

FELADATKIÍRÁS HELYE

A feladatkiírást a tanszéki adminisztrációban lehet átvenni, és a leadott munkába eredeti, tanszéki pecséttel ellátott és a tanszékvezető által aláírt lapot kell belefűzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell beleszerkeszteni ezt a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Irányítástechnika és Informatika Tanszék

Tárgymanipuláció kulcspontok detektálásán alapuló képfeldolgozással

DIPLOMATERV

Készítette
Ayhan Dániel

Konzulens
dr. Kiss Bálint

2017. május 20.

Tartalomjegyzék

Kivonat	4
Abstract	5
1. Bevezetés, feladatkitűzés	6
2. Kameramodell, térérzékelés kamerával	8
2.1. Kameramodell, kamerakalibráció	8
2.2. Külső paraméterek meghatározása, PnP probléma	10
2.3. RANSAC algoritmus	10
3. Koordináta-rendszerek, Kéz-szem kalibráció	13
3.1. Kabsch algoritmus	14
3.2. A rotációk becslése	15
3.3. A transzlációk becslése	17
4. Kulcspontkeresés és -párosítás	19
4.1. Kulcspontkereső algoritmusok	19
4.1.1. SIFT kulcspontkeresés	19
4.1.2. SURF kulcspont-keresés	21
4.2. Leíró-párosító algoritmusok, párosítás-szűrések	21
4.2.1. Arányteszt alapú párosítás-szűrés	22
4.2.2. Kereszttellenőrzéses szűrés	22
4.2.3. Epipoláris egyenes használata	22
4.2.4. Homográfia-alapú szűrés	23

5. 3D rekonstrukció	24
5.1. Többnézetes háromszögelés	24
5.2. Leíró-párosító algoritmusok sajátosságaiból adódó problémák	26
5.2.1. A ritkaság korrigálása	26
5.2.2. Téves párosítások szűrése	26
6. Megvalósítás	27
6.1. A szoftver feladatai	27
6.2. Kiválasztott implementáció	27
6.2.1. Kulcspontkeresés és -párosítás implementációja	27
6.2.2. A kéz-szem kalibráció implementációja	28
6.2.3. 3D rekonstrukció és a kéz-szem kalibráció implementációja	28
6.2.4. A pozíció és orientáció meghatározásának implementációja	29
6.3. A robotirányító rendszer felépítése	30
6.4. A PLC és robotprogram	32
6.5. A PC program	32
7. Eredmények	34
8. Értékelés, továbbfejlesztési lehetőségek	38
Melléklet	39
Irodalomjegyzék	41

HALLGATÓI NYILATKOZAT

Alulírott *Ayhan Dániel*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet ([nem kívánt törlendő](#)) meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a forrás megadásával megjelöltettem.

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

Budapest, 2017. május 20.

Ayhan Dániel
hallgató

Kivonat

Az előre nem definiált környezetben, például háztartásokban működő robotok egyik nem megoldott feladata tárgyak megtalálása ebben a tetszőleges környezetben. Jelen diplomatervbén egy robotikai alkalmazás működését mutatom be, amely ezt a feladatot hívatott megoldani. A szoftver feladata egy robotkar mozgatása, és az arra szerelt kamera képe alapján egy előzetesen betanított tárgy ismeretlen térbeli helyzetének meghatározása. A poz becslése ismert térbeli pozíójú pontok, és a nekik megfelelő képi pontok alapján PnP algoritmussal történik. A képi pontok megtalálását SURF kulcspontereső algoritmussal végzi a szoftver. A betanítás során az ismert térbeli pozíció meghatározását, a 3D rekonstrukciót több, ismert nézőpontból elkészített kép alapján többnézetes háromszögeléssel végzi a szoftver. A dolgozatban kitérek a robotirányító rendszer szoftveres és hardveres felépítésére, valamint a kéz-szem kalibráció implementálására is, végül pedig a megvalósítás eredményeinek elemzésére.

Abstract

Finding objects in an arbitrary environment is one of the unsolved problems about robots operating in such environments, eg. households. In this thesis I present a robotics application, which is supposed to solve this problem. The software is controlling a robotic arm, and estimates the spatial position and orientation of an object for which it has been trained previously. The estimation is done using images retrieved from a camera mounted on the end effector of the robot. The software uses PnP algorithm which estimates the spatial pose from object points with known 3D coordinates and the corresponding image points. The image points are found with SURF keypoint detector. During training the algorithm, 3D reconstruction is done via multi-view triangulation using multiple images taken from known positions. Furthermore I describe the hardware and software architecture of the robot controlling system, and the implementation of the hand-eye calibration. Finally I analyze the results of the implementation.

1. fejezet

Bevezetés, feladatkitűzés

A ma használatos robotok legtöbb esetben előre meghatározott környezetben, és körülmenyek között működnek. Az ipari robotkarok például jellemzően ugyanazt az alkatrészt szerelik folyamatosan ugyanabban a gyártócellában, a gyárakban működő mobilis robotok pedig jellemzően egy, a padlóra ragasztott csíkot követve előre meghatározott útvonalakon közlekednek. A robotikában jelenleg ezeket az alkalmazásokat részesítik előnyben, ugyanis reprodukálható körülmények között érhető el a leginkább robusztus működés.

Az előre szigorúan nem definiált környezetben való működés kevés alkalmazásra jellemző, bár azért ilyen is előfordul, például az otthoni takarítórobotok (amelyek robusztussága szintén hagy némi kívánnivalót maga után). Az ilyen környezetben való működés egyik feladata lehet tetszőleges környezetben elhelyezkedő ismert tárgyak megtalálása és megfogása robotkar és megfogó segítségével. Ezt a feladatot kell például lakásokban működő robotknak megoldaniuk, ha egy adott tárgyat a lakásban meg kell keresniük. Ez idős vagy mozgássérült személyeket segítő robotok esetében bírhat különös jelentőséggel. A tárgy pozíciójának alacsony pontosságú becslése történhet RFID technológiával, pontos meghatározása pedig képfeldolgozással.

Diplomatervem témája egy olyan alkalmazás fejlesztése, amely képes tetszőleges környezetben egy előre betanított tárgy térbeli pozícióját és orientációját meghatározni képfeldolgozás segítségével. Feltesszük, hogy a tárgy pozíciója olyan pontossággal ismert, hogy képesek vagyunk róla kamerával képet készíteni, tehát otthonban működő robot esetében nem a másik szobában, nagyon messze, vagy a szekrény mögött van. Az algoritmusokat egy Mitsubishi RV-2F-Q robotkarral kellett megvalósítanom.

Az általam választott módszer kulcsPontok keresésén alapul. Ezeknek a jellemző pontoknak a megkeresése után meghatározható a tárgy térbeli orientációja. Ehhez azonban szükség van a jellemző pontokhoz tartozó, a tárgyhoz kötött koordináta-rendszerben értelmezett térbeli koordinátákra, vagyis a 3D rekonstrukcióra.

A megvalósítás során OpenCV-t használtam, ami egy C++-ban fejlesztett képfeldolgozási függvénykönyvtár.

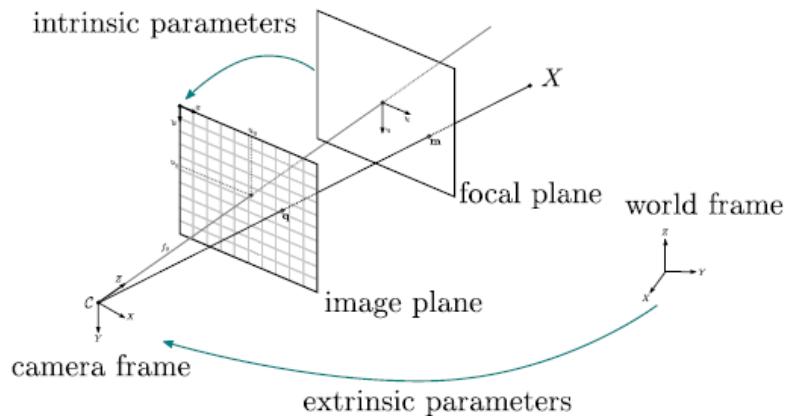
A dolgozat felépítése a következő. A 2. fejezetben részleteiben leírom a megvalósítandó feladatot. A 3. fejezetben a 3D látásnál használt alapvető modellt és algoritmusokat mutatom be. A 4. fejezet a különböző térbeli koordináta-transzformációk meghatározására szolgáló kéz-szem kalibrációról szól. Az 5. fejezetben a kulcspontok keresését tárgyalom, valamint, hogy a két képen megtalált kulcspontokat mi alapján lehet összepárosítani. A 6. fejezetben a 3D rekonstrukciót mutatom be. A 7. fejezet a megvalósítás részleteiről szól, valamint a szoftvernek azon részeiről, amelyek nem szigorúan a gépi látáshoz kapcsolódnak, de a működéshez elengedhetetlenek. A 8. fejezet az eredményeket tárgyalja be, a 9.-ben pedig értékelem a munkát és vázolom a továbbfejlesztési lehetőségeket.

2. fejezet

Kameramodell, térérzékelés kamerával

2.1. Kameramodell, kamerakalibráció

A gépi látásban a kamerát legtöbbször az úgynevezett pinhole-kameramodellel írjuk le. Ez a modell azt feltételezi, hogy a kép keletkezése olyan módon történik, hogy a háromdimenziós teret perspektivikusan vetítjük a képsíkra.



2.1. ábra. A pinhole-kameramodell.

