Mivel $h^{jT}x_i = x_i^T h^j$ minden $j = 1..3$ -ra, így három egyenletrendszeret kapunk H értékeire, amely a következő alakban írható fel:

$$\begin{bmatrix} 0^T & -w'_i x_i^T & y'_i x_i^T \\ w'_i x_i^T & 0^T & -x'_i x_i^T \\ -y'_i x_i^T & x'_i x_i^T & 0^T \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0$$

Ezek az egyenletek átírhatóak $A_i h = 0$ formájúra, ahol A_i egy 3×9 -es mátrix, és h egy 9-es vektor, amely H elemeiből épül fel.

$$h = \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix}, H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Ahol h_i jelöli h i -edik elemét.

Minden megtalált pontpárból két független egyenlet állítható fel H értékeire. Ha négy pontpárral rendelkezünk, megkapjuk az $Ah = 0$ egyenletrendszert, ahol A az egyenlet együtthatóiból készült mátrix, melynek A_i sorait a pontpárok alapján írhatjuk fel, valamint h jelöli H ismeretlen értékeinek vektorát. Erre a h vektorra keresünk nem triviális megoldást (mivel $a \cdot h = 0$ megoldást nem tudjuk felhasználni). Az A mátrix rangja mindenképpen 8 lesz, így rendelkezik egydimenziós nulltérrel, amely biztosítja, hogy h -nak legyen megoldása.

7.4.2. SVD (Singular Value Decomposition) algoritmus

Az algoritmus egyike a legjobban használható mátrix dekompozíciós módszereknek, különösen numerikus számítások esetén [15 p. 585].

Adott négyzetes A mátrix esetén, a mátrix SVD-je egy faktorizáció, a következő képlet alapján: $A = UDV^T$, ahol U és V ortogonális mátrixok, D pedig nem negatív elemekből álló diagonális mátrix. A dekompozíciót úgy is végrehajthatjuk, hogy D elmei csökkenő sorrendben legyenek, a későbbiekben pedig feltételezzük, hogy ez mindig így is történik. Így V -nek azon oszlopa, amely a legkisebb szinguláris értékhez tartozik, V utolsó oszlopa lesz.

Természetesen a módszer nem négyzetes mátrixokra is alkalmazható, azzal a módosítással, hogy ha a mátrixnak több oszlopa van, mint sora, négyzetes mátrixszá egészítjük ki nullákat tartalmazó sorok hozzáadásával.

Szinguláris értékek és sajátértékek: a módszer során előálló D mátrix elemei nem negatívak, és ezek jelentik az A mátrix szinguláris értékeit. Ezek ugyan nem egyeznek meg az A mátrix sajátértékeivel, de felfedezhető kapcsolat közöttük. A kiinduló $A = UDV^T$ egyenletből felírhatjuk az $A^T A = VDU^T UDV^T = VD^2 V^T$ összefüggést. Mivel V ortogonális, így $V^T = V^{-1}$ és $A^T A = VD^2 V^{-1}$. Ez a sajátértékek definíciójának egyenlete, tehát D^2 értékei valójában $A^T A$ sajátértékeivel, valamint V oszlopai $A^T A$ sajátvektoraival egyenlők. Egyszerűbben fogalmazva A szinguláris értékei $A^T A$ sajátértékeinek négyzetgyökei.

7.4.3. Givens-forgatás és mátrixok RQ dekompozíciója

A háromdimenziós Givens-forgatás a három koordinátatengely egyike körüli forgatást jelenti, a következők alapján [15 p. 579]:

$$Q_x = \begin{bmatrix} 1 & & \\ & c & -s \\ & s & c \end{bmatrix} Q_y = \begin{bmatrix} c & & s \\ & 1 & \\ -s & & c \end{bmatrix} Q_z = \begin{bmatrix} c & -s & \\ s & c & \\ & & 1 \end{bmatrix}$$

Ahol $c = \cos(\vartheta)$, $s = \sin(\vartheta)$ valamilyen ϑ szögre, továbbá az üres helyek nullákat jelölnek.

Ha egy 3×3 -as A mátrixot jobbról megszorozzuk például a Q_z mátrixszal, A utolsó oszlopa változatlan marad, első két oszlopa pedig az eredeti oszlopok lineáris kombinációjával cserélődik ki. A használt ϑ szög megválasztható úgy, hogy az első két oszlop bármelyik értéke végül 0 legyen.

Például, ha A_{21} értéket nullára szeretnénk állítani, a következő egyenletet kell megoldanunk: $ca_{21} + sa_{22} = 0$. A megoldás: $c = -a_{22} / (a_{22}^2 + a_{21}^2)^{1/2}$ és $s = a_{21} / (a_{22}^2 + a_{21}^2)^{1/2}$. A megoldáshoz szükséges, hogy $c^2 + s^2 = 1$ egyenlet teljesüljön, ez minden esetben igaz, mivel $c = \cos(\vartheta)$ és $s = \sin(\vartheta)$.

Az *RQ dekompozíciós* módszer lényege, hogy a mátrixunk alsó felét egyesével kinullázzuk, Givens-forgatásokat alkalmazva. Vegyük egy 3×3 -as A mátrix dekompozícióját a következő alapján: $A = RQ$, ahol R egy felső háromszögmátrix, Q pedig egy forgatási mátrix. A folyamat három lépésből áll.

Minden lépés egy jobbról szorzást jelent, a megfelelő Givens-mátrixszal, úgy, hogy A egy választott értékét nullára változtassa. Természetesen a lépéseket úgy kell végrehajtanunk, hogy a már nullázott értékek később ne módosuljanak.

