Tárgymanipuláció kulcspontok detektálásán alapuló képfeldolgozással

Szakmai gyakorlat beszámoló

Ayhan Dániel
LIIAE1

Témavezető:

Dr. Kiss Bálint

Budapest, 2017. május 2.

Tartalomjegyzék

1. Bevezetés, feladatkitűzés

A szakmai gyakorlatomat a Budapesti Műszaki és Gazdaságtudományi Egyetem Irányítástechnika és Informatika Tanszékén végeztem. A feladat egy tetszőleges tárgy lokalizálása volt képfeldolgozás segítségével. Lokalizálás alatt a tárgy pontos pozícióját és orientációját, azaz *helyzetét* értem. A tárgyról előzetesen bármilyen információnk lehet, célszerű erre az adathalmazra is képfeldolgozással szert tenni.

A választott módszer a következő volt: a tárgyról készült képen keressünk *kulcspontokat*, majd ezekhez generáljunk *leíróvektort*, amely a lehető legtöbb fajta torzításra invariáns. Ezeket a vektorokat két különböző helyzetből készült képen össze lehet párosítani. A kamera helyzetének ismeretében különböző módszerekkel ezekből a párosításokból 3D koordinátákat lehet kapni. Így olyan adathalmazhoz jutottunk, amelyben leíróvektorokhoz (vagy röviden: leírókhoz) 3D koordinátákat rendeltünk. Ezek után a tárgyról egy ismeretlen helyzetből készült képen ugyanezeket a kulcspontokat megkereshetjük. A megtalált kulcspontokhoz számíthatunk leírókat, amelyeket összepárosítva az adathalmaz leíróival megkaphatjuk a képen talált kulcspontok 3D koordinátáit. Innen a megfelelő algoritmussal kiszámolhatjuk a kamera külső paramétereit, azaz az objektumhoz képesti helyzetét. Ezzel meghatároztuk az objektum helyzetét a kamera koordináta-rendszerében.

2. A gépi látás alapfogalmai

Az algoritmusok áttekintése előtt ismerkedjünk meg a gépi látásban, képfeldolgozásban használatos fogalmakkal. A kamerát 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.

A kamerához definiálható egy kamera koordináta-rendszer. Ennek a koordináta-rendszernek az origójának a vetítés középpontját tekintjük, 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. 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:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{23} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

A vetítés mátrixát kameramátrixnak hívjuk, a paramétereit pedig a 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 pontpár segítségével. A pontpárok alatt a világ koordináta-rendszerben ismert pozíciójú pontokat és azok kép koordináta-rendszerben megadott vetületét értjük.

A kameramátrixot a feladat megoldása során ismertnek feltételeztem, hiszen az időben állandónak tekinthető.

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 elhanyagoljuk, így a fenti összefüggésben a γ paraméter 0 lesz.

3. 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, és ugyanazt a jellemző mennyiséget eredményezi. Ezzel a mennyiséggel szemben támasztott követelmény az is, hogy legyen invariáns minél többféle képalkotáskor előforduló transzformációra, mint például a skálázás, elforgatás. A jellemző mennyiség a legtöbb algoritmus esetében egy tulajdonság-leíró vektor, vagy leíróvektor, röviden csak leíró.

3.1. SIFT kulcspontkeresés

A SIFT (Scale Invariant Feature Transform) 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-hisztogramok számítása
6. Leíró-generálás

3.1.1. 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 kapott képeken megtalált szélsőértékeket tekintjük kulcspont-jelöltnek.

3.1.2. 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.

3.1.3. Kulcspont-jelöltek szűrése

Az algoritmus ezen a pontján sok olyan pontot is megjelölt, amelynek képi helyzete nem egyértelmű, pl. egy homogén régiót, vagy élt jellemeznek 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 mindkét 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 bemeneti paraméterei. Ezeket az élszerű és homogén környezetű pontokat ezen a ponton eldobjuk.

3.1.4. 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 interpolációnak alávetve határozzuk meg a jellemző orientációt. Amennyiben több kiugróan nagy orientáció van a hisztogramban, a kulcspontot lemásoljuk, és mindegyik orientációval újat hozunk létre. Így érjük el az elforgatás-invarianciát.