A kamerához definiálható egy kamera koordináta-rendszer. Ennek a koordináta-rendszernek az origójában van a vetítés középpontja, a z tengelye merőleges a vetítés síkjára. A kép koordináta-rendszer síkbeli koordináta-rendszer, a síkra vetített pontokat értelmezzük ebben. x és y tengelyei a kamera koordináta-rendszer tengelyeivel egyirányúak, és általában u és v betűkkel jelöljük őket, megkülönböztetve őket a kamera koordináta rendszer tengelyeitől. Amennyiben homogén koordinátákkal jellemezzük a tér pontjait, mind a perspektív vetítés, mind a világ koordináta-rendszer és a kamera koordináta-rendszer közötti eltolás, forgatás mátrixszal kifejezhető:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{T} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.1)$$

$$\mathbf{A} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

Ahol:

- u, v a képpont képi koordinátái. w a homogén koordináták miatti komponens
- x, y, z a tárgypont koordinátái világ koordináta-rendszerben
- \mathbf{A} az úgynevezett kameramátrix
- f_x, f_y a kamera fókusztávolsága (ideális esetben egyenlők, valós alkalmazásokban nem feltétlenül)
- γ egy *skew* nevű torzítási paraméter, ami ideális esetben 0
- u_0, v_0 a kamera principális pontjának koordinátái kép koordináta-rendszerben. Ez az a pont, ahol a kamera koordináta rendszer z tengelye döfi a képsíkot
- \mathbf{T} a világ koordináta-rendszerből kamera koordináta-rendszerbe való transzformáció.
- $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ forgatási mátrix
- $\mathbf{t} \in \mathbb{R}^{3 \times 1}$ eltolási vektor

A kameramátrix paramétereit kamera belső paramétereinek nevezzük. Mivel ezek a kamera rendeltetésszerű használata során nem változnak jelentősen, a kamera szerves részei, egyszeri kalibrálással meghatározhatók. A koordináta-rendszerek közti transzformáció paraméterei a külső paraméterek. Az összes paraméter meghatározható kellően nagy számú pont-pár segítségével, ezt kamera-kalibrációnak nevezzük. Pontpárok alatt a világ koordináta-rendszerben ismert pozíciójú pontokat és azok kép koordináta-rendszerben megadott vetületét értjük. A tényleges kalibrációnál általában egy kalibráló objektumról több, különböző nézőpontból készült képet használunk. A kalibráló objektum jellemzően több, egyszerűen, de akár szubpixels pontossággal detektálható pontból áll, például sakktábla-mintázat sarkai, vagy körökből álló rácsozat pontjainak közepe.

A valóságban a pinhole kameramodell sajnos nem állja meg a helyét, a képet többféle torzítás terheli például abból adódóan, hogy nem vetítést alkalmazunk, hanem lencserendszert, valamint, hogy a detektoron elhelyezkedő pixelek gyártástechnológiája nem tökéletes. Ezeket a paramétereket most nem mutatom be.

Az OpenCV-ben a kamera-kalibráció implementálva van, ami meghatározza mind a belső paramétereket, mind a torzítási paramétereket.

2.2. Külső paraméterek meghatározása, PnP probléma

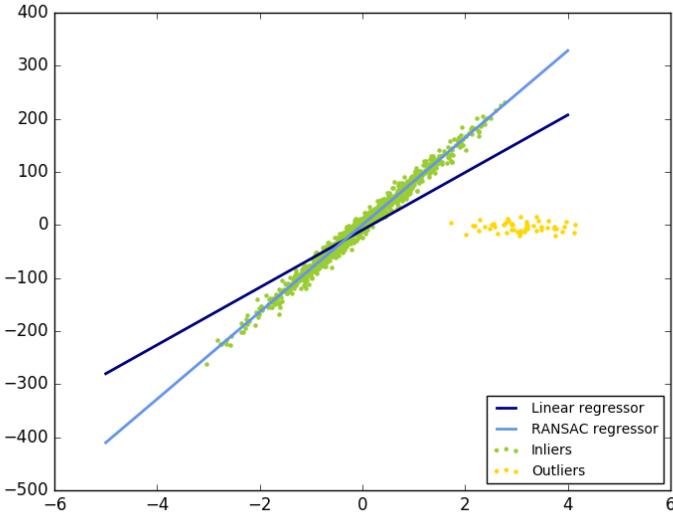
A kamera használata során szükség van a külső paramétereinek ismeretére. Azt a feladatot, amely ismert térbeli helyzetű tárgypontok és a hozzájuk tartozó képi pontok alapján próbálja meghatározni a kamera külső paramétereit PnP, azaz Perspective-n-Point problémának nevezzük.

Ennek egyik verziója a P3P algoritmus [4], amely csak 4 pontpárkból számítja a paramétereket. Három pont kell ahhoz, hogy véges sok megoldást kapunk (négy ponttal már túlhatározott lenne a feladat). A negyedik pontot az algoritmus arra használja fel, hogy a véges sok megoldás közül kiválassza a helyeset. Ezt úgy teszi meg, hogy a lehetséges négyféle megoldás közül azt választja, amelyikkel a negyedik pontot a képre vetítve a legkisebb hibát kapjuk a megadott képi ponthoz képest. Ezt aztán tetszőleges optimumkereséssel tovább lehet finomítani, pl. Levenberg-Marquardt eljárással. Ezt az eljárást használhatjuk akkor is, ha több, mint négy pont áll rendelkezése.

A PnP probléma megoldása során előfordulhat, hogy hibás pontpárosítások is kerülnek a bemeneti adatok közé. Többek között ennek a kiszűrésére alkalmas a következő szakaszban bemutatott algoritmus.

2.3. RANSAC algoritmus

A RANSAC (Random Sample Consensus) [3] algoritmus nem kifejezetten képfeldolgozási, hanem általános célú algoritmus, de az általam használt algoritmusok közül többnek részét képezi, ezért indokoltnak tartom bemutatni a működését. A RANSAC módszer annak a problémának egy megközelítése, amikor hibás mérésekkel terhelt adathalmazra próbálunk modellt illeszteni. Itt a modell lehet bármi, ami valamilyen pontossággal illeszkedik az adatokra. Ezt általában 2D ponthalmazra való egyenesillesztéssel szokás szemléltetni. A kiugró mérési hibák miatt jellemzően nem érdemes a legkisebb négyzetek módszerét használni, mert rossz illeszkedést fog mutatni. Ezt szemlélteti a következő ábra.



2.2. ábra. A RANSAC algoritmus és a legkisebb négyzetek módszerének összehasonlítása.

Az algoritmus leírása során az angol terminológiát használom, a helyes mérési adatot *inlier*-nek nevezem, a mérési hibát pedig *outlier*-nek.

A RANSAC algoritmus működése a következő:

RANSAC algoritmus

- 1: **eljárás** RANSAC(*adatok*, K , e)
 - 2: *legjobb modell* \leftarrow null
 - 3: **ciklus** $i \leftarrow 1, K$
 - 4: *vélt inlierek* $\leftarrow N$ darab véletlen adatpont
 - 5: *vélt modell* \leftarrow *vélt inlierek*-re illesztett modell
 - 6: *illeszkedők* \leftarrow *vélt modell*-re e hibahatáron belül illeszkedő pontok
 - 7: **ha** *illeszkedők* száma $> N$
 - 8: *becsült modell* \leftarrow *illeszkedők*-re illesztett új modell
 - 9: **ha** *legjobb modell*-nél jobb *becsült modell*
 - 10: *legjobb modell* \leftarrow *becsült modell*
 - 11: **visszatérés** *legjobb modell-lel*
-

Az algoritmus robusztussága abban rejlik, hogy ha a 4. sorban a *vélt inlierek* közé nem kerül egyetlen outlier sem, a *vélt modell*-re a legtöbb inlier illeszkedni fog e hibahatáron belül, és a 8. sorban egy nagyon jó illeszkedésű modellt kapunk. Ebben az esetben akár le is lehet állítani az iterálást, és rögtön visszatérni a legjobb modellel.

A RANSAC algoritmus előnye, hogy nagy mennyiségű outlier jelenlétében is képes megterállni az inlierekre illeszkedő modellt. Létezik olyan verziója is, amellyel 5% inlier arány

mellett is sikeres eredményeket értek el [5]. Hátránya, hogy a véletlenszerű működés miatt nem garantál "optimális" megoldást, és ha meg is találja, a végrehajtási idő véletlenszerűen változhat futásról futásra.

3. fejezet

Koordináta-rendszerek, Kéz-szem kalibráció

A robotikában különböző koordináta-rendszereket használunk. Jelen feladat során a következő koordináta-rendszereket használom:

- A robot bázis koordináta-rendszere, avagy világ koordináta-rendszer. A robot nulla-dik, álló szegmenséhez van rögzítve. Betűjele: R (Robot)
- Megfogó vagy szerszám koordináta-rendszer, amely a robot utolsó szegmensének végén található. Betűjele: T (Tool)
- Kamera koordináta-rendszer. Ennek a koordináta-rendszernek az origóját a pinhole modell alapján a vetítési pontba képzeljük el, a z tengelye egybeesik az optikai tengellyel és a kamera nézési irányába mutat. A valóságban az optikai tengely és a lencse síkjának metszéspontjában található az origó. Betűjele: C (Camera)
- Tárgy koordináta-rendszer. Ez az a koordináta-rendszer, amiben a megfigyelt tárgy pontjainak koordinátáit értelmezzük. Betűjele: O (Object)

A B koordináta-rendszerből az A koordináta-rendszerbe való transzformáció a következőképpen írható fel homogén koordinátákban:

$$\mathbf{T}_{AB} = \left[\begin{array}{c|c} \mathbf{R}_{AB} & \mathbf{t}_{AB} \\ \hline \mathbf{0}^T & 1 \end{array} \right] \in \mathbb{R}^{4 \times 4} \quad (3.1)$$

Itt \mathbf{R}_{AB} 3x3-as rotációs mátrix, \mathbf{t}_{AB} pedig oszlopvektor. A \mathbf{t}_{AB} vektor az A koordináta-rendszer origójából a B koordináta-rendszer origójába mutat (A -ban felírva), míg \mathbf{R}_{AB} oszlopai a B koordináta-rendszer egységevektorai szintén A -ban felírva.

A jelölésteknél következik, hogy $\mathbf{T}_{AB}^{-1} = \mathbf{T}_{BA}$

A kéz-szem kalibráció során a robot megfogójának és a kamerának a relatív helyzetét kívánjuk meghatározni, azaz \mathbf{T}_{TC} -t (vagy inverzét). Amennyiben ez ismert, a kamera helyzetét képesek vagyunk világ koordináta-rendszerben is megadni, ugyanis a megfogó és a világ koordináta-rendszer közti transzformációt a direkt kinematikai probléma megoldásával meghatározhatjuk.