Az RQ dekompozíciós módszer algoritmusát egy 3×3 -as A mátrix esetén, Givens-forgatásokat alkalmazva:

1. Szorzás Q_x -szel, így A_{32} értékét kinullázzuk.
2. Szorzás Q_y -nal, így A_{31} értéke is nulla lesz. Ez a művelet nem érinti A második oszlopát, így A_{32} nulla marad.
3. Szorzás Q_z -vel, így A_{21} értéke nulla lesz. Az első két oszlopot saját lineáris kombinációjukkal cseréljük ki, így A_{31} és A_{32} továbbra is nulla marad.

A művelet eredményeképpen megkapjuk R -t, vagyis $R = A Q_x Q_y Q_z$ és $Q = Q_x^T Q_y^T Q_z^T$ jelöli a forgatást, valamint a forgatáshoz használt ϑ_x , ϑ_y , ϑ_z szögek használhatók a forgatás paramétereiként.

8. Megjelenítés

A program megjelenítési részének két feladatot kell ellátnia: egyrészt láthatóvá kell tennie a roboton lévő kamera által felvett képet, a navigáció felhasználó általi végrehajtásához, másrészt meg kell jelenítenie a képek alapján addig felépült háromdimenziós modellt.

A modellépítés fázisait közvetlenül végigkövethetjük a használt MATLAB fejlesztőkörnyezet funkcióit használva. Ez a megjelenítés alapvetően az OpenGL algoritmusgyűjteményt alkalmazza. Az OpenGL funkciók közvetlen használatára így nincs lehetőségünk, gyakorlatilag a fejlesztőkörnyezet leegyszerűsíti számunkra a háromdimenziós vizualizációt.

9. Tesztelés

9.1. Első tesztelések notebook-notebook adat átvitel

Első lépésként elkészült a kommunikációs modul. Tesztkörnyezet: vezetékes hálózati kapcsolat két számítógép között. Mind a két számítógépen elindítottuk a tesztelő alkalmazást, ami a modult felhasználja. A tesztelési eljárás menete: kapcsolat felépítése, majd a felépített csatornán a programba egy inputmezőbe gépelést követően megjelent a másik félnél a begépelt szöveg. A tesztelés sikeres volt.

9.2. Második teszt: notebook-notebook képátvitel

Második lépésként az adatküldő modul került kibővítésre, hogy képes legyen kezelni a képátvitelt. Tesztkörnyezet: vezetékes hálózati kapcsolat két számítógép között. Mind a két gépen elindítottuk az alkalmazást, majd felépítettük a kapcsolatot. A tesztelés során vezetékes hálózaton a következő problémákat tapasztaltunk: a képek szétestek a megjelenítés során, valószínűleg egy nem megfelelő csomagküldési eljárás használata miatt. Emiatt egy alternatív képküldési eljárást kellett kidolgozni, ami a szerializáció segítségével küldi el a képeket. A modult módosítva, a vezetékes hálózati kapcsolaton már nem tapasztaltunk hibás működést. Második lépésként WIFI-s kapcsolatot használva történt a tesztelés. A feltárt hiba a következő: túlzott adatforgalmi igény merült fel a kapcsolatban, amit a WIFI nem volt képes kiszolgálni. A probléma a szerializáció által létrehozott objektum mérete, egy 640*480 pixel felbontású kép esetén 640kB méretű lett. Ez nagyon alacsony képátviteli sebességet eredményezett. A szerializációt ilyen formában nem célszerű használni. Megoldás: A memóriába létrehozni egy jpg kódolású képet, és ezt a képet átküldeni a csatornán. A csatornát minden egyes képátvitel esetén újra kell építeni és lebontani. Ezzel az eljárással már sikerült elérni a kellő képátviteli sávszélességet.

9.3. µC program tesztelés:

Az AVR Studio lehetőséget biztosít emulált tesztelésre, aminek segítségével az emulált processzor regisztereinek lehet értékeket adni. A program tesztelése ennek a környezetnek a segítségével történt.

Tesztelési környezet: számítógép soros kapcsolattal illesztett µC, a kapcsolati tesztre használt egyszerű echo program a µC-ben. Az éles teszt esetén már kommunikációs problémákba ütköztünk. A µC soros kommunikáció nem adott megfelelő eredményt, mert a teszteseteket nem sikerült lefuttatni. A probléma a következő volt: a számítógépen beírt karaktereket nem adta vissza, illetve nem azokat a karaktereket küldte vissza. Megoldás: egy külső órajel generátort kell illeszteni a µC-hez, ami stabilizálta a rendszer órajelét és ezáltal a tesztek is sikeresen végrehajtottak.

9.4. Üzenetküldési teszt a robotnak WIFI-ről

Teszt környezet: hálózati kapcsolat két számítógép között (WIFI). Valamint az egyik számítógépre rákötve a µC soros porton keresztül. A teszt arra irányult, hogy adatokat küldünk a távoli gépről a µC-nek és a visszaérkező adatok az elvártaknak megfelelően jelenjenek meg a küldő oldalán, ezáltal a kiépített csatorna stabilitását tudtuk vizsgálni. A teszt sikeres volt, nem tapasztaltunk adatvesztést.