3.1.5. Lokális orientáció-hisztogramok számítása

A képpont 16x16 pixeles környezetét felosztjuk 4x4 darab 4x4-es blokkra. Minden blokkhoz létrehozunk egy 8-osztályos orientáció-hisztogramot, amibe az adott pixeleknek az előző lépésben meghatározott orientációhoz képesti relatív gradiens-iránya kerül.

3.1.6. Leíró-generálás

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

3.2. SURF kulcspont-keresés

A SURF (Speeded Up Robust Features) 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 pixele azon pixelek összegét tartalmazza, amik 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. Érdekes pontoknak azokat fogja venni, ahol a Hesse-mátrix determinánsa lokálisan maximális. Itt is a pontosság érdekében 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. Leíró-párosító algoritmusok, párosítás-szűrések

Miután 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 a kulcspontok között találni. Mivel a leírók n-dimenziós vektorok, ezért két leíró közötti hasonlóságot egyszerűen lehet számszerűsíteni, pl. a különbségük L2 normájával. A probléma abból fakad, hogy nagyméretű képeken sok (1000-es nagyságrendű) leírónk 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, 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 algoritmusban egy d vektorhoz keresünk párt a B halmazból, ahol d az egyik képen megtalált leírók közül egy, B pedig a másik képen megtalált leírók halmaza.

4.1. 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. Amennyiben egy hamis párosítást veszünk, d vektorhoz a legközelebb eső B-beli vektor valószínűleg nem lesz d-hez sokkal közelebb, mint a második legközelebbi, hiszen a hamis párosításnál a legjobb megoldás nem sokkal jobb a második legjobbnál.

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

$$\frac{\|d - b_{true}\|}{\|d - b_{false}\|} \ll 1 \tag{2}$$

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

4.2. Kereszt-ellenőrzéses szűrés

Ez a módszer azt a feltételezést használja, hogy ha d és e párok a két képen, akkor nincs olyan más vektor e-n kívül az ő 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 d-hez e van legközelebb a második képen talált leírók közül, e-hez pedig d az első képen lévők közül.

4.3. Epipoláris egyenes használata

Amennyiben ismerjük a két képhez tartozó kamerapozíciót, minden ponthoz az első képen definiálható egy egyenes a második képen, 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.

4.4. Homográfia-alapú szűrés

A homográfia olyan transzformáció, ami egy síkbeli alakzat pontjaihoz tartozó képi pontok között teremt kapcsolatot. Segítségével az egyik képen található pont másik képen való pozícióját lehet meghatározni. A homográfia jól párosított pontok segítségével meghatározható. A hibás párosításokat RANSAC algoritmussal kiszűrhetjük, amikor a párosításokra homográfiát illesztünk, így az algoritmus nagy mennyiségű hibás adat esetén is jól működik. Ez a módszer csak sík objektumokra használható.

5. 3D rekonstrukció

A 3D rekonstrukció során egy térbeli objektum pontjainak koordinátáit kívánjuk meghatározni. Ez esetünkben különböző nézőpontból készült képek alapján történik. A problémát megoldó algoritmusokat Structure from Motion (SfM), vagy Multi-View Stereo (MVS) algoritmusoknak hívjuk. Amennyiben a kamera külső paraméterei ismertek, a probléma lineáris legkisebb négyzetek módszerével megoldható. A külső paraméterek hiányában a feladat nemlineáris legkisebb négyzetek problémává alakítható, amelyet célszerű Levenberg-Marquardt algoritmussal megoldani. Ennek a nemlineáris problémának a megoldása a szakirodalomban *bundle adjustment* néven található meg. Ezzel most nem foglalkozunk, ugyanis számunkra ismertek a kamera külső paraméterei.

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, vagy a teljes megfigyelt környezetet vissza akarjuk állítani, akkor sűrű pontfelhőre van szükségünk. Számunkra a ritka pontfelhő is elegendő.

A pontok térbeli pozíciójának meghatározása háromszögeléssel történik. A probléma precíz megfogalmazása a következő: Adott n pont ismert térbeli helyzetű kamerák által készített képeken. Keressük azt a 3D-s tárgypontot, amelyet a kamerák képsíkjaira visszavetítve az eredeti pontoktól való távolságok négyzetösszege minimális.

5.1. 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.