A kalibrációhoz használt módszer a belső paraméterek kalibrálásánál alkalmazott hasonló, egy kalibrációs objektumról képeket készítünk, amelyen megkeressük a tárgy ismert pozíciójú pontjait. Az így kapott összefüggésekkel számítjuk először a kamera külső paramétereit, majd az egyéb ismert paraméterek felhasználásával a kamera relatív helyzetét.

A fent ismertetett koordináta-rendszerek közötti transzformációs lánc felírható:

$$\mathbf{T}_{CO}\mathbf{T}_{OR}\mathbf{T}_{RT}\mathbf{T}_{TC} = \mathbf{I} \quad (3.2)$$

Az egyes transzformációknak speciális tulajdonságaik vannak, amit kalibráláskor ki lehet (sőt kell) használni. A kamera a végberendezésre szilárdan van rögzítve, nem mozdul el. A robot is rögzítve van a gépalaphoz, amihez képest a kalibráló tárgy nem mozog a kalibráció során. Ez azt jelenti, hogy a \mathbf{T}_{OR} és \mathbf{T}_{TC} mátrixok nem változnak, ha a robotkart mozgatjuk. Ezen felül ismert a robotkar pozíciója a bázis koordináta-rendszerben, hiszen megkaphatjuk a direkt geometriai feladat megoldásával. A PnP algoritmus továbbá megadja a kamera helyzetét a tárgy koordináta-rendszerben. Így \mathbf{T}_{CO} és \mathbf{T}_{RT} mátrixok ismertek minden kép esetében.

A kalibráció két lépésben történik. Az első lépésben a rotációs mátrixokat számítjuk ki, a másodikban a transzlációs vektorokat.

Először azonban a kalibrációhoz szükséges Kabsch algoritmus működését mutatom be.

3.1. Kabsch algoritmus

A Kabsch algoritmus [6] két vektorhalmaz közti optimális forgatási mátrixot számítja ki. A probléma felvetése a következő. Adottak $\mathbf{p}_i^T \in \mathbb{R}^{1 \times 3}$ és $\mathbf{q}_i^T \in \mathbb{R}^{1 \times 3}$ vektorok, $i = 1..N$. Keressük azt az $\hat{\mathbf{R}} \in \mathbb{R}^{3 \times 3}$ mátrixot, amelyre:

$$\hat{\mathbf{R}} = \arg \min_{\mathbf{R}} \sum_{i=1}^N \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i\|^2 \quad (3.3)$$

Az algoritmus a vektorhalmazok közti koordinátánkénti kovariancia-mátrixot számítja ki, majd ennek SVD felbontásából határozza meg az optimális forgatást:

$$\mathbf{P} := \begin{bmatrix} \vdots \\ \mathbf{t}_{RTi}^T \\ \vdots \end{bmatrix} \in \mathbb{R}^{N \times 3}, \quad \mathbf{Q} := \begin{bmatrix} \vdots \\ \mathbf{t}_{OQi}^T \\ \vdots \end{bmatrix} \in \mathbb{R}^{N \times 3}, \quad i = 1..N \quad (3.4)$$

$$\mathbf{A} = \mathbf{P}^T \mathbf{Q} \quad (3.5)$$

$$\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \mathbf{A} \quad (3.6)$$

$$\hat{\mathbf{R}} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U} \mathbf{V}^T) \end{bmatrix} \mathbf{V}^T \quad (3.7)$$

Érdemes megjegyezni, hogy a vektorhalmazok tömegközéppontja az origóban kell hogy legyen, amennyiben ez nem teljesül, könnyen korrigálható a vektorok eltolásával.

3.2. A rotációk becslése

A (3.2) egyenletet átalakítva kapjuk, hogy:

$$\mathbf{T}_{OC} = \mathbf{T}_{OR} \mathbf{T}_{RT} \mathbf{T}_{TC} \quad (3.8)$$

Ha az egész egyenletet beszorozzuk jobbról a nullvektorral:

$$\mathbf{T}_{OC} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{T}_{OR} \mathbf{T}_{RT} \mathbf{T}_{TC} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.9)$$

És elvégezzük a megfelelő szorzásokat, kapjuk:

$$\begin{bmatrix} \mathbf{t}_{OC} \\ 1 \end{bmatrix} = \mathbf{T}_{OR} \begin{bmatrix} \mathbf{R}_{RT} \mathbf{t}_{TC} + \mathbf{t}_{RT} \\ 1 \end{bmatrix} \quad (3.10)$$

Ebben az egyenletben számunkra ismeretlen a \mathbf{T}_{OR} mátrix és a \mathbf{t}_{TC} vektor. A kalibráció során készítünk N darab képet, amelyekhez meghatározzuk a külső paramétereket, vagyis a \mathbf{T}_{OC} mátrixot. Ezeket a képeket úgy készítjük el, hogy az \mathbf{R}_{RT} mátrix állandó, vagyis két kép készítése között nem forgatjuk a kamerát, csak eltoljuk. Ekkor kiegészíthető a fenti egyenlet egy, a kép sorszámát jelölő indexsel:

$$\begin{bmatrix} \mathbf{v}_{OCi} \\ 1 \end{bmatrix} = \mathbf{T}_{OR} \begin{bmatrix} \mathbf{R}_{RT} \mathbf{t}_{TC} + \mathbf{t}_{RTi} \\ 1 \end{bmatrix} \quad i = 1..N \quad (3.11)$$

Azok az ismeretlenek, illetve paraméterek, amik nem kaptak indexet, azok állandóak a képek készítése során. Vegyük a fenti N darab egyenlet átlagát:

$$\begin{bmatrix} \frac{\sum \mathbf{t}_{OCi}}{N} \\ 1 \end{bmatrix} = \mathbf{T}_{OR} \begin{bmatrix} \mathbf{R}_{RT} \mathbf{t}_{TC} + \frac{\sum \mathbf{t}_{RTi}}{N} \\ 1 \end{bmatrix} \quad (3.12)$$

Ha kivonjuk a (3.12) egyenletet a (3.11) egyenletből, a következőt kapjuk:

$$\begin{bmatrix} \mathbf{t}_{OCi} - \frac{\sum \mathbf{t}_{OCi}}{N} \\ 0 \end{bmatrix} = \mathbf{T}_{OR} \begin{bmatrix} \mathbf{t}_{RTi} - \frac{\sum \mathbf{t}_{RTi}}{N} \\ 0 \end{bmatrix} \quad (3.13)$$

Bevezetjük, hogy:

$$\begin{aligned} \overline{\mathbf{t}_{OCi}} &:= \mathbf{t}_{OCi} - \frac{\sum \mathbf{t}_{OCi}}{N} \\ \overline{\mathbf{t}_{RTi}} &:= \mathbf{t}_{RTi} - \frac{\sum \mathbf{t}_{RTi}}{N} \end{aligned}$$

Szemléletesen a felülvont változó az adott vektornak az összes vektor tömegközéppontjához képesti relatív helyzetét mutatja meg. Vegyük észre, hogy ezek kiszámíthatók, ismerjük őket. (3.13) írható a következő alakban is:

$$\begin{bmatrix} \overline{\mathbf{t}_{OCi}} \\ 0 \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{R}_{OR} & \mathbf{t}_{OR} \\ \mathbf{0}^T & 1 \end{array} \right] \begin{bmatrix} \overline{\mathbf{t}_{RTi}} \\ 0 \end{bmatrix} \quad (3.14)$$

Ebből pedig:

$$\overline{\mathbf{t}_{OCi}} = \mathbf{R}_{OR} \overline{\mathbf{t}_{RTi}} \quad (3.15)$$

\mathbf{R}_{OR} kiszámítása tehát ekvivalens az kapott vektorpárok közti optimális rotáció kiszámításával, amit Kabsch algoritmus számít ki.

Ez után \mathbf{R}_{TC} számítása történik. (3.2) alapján:

$$\mathbf{R}_{TRi} \mathbf{R}_{RO} \mathbf{R}_{OCi} = \mathbf{R}_{TCi} \quad i = 1..N \quad (3.16)$$

A jobb oldalon található rotációs mátrixok "átlagát" kellene venni, ezt is a Kabsch algoritmussal lehet kiszámítani. Az algoritmusnak azonban vektorpárokra van szüksége, nem pedig mátrixokra. Ebben az esetben $3N$ darab vektorpár alapján végezzük a számítást, minden átlagolandó rotációhoz 3-3 vektorpár tartozik: egy ilyen párhármas egyik tagja a rotációs mátrix megfelelő oszlopvektora, a másik tagja pedig a neki megfelelő egységvektor ($[1\ 0\ 0]^T$, $[0\ 1\ 0]^T$, vagy $[0\ 0\ 1]^T$). Ekkor a (3.4)-ben definiált mátrixok a következő alakot öltik:

$$\mathbf{P} := \begin{bmatrix} \vdots \\ \mathbf{R}_{TCi} \\ \vdots \end{bmatrix}, \quad \mathbf{Q} := \begin{bmatrix} \vdots \\ \mathbf{I} \\ \vdots \end{bmatrix}, \quad i = 1..N \quad (3.17)$$

ahol \mathbf{I} a 3×3 -as egységmátrix.

Így a rotációk átlagát véve a Kabsch algoritmussal megkaphatjuk az \mathbf{R}_{TC} mátrixot.