9.5. µC továbbfejlesztése motorvezérlési tesztel

Emulációs teszt során a megfelelő regiszterértékek beállítása után a dekódolási eljárás végeredményét figyelve folyt a tesztelés. Emulációs tesztek a *Proteus VSM for AVR* nevű

programcsomag segítségével is teszteltük. A programban lehetőség van egy áramkör szimulációjára. A felépített virtuális áramkörhöz kapcsolhatunk virtuális oszcilloszkópot, aminek a segítségével a kimeneti jelszinteket tudtuk virtuálisan figyelni. A szimuláció az elvárt eredményt hozta.

Éles tesztkörnyezet: számítógéphez kapcsolt μC és motor illesztve a μC -hez. A tesztelés arra irányult hogy a számítógépről kiküldött billentyűkódoknak megfelelően a motor forgása jó irányú, valamint megfelelő sebességű legyen. A tapasztalt probléma: a motor nem megfelelően forog. Mivel a dekódolás megfelelőnek látszik a szimuláció alapján, az adatok kiolvasása, illetve nem megfelelő változóba történő beolvasás lehet a probléma. A regiszter kiolvasási eljárást ellenőrizve, változó típus hibára lettünk figyelmesek. Char típusú kiolvasás történt, de uint8 (byte) típusú adatok kerültek elküldésre. A rossz előjel kezelési hibát kijavítva, a teszt újra futtatása után további probléma nem merült fel.

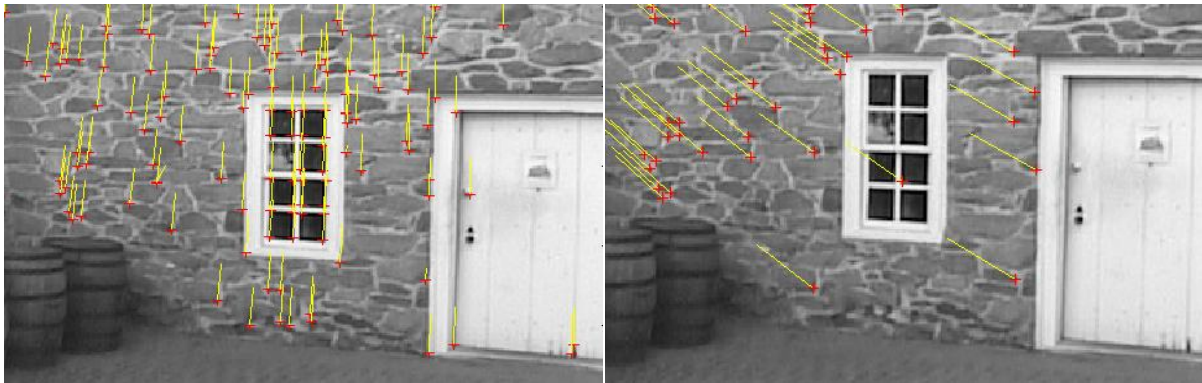
9.6. Irányítási modultesztelés (joystick)

Tesztelési környezet: két számítógép hálózati kapcsolatban, az egyik gépre kötve egy joystick és a felhasználó oldali szoftverkörnyezet futtatva. A másik számítógépen a robot oldali szoftver és az emulátor futott. A modul tesztelése alapvetően a helyes adatkinyerésre és a dekódolásra irányult. A tesztelés során kiderült, hogy a joystickról kiolvasott értékeket fordított tengely szerint programoztuk le. A hibát javítva, a további tesztelés sikeres volt.

9.7. Jelleghetes pontkereső és pontkövető tesztelése

Az affin módszeren alapuló pontkövető a FAST és SURF algoritmusoknál megbízhatóbb és pontosabb eredményeket produkál, kevesebb téves párosítással. Ha a két képkocka közti elmozdulás bizonyos határokon belül marad, az algoritmus által szolgáltatott pontpárok stabil alapul szolgálnak a modellépítési feladat további fázisai számára. Valamint, mivel a későbbiekben az F mátrix számítására a RANSAC módszert alkalmazzuk, az esetleg előforduló téves párok nem okoznak problémákat, mivel az algoritmus a túlságosan kirívó adatokat kívülállónak osztályozza, így ezek a végeredményt nem befolyásolják. Természetesen a téves pontpárok arányának egy bizonyos küszöbnél alacsonyabbnak kell lennie. Ezt az arányt legegyszerűbben úgy tarthatjuk alacsonyan, ha a képeket készítő kamerák közötti elmozdulás kicsi, hisz az algoritmus könnyebben megtalálja a közelebbi párokat, vagy ha a kamerák által okozott torzításokat minél jobban kiküszöböljük.

Az alábbi példák (9–1. ábra) a módszer pontosságát mutatják, a különböző távolságban lévő kamerák által készített képeken.