A párosításokat értelmezhetjük ritka erdő gráfként, ahol a csúcsok a kulcspontok, és azok leírói, az élek pedig a párosításokat jelentik. Értelemszerűen ha feltételezzük, hogy minden párosítás helyes, akkor minden fa (azaz egybefüggő 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 leírókhoz a többi 4 képből csak egyen sikerült párt találni. A tesztek azt támasztják tehát alá, hogy a gráf kellően ritka.

5.1.1. A ritkaság korrigálása

A gráf sűrítése érdekében megtehetjük azt, hogy pl vesszük az összes olyan csúcs-hármast, amelyik majdnem teljes algráfot (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-esre is.

5.1.2. Téves párosítások szűrése

A téves párosítások figyelembe vételéhez érdemes RANSAC elven működő algoritmust alkalmazni, vagyis kevés, feltehetőleg helyes adatból számolni az eredményt, majd ehhez a közelítő megoldáshoz hozzávenni az ezzel konzisztens adatokat, és téves adatnak minősíteni az inkonzisztens adatokat. Jelen esetben ez jelentheti például azt, hogy ha veszünk egy fához tartozó csúcsokat, két véletlenszerűen kiválasztott csúcs alapján háromszögeléssel kiszámítjuk a 3D pozíciót, majd megnézzük, hogy a pontot a képekre visszavetítve mekkora eltérést kapunk a fa többi csúcsában tárolt pozíciókhoz képest. Amennyiben kis eltérést kapunk, az adott csomópontot helyes adatnak tekinthetjük. Végül az összes helyes adat felhasználásával egy pontosabb pozíciót számolhatunk háromszögeléssel.

6. Külső paraméterek meghatározása, PnP probléma

Az előző fejezetekben leírt módszerekkel létrehozhatunk egy adathalmazt, amelyben leíróvektorokhoz tartozó tárgypontok 3D koordinátáit tároljuk. Ennek segítségével tudjuk egy ismeretlen pozícióból készült kép külső paramétereit meghatározni. Ehhez nem kell mást tennünk, mint ugyanazt a kulcspontdetektáló és leírógeneráló algoritmust végigfuttatni a képen, mint amit az adathalmaz generálásakor használtunk, majd az így kapott kulcspontokat a leírók segítségével az adathalmazban található leírókhoz párosítani. Ezzel olyan megfeleltetésekhez jutottunk, amelyek a tesztképen lévő pontokhoz rendel 3D koordinátát.

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, amely csak 4 pontpárból számítja a paramétereket. Három pont kell ahhoz, hogy véges sok egyértelműen megoldást kapjunk (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.

7. Megvalósítás

Az ismertetett algoritmusok mind implementálva vannak az OpenCV függvénykönyvtárban. Ennek a Python nyelvhez készült burkolóját használtam.

Amennyiben az adathalmaz, vagyis a tanítóképeken megtalált kulcspontok, leírók és tárgypont-koordináták közti megfeleltetések rendelkezésre állnak, a tesztkép külső paramétereit egyszerű meghatározni. Ehhez az OpenCV beépített SolvePnPRansac függvényét használtam. Ez az algoritmus a PnP probléma megoldását számolja ki, és mivel RANSAC módszert alkalmaz, ezért a nagy arányú hibás adatra is robusztusan működik.

A megvalósítás legfőbb kérdése az adathalmaz létrehozásának miként je. Az algoritmus főbb lépései a következők:

1. Tanítóképek készítése
2. A tanítóképek külső paramétereinek meghatározása
3. A képeken kulcspontok keresése, leírók generálása
4. A leírók párosítása képpáronként, szűrés

5. A párosítások kiegészítése
6. A 3D pozíciók meghatározása

A következő alfejezetekben az egyes lépésekhez használt módszereket mutatom be.

7.1. Tanítóképek készítése

A fejlesztés során 5 tanítóképet használtam. Mindegyiken egy távirányító volt látható, alatta pedig egy kalibráláshoz használt marker. A távirányító azért volt jó választás, mert kulcspontokban gazdag, sok felismerhető jellemző van rajta. Egy ilyen képet láthatunk alább:

7.2. Külső paraméterek meghatározása

A kamera helyzetének meghatározása a képen látható marker segítségével történik. Jelenleg ez teljesen manuális.

7.3. Kulcspontkeresés, leírógenerálás

A kulcspontkereséshez SURF algoritmust használok. Ez gyorsabb a SIFT algoritmusnál, kisebb leírót generál, és valamivel robusztusabb bizonyos esetekben.