3.3. A transzlációk becslése

Ha kifejtjük a (3.8) egyenletet (3.1) szerint, kapjuk, hogy:

$$\left[\begin{array}{c|c} \mathbf{R}_{OC} & \mathbf{t}_{OC} \\ \hline \mathbf{0}^T & 1 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{R}_{OR} & \mathbf{t}_{OR} \\ \hline \mathbf{0}^T & 1 \end{array} \right] \left[\begin{array}{c|c} \mathbf{R}_{RT} & \mathbf{t}_{RT} \\ \hline \mathbf{0}^T & 1 \end{array} \right] \left[\begin{array}{c|c} \mathbf{R}_{TC} & \mathbf{t}_{TC} \\ \hline \mathbf{0}^T & 1 \end{array} \right] \quad (3.18)$$

Összeszorozva a jobb oldalt a következő adódik:

$$\left[\begin{array}{c|c} \mathbf{R}_{OC} & \mathbf{t}_{OC} \\ \hline \mathbf{0}^T & 1 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{R}_{OR}\mathbf{R}_{RT}\mathbf{R}_{TC} & \mathbf{R}_{OR}\mathbf{R}_{RT}\mathbf{t}_{TC} + \mathbf{R}_{OR}\mathbf{t}_{RT} + \mathbf{t}_{OR} \\ \hline \mathbf{0}^T & 1 \end{array} \right] \quad (3.19)$$

Az ebben a fázisban elkészített képek számát jelöljük M -mel. Ebből, ha bevezetjük az indexet a változó paraméterekhez , kapjuk, hogy:

$$\mathbf{t}_{OCi} = \mathbf{R}_{OR}\mathbf{R}_{RTi}\mathbf{t}_{TC} + \mathbf{R}_{OR}\mathbf{t}_{RTi} + \mathbf{t}_{OR} \quad i = 1..M \quad (3.20)$$

Vegyük észre, hogy most már a megfogó orientációját, azaz az \mathbf{R}_{RTi} mátrixot is változtatjuk a képek készítése során. A fenti egyenletben a \mathbf{t}_{TC} és \mathbf{t}_{OR} vektorok az ismeretlenek. Átrendezve az egyenletet:

$$\mathbf{R}_{OR}^{-1}\mathbf{t}_{OCi} - \mathbf{t}_{RTi} = \mathbf{R}_{RTi}\mathbf{t}_{TC} + \mathbf{R}_{OR}^{-1}\mathbf{t}_{OR} \quad (3.21)$$

Bevezetve:

$$\begin{aligned} \mathbf{p}_i &:= \mathbf{R}_{OR}^{-1}\mathbf{t}_{OCi} - \mathbf{t}_{RTi} \\ \mathbf{A}_i &:= \mathbf{R}_{RTi} \\ \mathbf{B} &:= \mathbf{R}_{OR}^{-1} \end{aligned} \quad (3.22)$$

Kapjuk, hogy:

$$\mathbf{p}_i = \mathbf{A}_i\mathbf{t}_{TC} + \mathbf{B}\mathbf{t}_{OR} \quad (3.23)$$

Ha az összes M egyenletet egybefoglaljuk:

$$\mathbf{Cx} = \mathbf{k} \quad (3.24)$$

ahol:

$$\begin{aligned} \mathbf{x} &:= \begin{bmatrix} \mathbf{t}_{TC} \\ \mathbf{t}_{OR} \end{bmatrix}, \quad \mathbf{C} := \begin{bmatrix} \vdots & \\ \mathbf{A}_i & \mathbf{B} \\ \vdots & \end{bmatrix}, \quad \mathbf{k} := \begin{bmatrix} \vdots \\ \mathbf{p}_i \\ \vdots \end{bmatrix} \\ \mathbf{x} &\in \mathbb{R}^{6 \times 1}, \quad \mathbf{C} \in \mathbb{R}^{3M \times 6}, \quad \mathbf{k} \in \mathbb{R}^{3M \times 1}, \quad i = 1..M \end{aligned} \quad (3.25)$$

Innen a pszeudoinvertz segítségével az optimális megoldás számolható:

$$\hat{\mathbf{x}} = (\mathbf{C}^T \mathbf{C})^{-1} \mathbf{C}^T \mathbf{k} \quad (3.26)$$

4. fejezet

Kulcspontkeresés és -párosítás

4.1. Kulcspontkereső algoritmusok

A kulcspontkereső algoritmusok célja olyan pontok megtalálása a képen, amelyek valamely környezete vizuálisan érdekes, jól elkülöníthető, egyedi. Céljuk továbbá, hogy ezekhez a pontokhoz olyan jellemző mennyiséget rendeljenek, amely alapján az adott pont egy másik képen is megtalálható. Ennek a mennyiségnak minél inkább függetlennek kell lennie a kép készítésének minél több körülményétől. A mennyiséggel szemben támasztott követelmény tehát, hogy legyen invariáns a képalkotáskor előforduló minél többféle transzformációra, mint például a skálázás, az elforgatás, vagy a fényerő változása. A jellemző mennyiség a legtöbb algoritmus esetében egy tulajdonság-leíró vektor, vagy leíróvektor, röviden csak leíró.

4.1.1. SIFT kulcspontkeresés

A SIFT (Scale Invariant Feature Transform) [8] egy kulcspontkereső algoritmus. Az általa generált leíró egy 128-elemű vektor, amely invariáns eltolásra, elforgatásra, skálázásra, a megvilágítás kismértékű megváltozására, zajra, és a kamera pozíciójának kismértékű megváltozására. Nem invariáns a projektív transzformációra, tehát a kamera helyzetének jelentős megváltozására.

Az algoritmus főbb lépései a következők:

1. Kulcspont-jelöltek skálainvariáns detektálása
2. Szubpixeles lokalizáció
3. Kulcspont-jelöltek szűrése
4. Jellemző orientáció meghatározása
5. Lokális orientáció-hiszogramok számítása
6. Leíró-generálás

Kulcspont-jelöltek skálainvariáns detektálása

Az algoritmus első lépéseként a képen érdekesnek tűnő pontokat keresünk. Ezt DoG (Difference of Gaussians) szűrőkkel érjük el. A DoG szűrő a képet különböző mértékű (szórású) Gauss-szűrésnek veti alá, majd ezek különbségét veszi. Így az élkiemeléshez hasonló képet kapunk, ám legjobban a szórás méretébe eső térfrekvenciájú részletek kerülnek kiemelésre. A szűrést különböző szórású kernelekkel elvégezve különböző mérettartományokba eső képrészleteket emelhetünk ki. Az így megtalált szélsőértékeket tekintjük kulcspont-jelöltnek.

Szubpixeles lokalizáció

A kulcspontok szubpixeles lokalizációjához az egyes dimenziókban (x , y , skála) másodfokú közelítést alkalmazunk, és így keressük meg a valódi szélsőérték helyét pixel alatti és skálafaktornál nagyobb pontossággal.

Kulcspont-jelöltek szűrése

Az algoritmus eddig a lépésig sok olyan pontot is megjelölt, amelynek képi helyzete nem egyértelmű, pl. egy homogén régiót, vagy élt jellemzők a képen, nem pedig egy pontot. Ezek kerülnek kiszűrésre ebben a lépésben. A szűrés során az adott pontban kiszámoljuk a másodfokú deriváltakból álló Hesse-mátrixot. Ennek a mátrixnak a sajátértékei szolgálnak információval a pont környezetéről. Amennyiben minden két sajátérték kicsi, a pont környezete homogén. Ha az egyik kicsi, a másik pedig nagy, élszerű pontról beszélhetünk. A "kicsi" és "nagy" fogalmak természetesen kvalitatívak, a számszerű értékek tapasztalati úton kerülnek meghatározásra, az algoritmus paramétereit. Ezeket az élszerű és homogén környezetű pontokat ebben a lépésben eldobjuk.

Jellemző orientáció meghatározása

Az orientációhoz hisztogramot használunk, amelynek 36 osztálya van, mindegyik 10° -ot fed le. A kulcspont környezetében minden pontban kiszámítjuk a gradiensvektort, majd ennek irányának megfelelően a hisztogram adott osztályába tesszük. A maximális orientációt másodfokú interpolációval alávetve határozzuk meg a jellemző orientációt. Amennyiben több kiugróan gyakori orientáció van a hisztogramban, a kulcspontot lemásoljuk, és mindegyik jellemző orientációval újat hozunk létre. Így érjük el az elforgatás-invarianciát.

Lokális orientáció-hisztogramok számítása

A képpont 16×16 pixeles környezetét felosztjuk 4×4 darab 4×4 -es blokkra. minden blokkhoz létrehozunk egy 8-osztályos orientáció-hisztogramot, amibe az adott pixeleknek az előző lépésben meghatározott orientációhoz képesti relatív gradiens-irányt kerül.

Leíró-generálás

A leíró-vektor elemeit az előző lépésekben kiszámolt hisztogram osztályainak gyakoriságai adják. A 4x4 darab 8-osztályos hisztogramból így 128-elemű vektort kapunk. Ezt a vektort megvilágítás-invariancia céljából normalizáljuk.

4.1.2. SURF kulcspont-keresés

A SURF (Speeded Up Robust Features) [1] algoritmus hasonló elven működik, mint a SIFT, ám az egyes lépésekhez különböző megoldásokat alkalmaz. A kulcspont-jelöltek keresésénél a Gauss-szűrést ún. box-filterrel közelíti, amit az integrális képből számol. Az integrális kép (x, y) koordinátájú pixelének intenzitását az alábbi összefüggés adja:

$$S(x, y) = \sum_{i=0}^x \sum_{j=0}^y I(i, j) \quad (4.1)$$

Ahol $S(x, y)$ az integrális kép intenzitása az (x, y) pontban, $I(i, j)$ pedig az eredeti kép intenzitása az (i, j) pontban.

Az integrális kép tehát azon pixelek összegét tartalmazza, amelyek egy megadott téglalapon belül találhatók. E téglalap bal felső sarka a kép bal felső sarkával esik egybe, a jobb alsó pedig az (x, y) pixel. Az integrális kép használatának előnye, hogy egy adott téglalapon belüli pixelek összegzését nagyon megyorsítja, amire minden a box-filterhez, minden a Hesse-mátrix közelítéséhez szükség van.

Az algoritmus azokat fogja érdekes pontoknak venni, ahol a Hesse-mátrix determinánsa lokálisan maximális. A pontosság érdekében itt is interpoláljuk a megoldást. Az orientáció meghatározásnál egy adott Euklideszi norma szerinti (kör alakú) környezeten belül található pontok (Haar-wavelet válaszával közelített) gradiens-irányát veszi figyelembe, és egy csúszóablakkal történő kereséssel határozza meg a domináns orientációt. A leíró-generálásnál a kulcspont környezetét 4x4-es blokkokra osztja, és ezekhez a blokkokhoz rendel egy négyelemű vektort, amit a gradiensek alapján számít. Így 4x4x4, azaz 64-elemű leíróvektort kapunk.