9-1. ábra: Pontkövetési példák

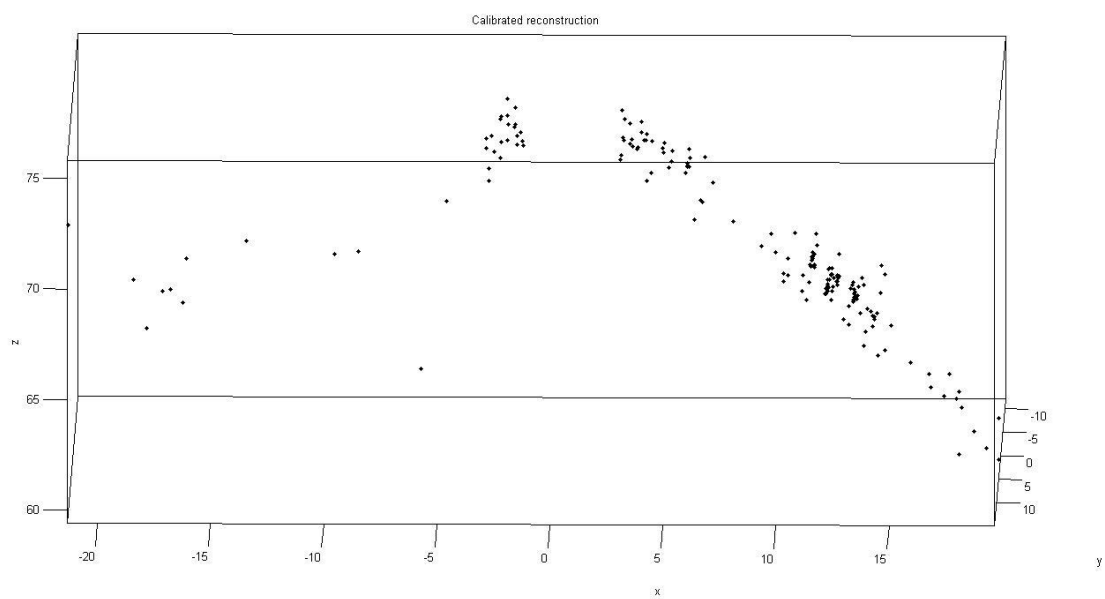
9.8. Pontok elhelyezése a térben

Mivel a pontkövetési algoritmus többnyire megbízható eredményeket adott, a pontpárok térbeli pozíciójának kiszámításánál is jó értékekre számítottunk. A gyakorlati tesztek során több problémával is szembe kellett néznünk. Mind az önkalibrációs, mind az előre kalibrált esetben előfordultak elfajuló eredmények, amikor a rekonstruált pontok szemmel láthatóan nem jó pozícióba kerültek, például a modellünk „lapos lett”, mintha minden pont egyetlen síkon helyezkedne el. Ez megfigyeléseink alapján tipikusan két esetben fordult elő: ha az F mátrix számításához használt kameraképek egymáshoz túl közel vannak, vagyis a kamerák közötti relatív távolság kicsi, vagy ha a használt kalibrációs mátrix valami miatt rossz értékeket tartalmaz, ez utóbbi tipikusan az önkalibrációnál fordulhat elő. Az is megfigyelhető, hogy a használt önkalibrációs módszer csak bizonyos speciális mozgások jelenléte esetén ad megbízható eredményt. A kamerák közti kis távolság okozta problémát egy egyszerű módszerrel küszöböltük ki: folyamatosan figyeljük a megtalált pontpárok közötti átlagos távolságot, és ha ez eléri egy bizonyos nagyságot, csak akkor kezdjük el kiszámítani az F mátrixot. Ezzel egyrészt számítási időt is megtakarítunk, másrészt csökken a veszélye annak, hogy elfajuló eredményt kapjunk.

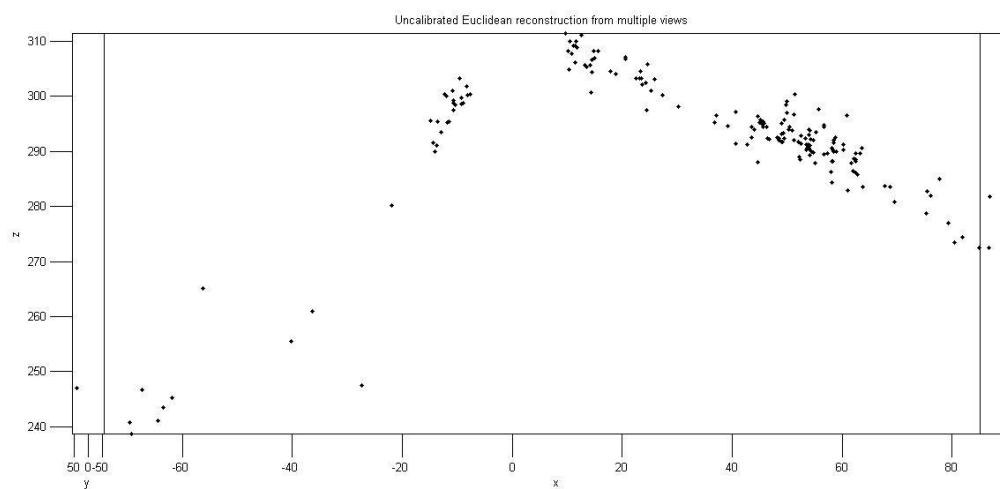
Az első példában a módszer eredményeit mutatjuk be egy felhasznált minta képsorozaton. A 9-2. ábrán látható egy mintakép a jelenetből. A 9-3. és 9-4. az előre kalibrált és önkalibrációs módszer eredményeit mutatják. Látható, hogy a két metódus közel azonos eredményt produkál, és mindkettő jól visszaadja az eredeti jelenet struktúráját.



9-2. ábra: Kültéri tesztjelenet



9-3. ábra: Kalibrált módszer eredményei (felülnézet)



9-4. ábra: Önkálibrációs módszer eredményei (felülnézet)



9-5. ábra: Kültéri testtjelenet



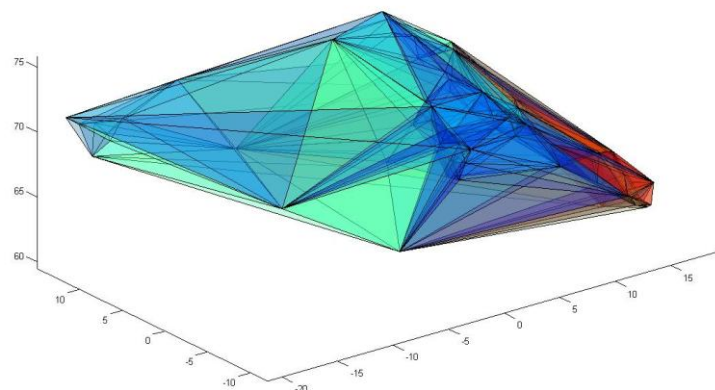
9-6. ábra: Az előre kalibrált módszer eredményei (felülnézet)

Második példaként egy általunk készített képsorozaton elvégzett tesztek eredményeit mutatjuk be. A 9-5. ábrán látható a jelent egy mintaképe. A kamera kalibrációja előzetesen megtörtént egy sakktábla minta használatával, az így kinyert kalibrációs mátrixot használó előre kalibrált módszer eredményei a 9-6. ábrán láthatóak. A pontok jól detektálhatóan tükrözik az eredeti jelenet felépítését. Az önkalibrációs módszer erre a jelenetre sajnos teljesen elfajuló eredményeket adott, jól mutatva, hogy nem minden esetben alkalmazható biztonságosan.

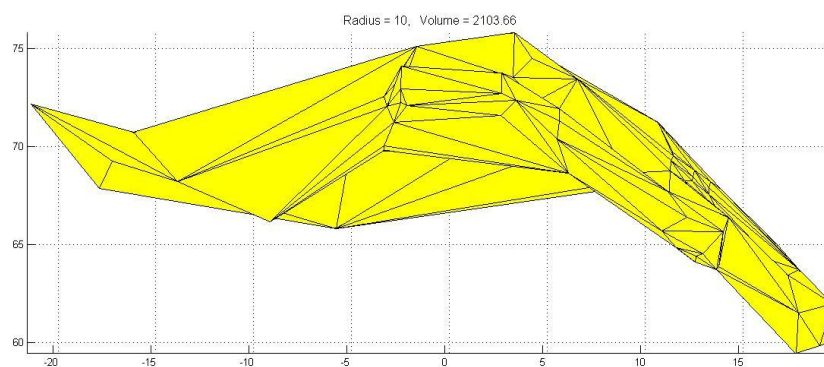
9.9. Testháló generálása

A pontokra épülő testháló megalkotása során a pontok konvexitása okozza a legnagyobb problémát. Első lépésként megalkotjuk a pontokra épülő Delaunay-testhálót, amely minden esetben teljesen konvex lesz. Konkáv ponthalmaz esetén ez komoly problémákat okozhat a vizualizációban, ezért a modellt tovább finomítjuk az alfa-háló segítségével. Ennél a lépésnél sarkalatos kérdésnek bizonyult az α paraméter megválasztása. Mivel a 0 érték által előállított modell a kiinduló pontjainkat jelenti, a végtelen érték pedig a konvex hálót, úgy kell megadnunk α -t, hogy a modellünk lehetőleg még összefüggő legyen, és minél több konkáv részletet ki tudjon emelni az eredeti jelentből.

A következő képek (9-5, 9-6) a hálót előállító algoritmus eredményeit mutatják az első képsorozat alapján készült pontokon. Jól látható a különbség a Delaunay- és alfa háló között, az utóbbin már körvonalazódik a jelenet valódi struktúrája.

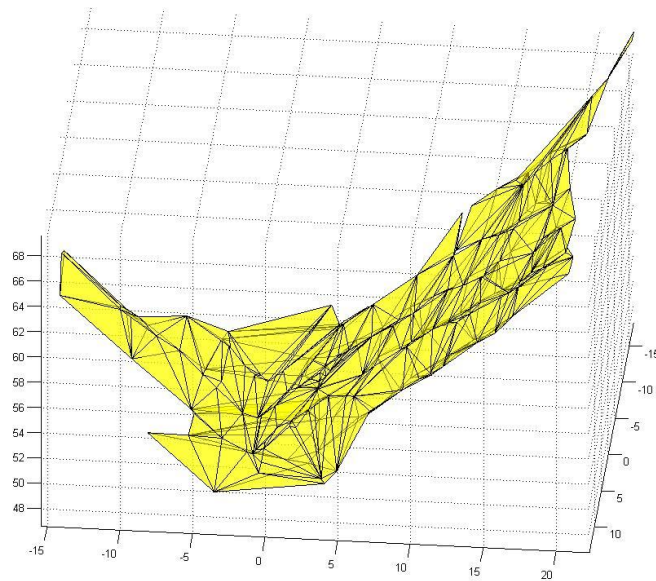


9-7. ábra: Konvex Delaunay-testháló

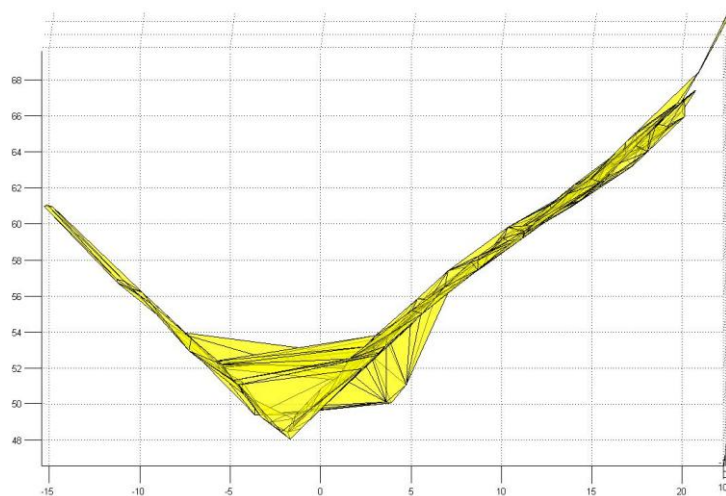


9-8. ábra: Finomított Alfa-háló (felülnézet)

A további ábrák (9-9, 9-10) a második példa során kiszámított pontok alapján elkészült alfa-hálót ábrázolják. Jól megfigyelhetőek rajtuk a jelenet sajátosságai.