4.2. Leíró-párosító algoritmusok, párosítás-szűrések

A kulcspontdetektáló algoritmus megkereste a képen az érdekes pontokat, leíró-vektort generált hozzájuk. Ezt két különböző képen elvégezve az így kapott leírókat párosítanunk kell, ha megfeleltetéseket akarunk találni a kulcspontok között. Mivel a leírók n-dimenziós vektorok, ezért két leíró közötti hasonlóságot egyszerűen lehet számszerűsíteni, pl. a különbségük L2 normájával. A probléma abból adódik, hogy nagyméretű képeken sok (1000-es nagyságrendű) leíró van, és ezeket kell másik, hasonló méretű leíró-halmazhoz párosítani.

A legegyszerűbb megoldás a brute force módszer, vagyis mindegyiket mindegyikkel összehasonlítjuk. Ez lassú ugyan, de garantáltan a legjobb megoldást adja. Egy másik megoldás a FLANN (Fast Library for Approximate Nearest Neighbors) algoritmus [9], ami gyors, de csak közelítő megoldást ad.

Az algoritmusok garantáltan sok hibás párosítást is eredményeznek, ezeket feltétlenül szűrni kell, amelyekre több módszer is van, a következőkben ezeket mutatom be. Mindegyik algoritmus bemutatása során egy d vektorhoz keresünk párt a B halmazból, ahol d egy leíróvektor az egyik képen megtaláltak közül, B pedig a másik képen megtalált leírók halmaza. Az alábbiakban a [2]-ben található, valamint az epipoláris egyenesekkel történő szűréseket mutatom be.

4.2.1. Arányteszt alapú párosítás-szűrés

Ez a heurisztikus módszer azon a feltételezésen alapul, hogy a helyes párosítás esetén d sokkal közelebb van a párjához, mint más vektorokhoz. Hamis párosításnál azonban semmi nem indokolja, hogy a legjobb párosítás lényegesen jobb legyen a többinél, mondjuk a második legjobbnál.

Éppen ezért ez a szűrés azokat a párosításokat tartja meg, amelyeknél a legjobb megtalált B -beli vektor (b_{best}) és a második legjobb vektor (b_{best2}) közötti távolságok aránya nagy:

$$\frac{\|d - b_{best}\|}{\|d - b_{best2}\|} \ll 1 \quad (4.2)$$

A gyakorlatban az aránynak a küszöbértékét 0,6..0,8 közé szokás választani.

4.2.2. Kereszellenőrzéses szűrés

Ez a módszer azt a feltételezést használja, hogy ha d és e pontok párok a két képen, akkor nincs olyan más vektor e -n kívül az ō halmazában, ami d -hez közelebb lenne, viszont e -hez sem találni d -nél közelebb lévő vektort. Ez a szűrés akkor fogadja el a párt, ha kölcsönösen a legközelebb helyezkednek el egymáshoz.

4.2.3. Epipoláris egyenes használata

Amennyiben ismerjük a két képhez tartozó kamerapozíciót, az első képen megtalált összes ponthoz definiálható a második képen egy síkbeli egyenes, amely egyenes mentén a pont párja elhelyezkedhet. Ezt hívjuk epipoláris egyenesnek. Ennek az információnak a birtokában a párosításnál vehetjük eleve csak az epipoláris egyenesre illeszkedő pontokat, és azokhoz párosítjuk a d leírót, vagy a párosítás után eldobhatjuk azokat a párokat, ahol a megtalált kulcspont nem illeszkedik az egyenesre, így kevesebb párt kapunk majd, viszont gyorsabb a számítás.

4.2.4. Homográfia-alapú szűrés

A homográfia megfeleltetés két kép pontjai között. Amennyiben a képek egy sík tárgyról (pl. könyvborítóról) készültek és egy adott tárgypontnak a képi koordinátái ismertek az egyik képen, a homográfia segítségével meghatározhatók ugyanannak a tárgypontnak a képi koordinátái a másik képen is. A homográfia meghatározható véges sok pontpár segítségével. A hibás párosításokat RANSAC algoritmussal kiszűrhetjük, amikor a párosításokra homográfiát illesztünk, így az algoritmus nagy mennyiségű hibás adat esetén is jól működik. A módszer hátránya, hogy csak sík objektumokra használható.

5. fejezet

3D rekonstrukció

A 3D rekonstrukció során egy térbeli objektum pontjainak koordinátáit kívánjuk meghatározni. Ez sok esetben különböző nézőpontból készült képek alapján történik. A problémát megoldó algoritmusokat a szakirodalomban *Structure from Motion (SfM)*, vagy *Multi-View Stereo (MVS)* algoritmusoknak hívják. Amennyiben a kamera külső paramétereit ismertek, a probléma lineáris legkisebb négyzetek módszerével megoldható. A külső paraméterek hiányában a feladat nemlineáris legkisebb négyzetek problémává alakítható, amelyet cél-szerű Levenberg-Marquardt algoritmussal megoldani. Ennek a nemlineáris problémának a megoldását *bundle adjustment*-nek hívják.

A 3D rekonstrukció kimenete általában egy pontfelhő. Amennyiben csak kulcspontok párosítását használjuk, és azok térbeli helyzetét határozzuk meg, ún. ritka pontfelhőt kapunk. Ha teljes 3D-s modellt akarunk kapni, vagyis a teljes megfigyelt környezetet vissza akarjuk állítani, akkor sűrű pontfelhőre van szükségünk. Jelen alkalmazásban a ritka pontfelhő elegendő.

5.1. Többnézetes háromszögelés

Mivel számunkra ismertek a kamera külső (és belső) paraméterei, ezért a feladat egy (túlhatározott) lineáris egyenletrendszer megoldása. A probléma precíz megfogalmazása a következő: Adott egy ismeretlen helyzetű 3D-s tárgypont, amelyről N darab képet készítettünk ismert térbeli helyzetből. A tárgypontnak megfelelő képi pontokat az összes képen ismerjük. Keressük azt a 3D-s pontot, amelyet a kamerák képsíkjaira visszavetítve az ismert képi pontuktól való távolságok négyzetösszege minimális.

A (2.1) egyenlet alapján írható:

$$w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{P}_i \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad \mathbf{P}_i \in \mathbb{R}^{3 \times 4}, \quad i = 1..N \quad (5.1)$$

Az (5.1) egyenlet minden képre felírható, ismertek \mathbf{P}_i, u_i, v_i , és keressük x, y és z értékét. Ha bevezetjük, hogy:

$$\mathbf{P}_i := \begin{bmatrix} \mathbf{p}_{1i} & a_{1i} \\ \mathbf{p}_{2i} & a_{2i} \\ \mathbf{p}_{3i} & a_{3i} \end{bmatrix}, \quad \mathbf{p}_{1i}, \mathbf{p}_{2i}, \mathbf{p}_{3i} \in \mathbb{R}^{1 \times 3} \quad (5.2)$$

valamint:

$$\mathbf{x} := \begin{bmatrix} x & y & z \end{bmatrix}^T \quad (5.3)$$

Az (5.1) egyenlet írható a következő formában:

$$\begin{aligned} w_i u_i &= \mathbf{p}_{1i} \mathbf{x} + a_{1i} \\ w_i v_i &= \mathbf{p}_{2i} \mathbf{x} + a_{2i} \\ w_i &= \mathbf{p}_{3i} \mathbf{x} + a_{3i} \end{aligned} \quad (5.4)$$

A fenti két egyenletbe az alsót behelyettesítve adódik:

$$\begin{aligned} (\mathbf{p}_{3i} \mathbf{x} + a_{3i}) u_i &= \mathbf{p}_{1i} \mathbf{x} + a_{1i} \\ (\mathbf{p}_{3i} \mathbf{x} + a_{3i}) v_i &= \mathbf{p}_{2i} \mathbf{x} + a_{2i} \end{aligned} \quad (5.5)$$

Ezt az egyenletet \mathbf{x} -rendezve és az összes képre felírva kapjuk, hogy:

$$\mathbf{Gx} = \mathbf{b} \quad (5.6)$$

ahol:

$$\mathbf{G} := \begin{bmatrix} \vdots & \\ \mathbf{p}_{3i} u_i - \mathbf{p}_{1i} & \\ \mathbf{p}_{3i} v_i - \mathbf{p}_{2i} & \\ \vdots & \end{bmatrix}, \quad \mathbf{b} := \begin{bmatrix} \vdots & \\ a_{1i} - a_{3i} u_i & \\ a_{2i} - a_{3i} v_i & \\ \vdots & \end{bmatrix} \quad (5.7)$$

Innen a pszeudoinvertor segítségével számolható az optimális megoldás:

$$\hat{\mathbf{x}} = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \mathbf{b} \quad (5.8)$$

Fontos megjegyezni, hogy ez a megoldás nem a visszavetítési hibát minimalizálja, hanem az egyenletrendszer algebrai hibáját ($\mathbf{G}\hat{\mathbf{x}} - b$ mennyiséget). Amennyiben ez lényeges eltérést

okoz, nemlineáris optimumkereséssel meghatározható a valódi optimum. Ezt az implementáció során elhanyagoltam.

5.2. Leíró-párosító algoritmusok sajátosságaiból adódó problémák

A feladat bonyolultsága fokozható, ha figyelembe vesszük, hogy a párosító algoritmusok téves párosításokat is tartalmaznak, valamint, hogy egy leíróhoz lehet, hogy csak kevés (egy vagy kettő) másikat sikerül párosítani még több kép esetén is. Minél több párt talál a párosító algoritmus egy leíróhoz, annál pontosabban határozható meg a hozzá tartozó kulcspontról 3D pozíciója.

A párosításokat értelmezhetjük ritka, nem összefüggő gráfként, ahol a csúcsok a kulcspontról és azok leírói, az élek pedig a párosításokat jelentik. Értelemszerűen ha feltételezzük, hogy minden párosítás helyes, akkor minden összefüggő részgráf egy-egy tárgyponthoz tartozó leírókat tartalmaz.

A tesztek alapján, amiket végeztem, azt lehet mondani, hogy 5 tesztkép esetén mindegyiket mindegyikkel párosítva a csomópontok több, mint feléből csak egyetlen él fut ki, azaz a hozzájuk tartozó leírókhöz a többi 4 képből csak egyen sikerült párt találni. A tesztek azt támasztják tehát alá, hogy a gráf kellően ritka.