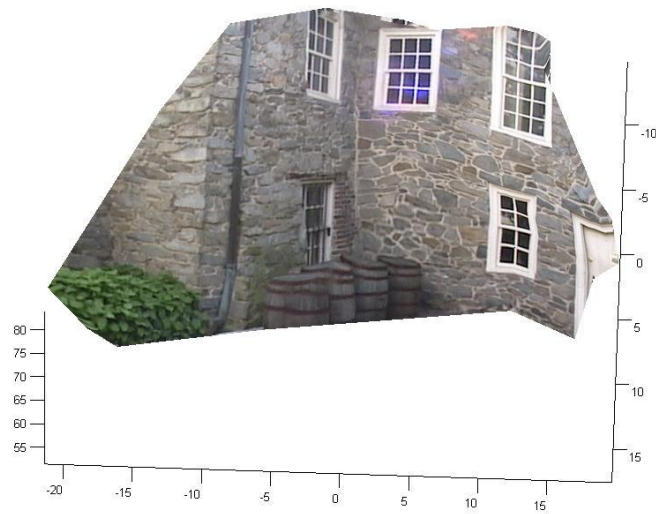
9-9. ábra: Elkészült alfa-háló



9-10. ábra: Elkészült alfa-háló (felülnézet)

9.10. Textúrázás

Az alábbi ábrák (9-11, 9-12) mutatják a végleges textúrázás által előállított modell eredményeit. A kapott textúra élethűsége valójában a felhasznált háló pontosságától függ. Minél több részletet emel ki a testháló, annál részletesebb textúrázott modell előállítására vagyunk képesek.



9-11. ábra: Az első tesztjelenet textúrázva



9-12. ábra: A második tesztjelenet textúrázva

10. Eredmények értékelése

Ebben a dolgozatban bemutatott rendszer egy, a felhasználó közreműködésével irányított robot és a robotról kiolvasható képek feldolgozását megvalósító, összetett alkalmazást írt le. A robot megvalósításánál az elektronikai vezérlés és magának a szerkezetnek az elkészítése mellett, a robotra elhelyezett eszközök megfelelő felprogramozása is cél. A kezdeti próbálkozások és első prototípusok után a hibák kijavításával egyre inkább elértük, hogy a hordozó szerkezet Bluetoothos kommunikációt használva teljesen autonóm lett. Az első eredményeket hordozható számítógépekkel kaptuk. A fejlesztések kezdetekor igen egyszerűnek tűnt a kommunikáció megvalósítása a felek között, mert csak egy univerzális modul megalkotását tartottuk szükségesnek, amit .NET-es környezetben készítettünk. Az itt adódó adatátvitel hibáinak megértése után sikeresen készítettünk egy hálózati kommunikációra alkalmas modult. Amikor már a fejlesztés más területein is előrehaladást értünk el, a .NET-es környezet mellett MATLAB illetve Objective-C is előtérbe került. A mikrokontroller programját C nyelven készítettük, és az adatkommunikációt is ezzel oldottuk meg – tesztelések után, végül – sikeresen. Az Objective-C alkalmazására a választott mobil rendszer miatt volt szükségünk, mivel nem volt lehetőség akkora hordozót építeni, amire egy hordozható számítógép elhelyezhető lenne, alternatívák felkutatásába kezdtünk. Ilyen lehetőség volt egy iPhone alkalmazása. A kezdeti fókuszproblémák után - ami az automata fókusz miatti feldolgozási nehézségből adódott - a rendszerbe integrálás sikeresen lezajlott, és ezzel a lépéssel egy kisméretű eszközt alkothattunk meg robotoldalon. A vezérlő modul esetében a kezdeti meghajtó motor alkatrész típusának bizonytalansága miatt kétféle motorvezérlést is kidolgoztunk a későbbi bővíthetőség céljából. Az irányíthatóság megkönnyítésére egy Direct Inputot - amit egy DirectX-es kiegészítő segítségével értünk el - kezelni képes modult is kifejlesztettünk, így sokkal pontosabb irányítás vált lehetővé. A joystick használatára a vezérlő rendszert is fel kellett készíteni. A webkamera képek kiolvasására is több lehetőséget vizsgáltunk. Ezek általában valamilyen OpenCV-re épülő C# os wrapperek voltak. Az elkészült képek összevetve az iPhone által készített képekkel zajosabbak voltak és a szín összetettségük sem mozgott olyan széles spektrumon mint a mobil eszköz esetén.