5.2.1. A ritkaság korrigálása

A gráf sűrítése érdekében megtehetjük azt, hogy pl. vesszük az összes olyan csúcs-hármasat, amelyik majdnem teljes részgráfot (kvázi-klikket) alkot, vagyis a 3 élből csak egy hiányzik, és teljessé tesszük, bízva abban, hogy a párosító algoritmus valamilyen oknál fogva a behúzott élnek megfelelő párosítást nem találta helyesnek, pedig az. Ezt megtehetjük bármilyen csúcs- n -re is.

5.2.2. Téves párosítások szűrése

A téves párosítások figyelembe vételéhez érdemes RANSAC elven működő algoritmust alkalmazni. Ekkor a meghatározandó modell a 3D pozíció lesz. A 3D pozíciót visszavetítve a képekre az illeszkedés mértékét definiálhatjuk az így kapott képi pontok és az eredetileg detektált pontok közti négyzetes távolságösszegként.

6. fejezet

Megvalósítás

6.1. A szoftver feladatai

A konkrét implementáció a BME IIT robotlaborjában történt egy Mitsubishi robotkaron. A komplett, több komponensből álló szoftver feladatai a következők:

- A robotkar mozgatása a PC-n bevitt helyzetbe.
- A kéz-szem kalibráció automatikus elvégzése.
- A 3D rekonstrukció elvégzése.
- A tárgy helyzetének meghatározása 3D pontfelhőt tartalmazó adatbázis alapján.

6.2. Kiválasztott implementáció

A tárgy helyzetének meghatározásához készítünk egy képet a tárgyról. Ezen a képen kulcs-pontkeresést hajtunk végre, majd párosítjuk ezeket a pontokat a 3D pontfelhő pontjaival. A pozíció és orientáció becslése PnP algoritmussal történik.

Az előző fejezetekben a legtöbb probléma megoldására több módszert is ismertettem. Itt szeretném bemutatni az implementációhoz kiválasztottakat.

6.2.1. Kulcspontkeresés és -párosítás implementációja

A kulcspontkeresés SURF algoritmussal történik.

A párosító algoritmus minden esetben a 4.2 szakaszban ismertetett brute force párosító, mivel az több párosítást eredményez, mint a FLANN algoritmus.

A 3D rekonstrukció során a kulcspontpárosításnál ismerjük a külső paramétereket, ezért a 4.2.3 és 4.2.4 szakaszokban ismertetett, epipoláris egyeneseket és homográfiát használó szűrést alkalmazom. A homográfiás szűrés esetén RANSAC elvű algoritmust használok.

Mivel a tárgy pozíciójának becslésekor épp a külső paramétereket kívánjuk meghatározni, ezért itt az epipoláris egyenesek használata nem lehetséges. Ennél a lépésnél a 4.2.2 szakaszban ismertetett kereszttellenőrzéses szűrést alkalmazom.

6.2.2. A kéz-szem kalibráció implementációja

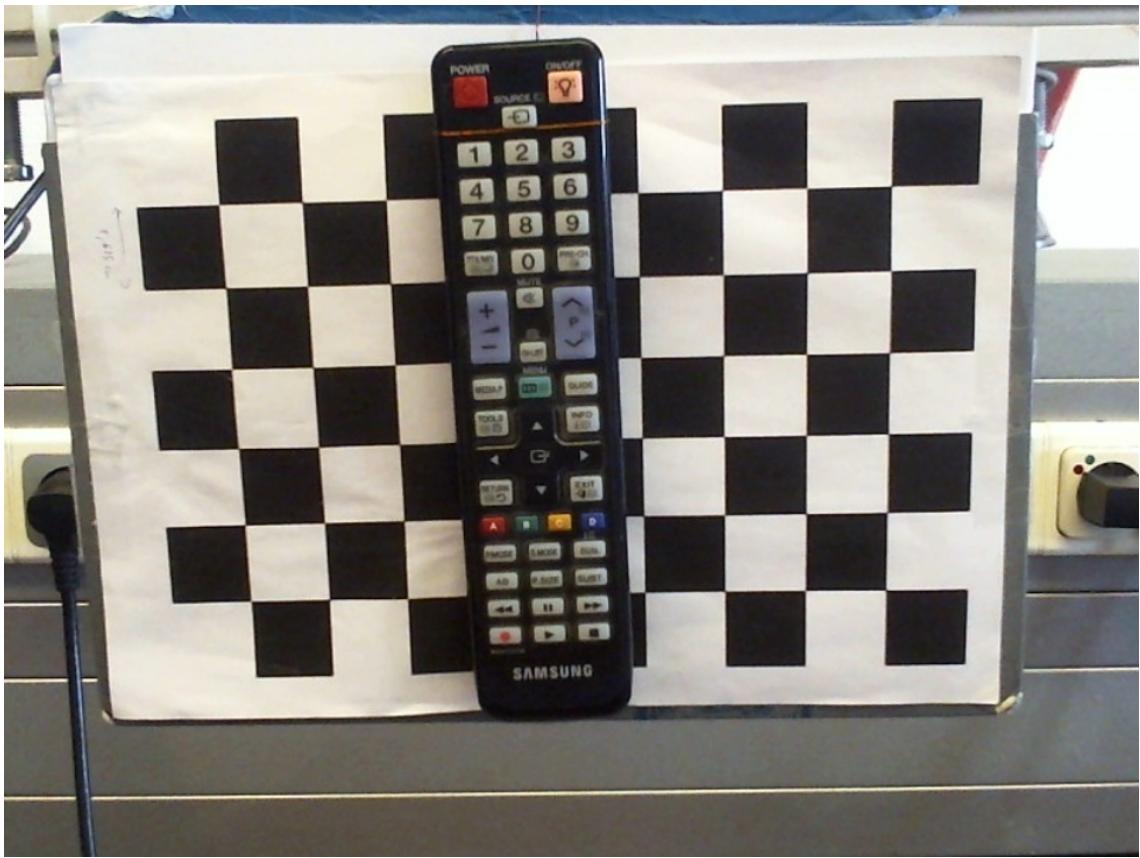
A kéz-szem kalibrációnál különböző nézőpontokból képeket kell készíteni egy kalibráló objektumról. Először az orientáció megtartásával, majd annak változtatásával. Sajnos a megvalósításhoz használt robotkar munkatere nem túl nagy, valamint a kamera látószöge is elég kicsi, ezért a kamera helyzetét nem lehet széles tartományban változtatni. Ez a kéz-szem kalibráció esetében az első fázisban (állandó orientáció) ± 5 cm-es tartományt jelent minden három tengely mentén. Így a program 11 képet készít mind az x , y és z tengely mentén. A második fázisban a kalibráló objektum közepét a kép közepén tartva különböző szögekből készít képeket az algoritmus. Az x tengely körül $\pm 15^\circ$ elfordulást képes megvalósítani, az y tengely körül a $[-9^\circ, +30^\circ]$ tartományban lehet változtatni az orientációt. Mindkét irányban szintén 11-11 képet készít a program.

A kalibráló objektum egy sakktáblamintázat volt, amit a 6.1 ábrán is láthatunk. A minta sarokpontjainak meghatározásához az OpenCV-ben kész függvény van megírva, ezért ez nagyban megkönnyíti a kalibráció implementálását.

6.2.3. 3D rekonstrukció és a kéz-szem kalibráció implementációja

A 3D rekonstrukció során a megtalált kulcspontpárok alapján a program egyszerű háromszögeléssel számít 3D pozíciót. A többnézetes háromszögelés módszerét végül elvettem.

Ahogy a kéz-szem kalibrációnál, úgy a 3D rekonstrukciónál is különböző nézőpontból kell képeket készíteni. Itt azonban a rekonstruálandó tárgyról készülnek a képek. Itt is probléma a limitált munkatér. A megvalósítás során az egyik tengely mentén $[-10^\circ, +30^\circ]$, a másik mentén pedig $\pm 30^\circ$ tartományban készít a program képeket a tárgyról. A képek középpontjában ugyanúgy a tárgy "középpontját" tartja, viszont a két tengely körüli elforgatást egymástól függetlenül hajtja végre. Mindkét irányban 7 pontra osztottam fel az elfordulást, így összesen 49 kép készül a tárgyról. A külső paraméterek meghatározásához PnP algoritmust használ a program. Ugyanazt a sakktáblamintát használom, mint a kalibráció során.



6.1. ábra. A rekonstrukcióhoz használt elrendezés. A rekonstruálendő tárgy egy TV-távirányító.

6.2.4. A pozíció és orientáció meghatározásának implementációja

A pozíció és orientáció meghatározása a PnP algoritmus RANSAC elvű megvalósításával történik. Amennyiben ismerjük a kamera pozícióját a tárgy koordináta-rendszerében, a (3.2) egyenlet alapján kiszámolhatjuk a tárgy és a robot koordináta-rendszerök közti transzformációt, \mathbf{T}_{OR} -t. Ennek az ismeretében pedig a tárgyhoz képesti tetszőleges célkonfigurációba irányíthatjuk a robotkart.

A tárgy pozíciójának meghatározásához a szoftver öt különböző nézőpontból készít képeket. Mindegyikhez megpróbálja meghatározni a kamera külső paramétereit, és végül a mellett a konfiguráció mellett dönt, amelyet a legtöbb inlier alapján sikerül meghatározni. Ez után kiszámítja azt a konfigurációt, amelyben a kamera egyenesen ránéz a 6.2(c) ábrán bekarikázott gombra, és majdnem érinti a távkapcsolót, ahogy az a 6.2(a) ábrán látszik. Ez a megfogási konfiguráció. A program azonban (lehetséges ütközésekkel elkerülni) a megfogási konfigurációhoz képest egy eltolt célkonfigurációba irányítja a robotot. A célkonfiguráció a világ koordináta-rendszerben pozitív z irányban (fölfele) 20 cm-rel van eltolva a megfogási konfigurációhoz képest. Ezt a 6.2(b) ábrán látjuk.



(a) Megfogási konfiguráció

(b) Célkonfiguráció



(c) A kamera képe a célkonfigurációban.

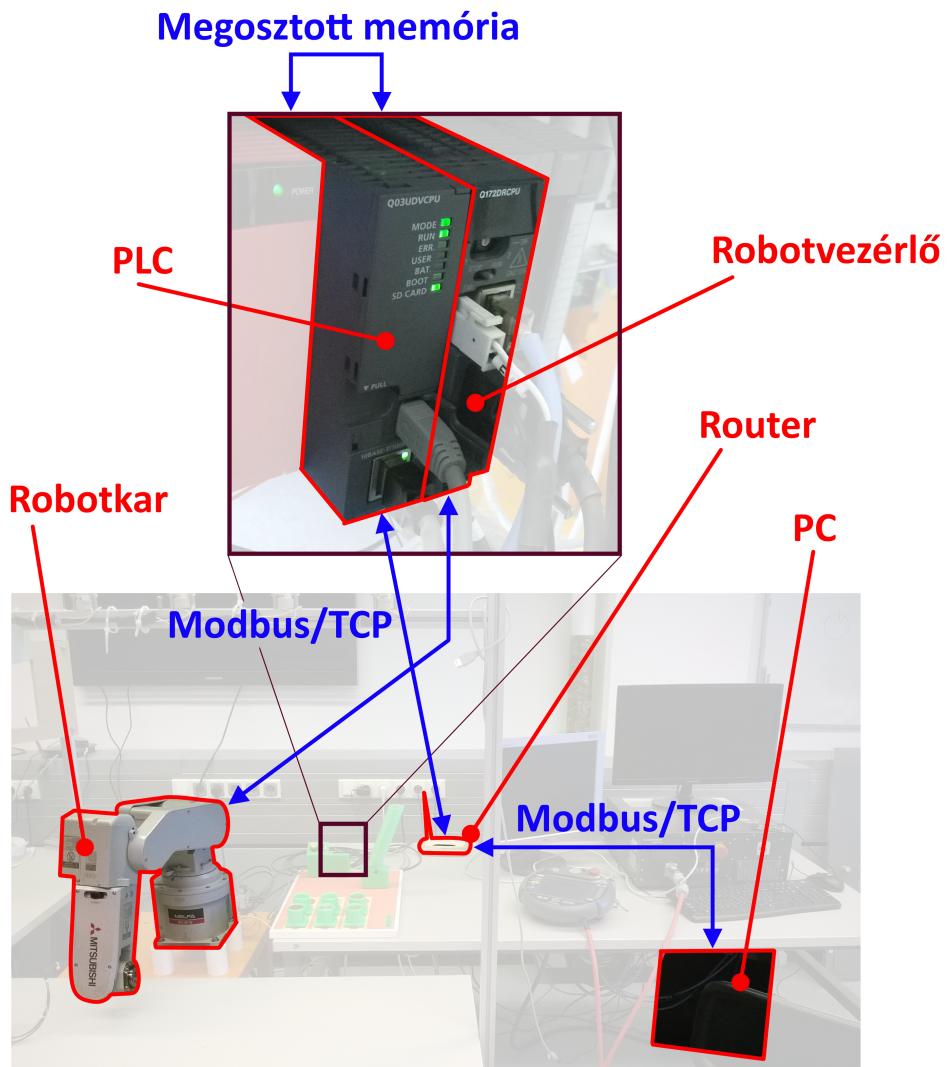
6.2. ábra. A célkonfiguráció meghatározása

6.3. A robotirányító rendszer felépítése

A komplett rendszer hardverkomponensei között lévő kommunikációs lánc a következő elemekből épül fel:

1. PC
2. PLC (Q03UDVCPU)
3. Robotvezérlő modul (Q172DRCPU)
4. Robotkar (Melfa RV-2F-Q)

A kommunikáció a következő módon valósul meg. A PC és PLC között a kommunikáció Modbus/TCP kapcsolaton keresztül történik, a PLC a belső hálózaton DHCP-vel kiosztott statikus IP-n elérhető. A Modbus/TCP egy Ethernet-alapú kommunikációs protokoll, amelyben a szerver elérhetővé teszi a hozzá csatlakozott kliensek számára a regisztereit, azok szabadon írhatók, olvashatók. A PLC és a robotvezérlő modul megosztott memórián keresztül kommunikálnak. A robotvezérlő közvetlenül irányítja a robotkart, azzal egy komplett berendezést alkot, a köztük lévő kommunikáció megoldása nem a felhasználó feladata.



6.3. ábra. A robotirányító rendszer felépítése.

A robot megfogójára egy Logitech c525 típusú webkamera van szerelve, ez szolgáltatja a képfeldolgozás számára a vizuális információt.



6.4. ábra. A robotra szerelt kamera.

6.4. A PLC és robotprogram

A PLC-n két, a működéshez szükséges program fut. Az egyik egy Modbus szerver, ami a PC-vel való kommunikációt kezeli. Ez a Mitsubishi cég (és a programot korábban felfedező és használó hallgatók) jóvoltából már rendelkezésre állt számomra.

A másik program a robot felé történő kommunikációért felelős. Minthogy a PLC csak összekötő szerepet játszik a robot és a PC között, ezért ennek a programnak a feladatköre arra korlátozódik, hogy a Modbus-on keresztül kapott értékeket továbbítsa a megosztott memóriába. Ezt a programot Menyhárt Balázs MSc hallgató készítette, én minimális módosításokkal az ő programját használom.

A roboton futó program kiolvassa a megosztott memóriából a megfelelő értékeket, és ennek megfelelően mozgatja a robotkart. Ennek a megvalósítása úgy történik, hogy egy-egy regiszterben kerül átadásra a pozíció-orientáció 6 paramétere. A robotprogram elvégzi a megfelelő korrekciókat (szög-radián átváltás), majd a megadott helyzetbe irányítja a megfogót. Ez ki van még egészítve egy utasításszámlálóval, amelynek értékét a PC-n futó programban vizsgálva megtudhatjuk, hogy az adott utasítást a robot elvégezte-e már.

6.5. A PC program

A PC programot Python nyelven írtam. Képfeldolgozásra az OpenCV függvénykönyvtárat, a Modbus kommunikáció megvalósításához a PyModbusTCP könyvtárat használtam.

Csatlakozás után a megadott helyzetbe lehet irányítani a robotot, illetve a megfelelő gomb megnyomásával a kéz-szem kalibráció és a tárgy megkeresése is elvégezhető.



6.5. ábra. A program felhasználói felülete.

A legtöbb általam használt algoritmus OpenCV-s implementációját használom. Saját implementációk közül a legfontosabb a kéz-szem kalibráció algoritmusa és az 5.1 fejezetben ismertetett háromszögelés algoritmusa.

A Kabsch algoritmust implementáló kód részt Jimmy Charnley Kromann és Lars Andersen Bratholm írta, az ő kódjukat használom [7].

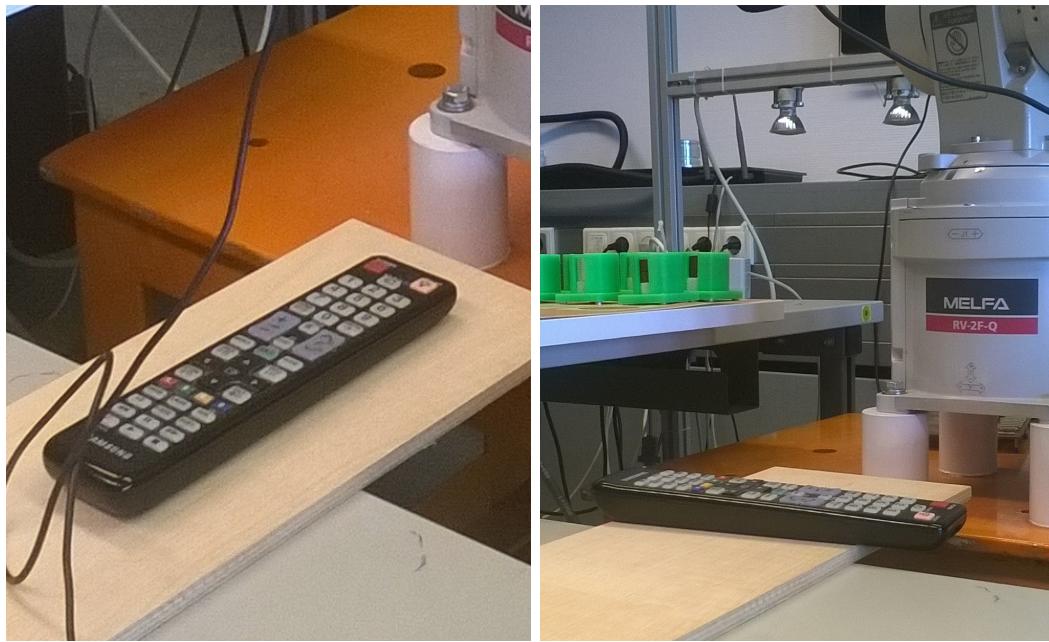
7. fejezet

Eredmények

A szoftver teszteléséhez a megtalálandó tárgy a 6.1 ábrán látható TV-távirányító volt. A választás azért esett erre a tárgyra, mert kellően részletgazdag, és a mérete megfelelő a teszteléshez.

A tárgy pozíciójának meghatározását több elrendezésre is elvégeztem, elrendezéseként többször is lefuttatva a keresést. itt három ilyen teszt eredményét mutatom be, a tesztek során a keresést rendre 7, 7 és 6 alkalommal futtattam le. A megfogási konfiguráció ideális értékét kézzel állítottam be, ehhez képest mértem az ideális célkonfigurációt.

A három elrendezés a következő volt:



(a) 1. elrendezés

(b) 2. elrendezés



(c) 3. elrendezés

7.1. ábra. A különböző elrendezések

Az eredmények közlésénél a következő jelölést használom:

- $\delta_x, \delta_y, \delta_z, \delta$ a két végpont közötti koordinátánkénti és abszolút különbség. $\delta^2 = \delta_x^2 + \delta_y^2 + \delta_z^2$
- $\Delta_x, \Delta_y, \Delta_z, \Delta$ rendre a két orientáció x, y és z tengelyei által bezárt szög és ezek számtani közepe. Amennyiben az orientációk megegyeznek, ezek természetesen nulla-val egyenlők.