Az elkészített rendszer modellezési modulja a lehetőségekhez mérten képes a környezet élethű háromdimenziós vizualizációjának elkészítésére. A megvalósításhoz szükséges algoritmusok közül elsőként kiemelendő a jellegzetes pontok keresése és követése. A funkció megbízhatóan működik nagy távolságok esetén is, gyakorlatilag a tesztek során nem produkált hibákat. A pontok térbeli elhelyezése kulcskérdés az élethűség szempontjából. A tapasztalatok alapján az előre kalibrált módszer hatékonyabb az önkalibrációnál, egyetlen hátránya, hogy a belső paraméterek meghatározásához előzetes tesztek és számítások kell végeznünk azzal a kamerával, amellyel a későbbi modellezéshez szánt képek készülni fognak, továbbá biztosítanunk kell, hogy ezek a paraméterek változatlanok maradjanak. Az önkalibrációs módszer saját határai miatt csak a tesztek kis részében produkált helyes eredményt, tökéletesítésével viszont a program kevesebb korláttal működhetne biztonságosan. A testháló generálásánál sarkalatos pont a megfelelő küszöbérték megválasztása, hogy azt a hálót állíthassuk elő, amely a legjobban tükrözi az eredeti jelenet részleteit. Erre a rendszer jelenleg nem tartalmaz automatikus megoldást, a felhasználónak kell eldöntenie a modell aktuális kinézetétől függően, hogy milyen paramétert szeretne használni. A környezetet legjobban visszaadó textúra utolsó lépésként kerül fel a testhálóra, teljesen automatikusan, az első felhasznált képkocka alapján. A jövőbeni fejlesztésekben érdemes lehet megvizsgálni további

módszereket, hogy a testháló minden eleme a legideálisabb kamerakép alapján kapja meg a hozzá tartozó textúrát. A megjelenítési modul a MATLAB fejlesztőkörnyezet beépített funkciót használja. Egy ezt a célt szolgáló, önálló alkalmazás a későbbiekben tovább javíthatja a program felhasználója számára a háromdimenziós élményt.

11. Összefoglalás

Az általunk elkészített rendszer mobilizálható részére igyekeztünk megtalálni az irányítója számára legkényelmesebb megoldást. Az alkalmazott joystick-os vezérlés véleményünk szerint pontosabban és egyszerűbben teszi ezt lehetővé, mint a hagyományos technikák (billentyűzet és egér). A robotra csatlakoztatott telefonkészülék által küldött másodpercenkénti több képkocka, a kábelmentes kapcsolatnak köszönhetően érkezik meg a felhasználóhoz, ezáltal folyamatos képet adva a robot aktuális környezetéről, elősegítve a zökkenőmentes irányítást.

A mobil egységtől kapott képek alapján készül el a rendszer fő célja, a környezet háromdimenziós modellje is. A képeket készítő kamera előzetesen kiszámított belső paramétereivel, vagy ennek hiányában önkalibrációval pontosított modell a valós jelenet olyan leképezése, amely annak arányait és részleteit élethűen tükrözi. A jobb vizualizáció érdekében a jelenet virtuális mására felkerül az eredeti képkockák alapján elkészített textúra is. Az eredmények bármikor megjeleníthetők térbeli ábraként, valamint későbbi felhasználás céljából elmenthetők a számítógépre.

Irodalomjegyzék

1. **He, Ray C.** *Stereo Vision and Mapping with Unsynchronized Cameras*. Massachusetts : MIT, 2008.
2. **Rusinkiewicz, Szymon, Hall-Holt, Olaf and Levoy, Marc.** *Real-Time 3D Model Acquisition*. 2002.
3. **4D View Solutions.** 4D View Solutions: real-time 3d video capture systems. [Online] 4D View Solutions. [Cited: 2011. évi 6-11.] <http://www.4dviews.com/>.
4. **Fitzgibbon, Andrew and Zisserman, Andrew.** *Automatic 3D model acquisition and generation of new images from video sequences*. Dept. of Engineering Science, University of Oxford : s.n.
5. *CCD vs. CMOS: Facts and Fiction*. **Litwiller, Dave.** hely nélkül. : Laurin Publishing Co. Inc., 2001. Jan, PHOTONICS SPECTRA.
6. Technology of Robotics. *History of Microcontroller ATME1 AVR*. [Online] Technology of Robotics. [Hivatkozva: 2011. november 2.] <http://robotechno.us/history-microcontroller-atmel-avr.html>.
7. Wikipedia. *PIC microcontroller*. [Online] 2011. November 4. [Hivatkozva: 2011. November 8.] http://en.wikipedia.org/wiki/PIC_microcontroller.
8. **Czarkowski, Dariusz.** DC-DC Converters. [book auth.] MUHAMMAD H. RASHID. *Power Electronics Handbook*. Canada : Academic Press, 2001.
9. **Axelson, Jan.** *Serial Port Complete*. Medison : Lakeview Research LLC, 2007. 978-1931448-07-9.
10. **Gibilisco, Stan.** *Teach Yourself Electricity and Electronics*. s.l. : McGraw-Hill/TAB Electronics, 2001. ISBN 9780071377300.
11. **Control, Honeywell - MICRO SWITCH Sensing and.** Proximity Sensors.
12. **Rosten, Edward, Porter, Reid and Drummond, Tom.** *Faster and better: a machine learning approach to corner detection*. IEEE International Conference on Computer Vision : s.n., 2005.
13. **Bay, Herbert, Tuytelaars, Tinne and Gool, Luc Van.** *SURF: Speeded Up Robust Features*. 2006.
14. **Ma, Yi, et al.** *An Invitation to 3-D Vision*. 2001.
15. **Hartley, Richard and Zisserman, Andrew.** *Multiple View Geometry in Computer Vision*. Cambridge : Cambridge University Press, 2003. ISBN 0521 54051 8.
16. **Du, Dingshu and Hwang, Frank.** *Computing in Euclidean geometry*. s.l. : World Scientific, 1995. ISBN 9810218761.
17. **Edelsbrunner, Herbert and Mücke, Ernst P.** *Three-Dimensional Alpha Shapes*. s.l. : ACM Transactions on Graphics, 1994.
18. **Corcoran, Andrew.** *3D Object Extraction from Multiple Images*. University of Dublin : s.n., 2008.
19. **Trucco, Emanuele és Verri, Alessandro.** *Introductory Techniques for 3-D Computer Vision*. hely nélkül. : Prentice Hall, 1998. 0132611082.

20. **Rosten, Ed és Drummond, Tom.** *Machine Learning for High Speed Corner Detection*. 2006.
21. Geomview. *Geomview*. [Online] 2007. 8 1. [Hivatkozva: 2010. 11 5.] <http://www.geomview.org/>.
22. Edge Detection. *Wikipedia*. [Online] 2010. 11 1-11. [Cited: 2010. 11 03-11.] http://en.wikipedia.org/wiki/Edge_detection.
23. Corner detection. *Wikipedia*. [Online] 2010. november 3. [Hivatkozva: 2010. november 4.] http://en.wikipedia.org/wiki/Corner_detection.
24. Computational Geometry Algorithms Library. *CGAL*. [Online] 2010. 10 15. [Hivatkozva: 2010. 11 5.] <http://www.cgal.org/>.
25. Bundle adjustment. *Wikipedia*. [Online] 2010. 11 03. [Hivatkozva: 2010. 11 05.] http://en.wikipedia.org/wiki/Bundle_adjustment.
26. **Reid, Fiach.** *Network Programming in .Net with C# and Visual Basic .Net*. United States of America : Elsevier Digital Press, 2004.
27. **Jones, Anthony, Ohlund, Jim and Olson, Lance.** *Network Programming for the Microsoft .NET Framework*. Redmond : Microsoft Press, 2004. 073561959x.
28. **Krowczyk, Andrew, et al.** *Professional .NET Network Programming*. New York : Apress, Inc., 2002. 1861007353.
29. **Tanenbaum, Andrew S.** *Computer Networks*. 2004. 978-0-13-212695-3.
30. **Mark S. Nixon, Alberto S. Aguado.** *Feature Extraction and Image Processing*. Great Britain : s.n., 2002. ISBN 0 7506 5078 8.
31. **Pan, Qi, Reitmayr, Gerhard and Drummond, Tom.** *ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition*. Machine Intelligence Laboratories, Department of Engineering, Cambridge University. Cambridge, United Kingdom : s.n., 2009.
32. *CMOS vs CCD: Maturing Technologies, Maturing Markets*. **Litwiller, Dave.** hely nélkül : Laurin Publishing Co. Inc., 2005., PHOTONICS SPECTRA .

Ábrajegyzék

3-1. ábra: Quadrotor [http://groups.csail.mit.edu/rrg/].....	6
3-2. ábra: Kinect Szenzor.....	6
3-3. ábra: Clearpath Husky A200 [http://www.clearpathrobotics.com/husky]	6
3-4. ábra: A ProFORMA rendszer modellezési lépései.....	7
4-1. ábra: Wireless Camera	9
4-2. Ábra: WiFi Camera	9
4-3. Ábra: Webkamera	9
4-4. ábra: Bluetooth osztályok [Wikipedia].....	11
4-5. ábra: A projecthez használt Sparkfun BlueSMiRF modul.....	11
4-6. ábra: AVR.....	12
4-7. ábra: PIC	12
4-8. ábra: Joystick, Gamepad	13
5-1. ábra: Előzetes rendszerterv.....	16
5-2. ábra: Az újragondolt rendszer terve	17
6-1. ábra: Robot.....	18
6-2. ábra: Vezérlő kapcsolási rajza	19
6-3. ábra Vezérlő elektronika	20
6-4. ábra PWM jelek.....	21
6-5. ábra szervó motor vezérlőjel	21
6-6. ábra Sharp GP2D12	22
6-7. ábra Maxbotix LV-EZ0	22
6-8. ábra Működési ciklus.....	22
7-1. ábra: Bresenham-kör [11 p. 5]	29
7-2. ábra: FAST módszer eredményei	31
7-3. ábra: SURF módszer eredményei	31
7-4. ábra: A kamera síkjai [15 p. 178].....	33
7-5. ábra: Hasonlósági és projektív rekonstrukció [15 p. 265].....	36
7-6. ábra: Az epipoláris geometria [14 p. 81].....	40
7-7. ábra: Transzformáció síkon keresztül [15 p. 243]	40
7-8. ábra: A négy lehetséges kameramátrix [15 p. 260]	47
7-9. ábra: A reprojekciós hiba [15 p. 311].....	47
7-10. ábra: A kalibrációhoz használt képek.....	49
7-11. ábra: Delaunay-háromszögek [15].....	50
7-12. ábra: Az α paraméter változása [17 p. 45]	51
9-1. ábra: Pontkövetési példák.....	58
9-2. ábra: Kültéri tesztjelenet.....	59
9-3. ábra: Kalibrált módszer eredményei (felülnézet)	59
9-4. ábra: Önkalibrációs módszer eredményei (felülnézet)	60
9-5. ábra: Kültéri tesztjelenet.....	60
9-6. ábra: Az előre kalibrált módszer eredményei (felülnézet).....	61
9-7. ábra: Konvex Delaunay-testháló	62

9-8. ábra: Finomított Alfa-háló (felülnézet)	62
9-9. ábra: Elkészült alfa-háló	63
9-10. ábra: Elkészült alfa-háló (felülnézet).....	63
9-11. ábra: Az első tesztjelenet textúrázva	64
9-12. ábra: A második tesztjelenet textúrázva.....	64