Az eredményeket a következő táblázatok mutatják.

δ_x (cm)	δ_y (cm)	δ_z (cm)	δ (cm)	Δ_x ($^{\circ}$)	Δ_y ($^{\circ}$)	Δ_z ($^{\circ}$)	Δ ($^{\circ}$)	inlierek száma
4.23	1.07	2.1	4.84	28.28	21.74	32.24	27.42	189
0.46	-1.5	4.63	4.89	0.4	21.62	21.62	14.55	157
0.81	0.02	4.08	4.16	2.99	5.96	6.6	5.18	76
4.32	1.06	1.97	4.86	27.9	22.76	32.58	27.75	183
-0.15	-0.64	-1.61	1.74	41.91	16.33	44.05	34.10	130
-0.38	-0.3	-1.91	1.97	31.89	3.97	32.07	22.65	152
0.33	0.07	2.25	2.27	10.14	0.37	10.13	6.88	101

7.1. táblázat. Az 1. elrendezés eredményei

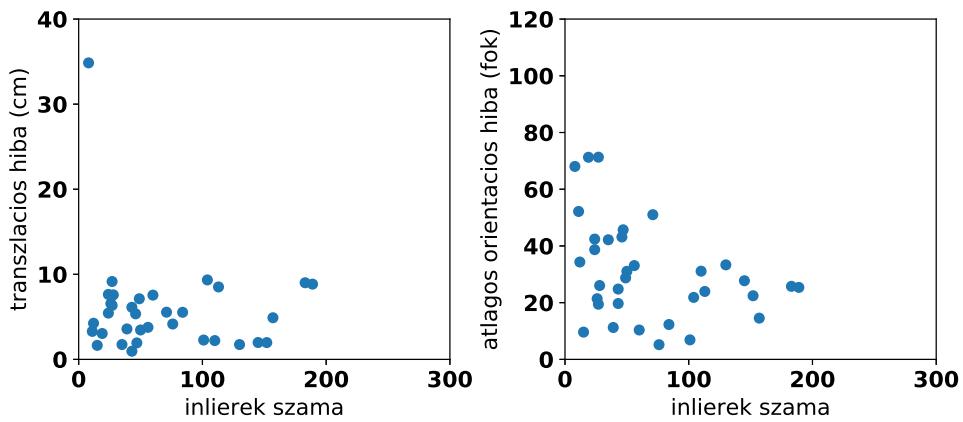
δ_x (cm)	δ_y (cm)	δ_z (cm)	δ (cm)	Δ_x ($^{\circ}$)	Δ_y ($^{\circ}$)	Δ_z ($^{\circ}$)	Δ ($^{\circ}$)	inlierek száma
1.76	-0.58	-5.35	5.66	11.25	34.15	35.77	27.06	116
0.16	-2.24	4.23	4.79	27.93	4.36	28.2	20.16	251
4.24	0.31	3.56	5.55	12.45	19.	22.76	18.07	187
0.	-2.25	4.16	4.73	27.8	3.06	27.92	19.59	274
0.14	-0.75	3.92	3.99	2.02	8.85	8.71	6.52	248
0.03	-2.61	3.29	4.20	23.13	4.56	23.5	17.06	212
-0.11	-0.72	3.72	3.79	3.34	13.95	14.01	10.44	209

7.2. táblázat. A 2. elrendezés eredményei

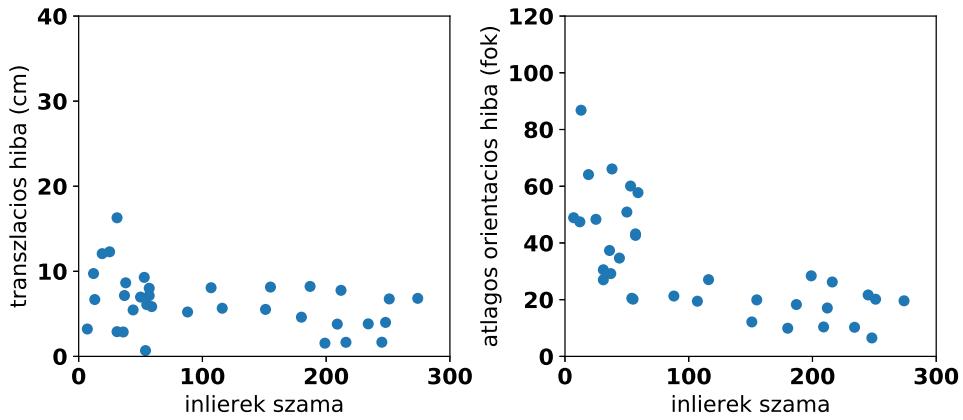
δ_x (cm)	δ_y (cm)	δ_z (cm)	δ (cm)	Δ_x ($^{\circ}$)	Δ_y ($^{\circ}$)	Δ_z ($^{\circ}$)	Δ ($^{\circ}$)	inlierek száma
4.64	3.	-6.46	8.50	5.87	39.6	39.71	28.39	61
1.59	-0.34	3.19	3.58	13.12	16.53	20.95	16.87	115
1.93	-0.63	-3.54	4.08	26.85	32.2	39.79	32.94	71
4.28	-0.52	-5.48	6.97	18.82	75.	75.08	56.30	29
1.31	-1.03	5.48	5.73	10.83	10.03	14.32	11.73	100
1.17	-0.65	4.08	4.30	4.02	9.03	9.64	7.56	190

7.3. táblázat. A 3. elrendezés eredményei

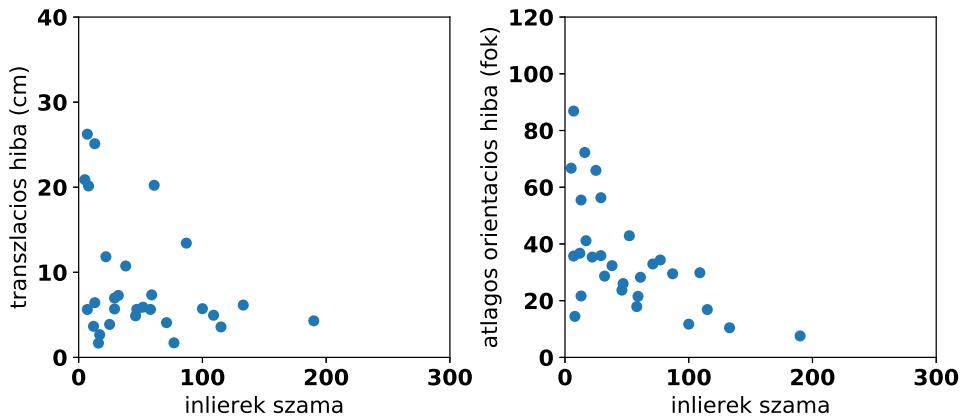
Az alábbi diagramok azt szemléltetik, hogy a különböző mennyiségű inliernél az egyes hibák mekkorák. A keresésnél a program végül 5 mérés eredményéből veszi a legjobbat (legtöbb inlier alapján számítottat). Ezeken az ábrákon minden mérés szerepel, nem csak a legjobbak, mint a fenti táblázatokban.



7.2. ábra. A teszt eredménye az 1. elrendezés esetében.



7.3. ábra. A teszt eredménye a 2. elrendezés esetében.



7.4. ábra. A teszt eredménye a 3. elrendezés esetében.

8. fejezet

Értékelés, továbbfejlesztési lehetőségek

A tesztek eredményein látszik, hogy a program viselkedése kevéssé robusztus, ugyanarra az elrendezésre nem tud reprodukálható eredményt mutatni. A 7.2, 7.3 és 7.4 ábrák alapján azt lehet mondani, hogy az inlierek számának növekedésével a hiba konziszenessé válik és csökken, de a tárgy megfogásához még mindig közel sem elég kicsi.

A hibáknak több oka is lehet. Mind a kéz-szem kalibrációnál, mind a 3D renkonstrukciótól problémát okoz, hogy a kamera látószöge és a robot munkatere kicsi. Nagyobb látószögű kamerát nem lehet használni, mert annak a torzítása is nagyobb. Nagyobb munkaterű robottal lehet ezen a hibán javítani.

Probléma lehet a kalibráló objektum minősége is. Az általam használt sakktáblamintázat hagyományos nyomatopapírra készült, és a tesztek során akaratlanul is meggyűrődött, így a sarkai elmozdultak a feltételezett helyükről. Egy nehezebben deformálódó, pontosabb kalibráló objektum használata sokat lendíthetne a pontosság növelésén.

A kamerát technikai okokból nem tudtam a legnagyobb felbontásán használni, ez lehet, hogy segítene több kulcspontot detektálni a tárgyon és több párosítást használni. Természetesen ez valószínűleg a futásidő rovására menne. További probléma lehet még, hogy a labor megvilágítása nem kifejezetten erős, a képek zajosak, ez is zavarhatja a kulcspont-párosítást.

Melléklet

A mellékletben a PC és robotprogram forrása megtalálható a hozzájuk tartozó utasításokkal a *readme.txt* fájlban.

Ábrák forrásai

(2.1). Ábra: A pinhole-kameramodell.

Forrás: <http://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/>

(2.2). Ábra: A RANSAC algoritmus és a legkisebb négyzetek módszerének összehasonlítása.

Forrás: http://scikit-learn.org/stable/auto_examples/linear_model/plot_ransac.html

Irodalomjegyzék

- [1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.
- [2] Daniel Lelis Baggio et al. *Mastering OpenCV with practical computer vision projects*. Packt Publishing, 2012.
- [3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [4] Xiao-Shan Gao. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.
- [5] Anders Hast, Johan Nysjö, and Andrea Marchetti. Optimal RANSAC - Towards a repeatable algorithm for finding the optimal set. *Journal of WSCG*, 21(1):21–30, 2013.
- [6] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976.
- [7] Jimmy Charnley Kromann and Lars Andersen Bratholm. Root-mean-square deviation (RMSD) of two XYZ structures. <https://github.com/charnley/rmsd>.
- [8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [9] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP’09*, pages 331–340. INSTICC Press, 2009.