7.4. Leíró-párosítás, szűrés

A leíró-párosításnál többféle módszert is kipróbáltam.

7.4.1. Párosítás epipoláris egyenesekkel

Ennél a módszernél a párosítás során az adott kulcspont párját csak a másik képen behúzott epipoláris egyenes mentén keressük. a keresés brute force módszerrel történik. A párosítást a másik irányba is lefuttatjuk, és kereszt-ellenőrzéses szűrésnek vetjük alá.

7.4.2. Párosítás epipoláris egyenesekkel több kulcsponthoz

Ez a módszer annyiban különbözik az előzőtől, hogy nem csak a legjobb párosítást tartjuk meg, hanem a legjobb n darabot. A tesztelés során $n = 4$-et használtam.

7.4.3. Párosítás epipoláris egyenesekkel és homográfiás szűréssel

A módszer ugyanaz, mint a 7.4.1 részben ismertetett, de a végén lefuttatunk egy homográfia-alapú szűrést.

7.5. A párosítások kiegészítése

A gráf kiegészítéséhez az (5.1.1) részben ismertetett módszert használtam csúcs-hármasokra. Ezt az algoritmust kétszer futtattam egymás után.

7.6. 3D koordináták meghatározása

7.6.1. Klikkalapú rekonstrukció

Ennél a módszernél a párosítási gráfban megtalált klikkeket (teljes részgráfokat) veszem figyelembe. Feltételezem, hogy a klikkek minden éle helyes párosítás, és így a klikk csúcsaiból számítom a 3D pozíciót többnézetes háromszögeléssel. Megadható egy-egy küszöbérték a visszavetítési hiba átlagos és maximális értékére, ha ezt túllépi a hiba, a klikket eldobjuk, és nem számítunk hozzá 3D pozíciót. Az algoritmust 3-as klikkekre futtattam.

7.6.2. Egyszerű háromszögelés

Egy pontpárnak a koordinátáit a két képi pont alapján történő háromszögeléssel határozom meg. Ekkor nem történik vizsgálat a pontpárosítás helyességére nézve.

7.6.3. RANSAC háromszögelés

Ennél a módszernél a 3-as klikkek minden éléhez határozunk meg pozíciót. Egy-egy él pozíciója a következőképp kerül kiszámításra: Kiszámítjuk az egyszerű háromszögeléssel 7.6.2-ben kapott 3D pozíciót. Ezek után vesszük azokat a csúcsokat, amelyek az éppen számított él egyik csúcsához kapcsolódnak. Ezek között helyes adatnak tekintjük azokat, amelyek képekre visszavetítve a 3D pozíciót kis hibát kapunk (a küszöbérték meghatározható), majd ezekre a helyes adatokra többnézetes háromszögeléssel kiszámoljuk a pontos pozíciót. Amennyiben nincs elég helyes adat, a pozíciót eldobjuk, hiszen valószínűleg az eredeti párosítás sem volt helyes.

7.7. A kész algoritmus-láncok

Az előző alfejezetekben ismertetett módszerek különböző kombinációját teszteltem. Az alábbi változatok kerültek sorra. (Az egyes lépéseknél használt módszerre az alfejezet-számával hivatkozok)

1. 7.4.1, 7.5, 7.6.1
2. 7.4.2, 7.6.1
3. 7.4.1, 7.6.3
4. Az 1. és a 3. verzió adatainak uniója
5. Az 1., 2. és 3. verzió adatainak uniója
6. 7.4.3, 7.6.2
7. 7.4.3, 7.6.3

8. Eredmények

Az algoritmust 4 tesztképen vizsgáltam. Minden képhez adottak voltak a pontos külső paraméterek markeres kalibráció alapján. Ezzel hasonlítottam össze az adathalmaz segítségével meghatározott külső paramétereket.

Az eredményeket az alábbi táblázatokban foglaltam össze. A sorszám a fent leírt algoritmus-kombináció sorszáma. Mind a négy képen kiszámoltam a kamera valódi és adathalmaz alapján becsült pozíciójának különbségét. A ”min”, ”max”, ”avg” sorokban rendre a hiba minimumát, maximumát és átlagos értékét láthatjuk. Az X, Y, Z és LEN oszlopokban az egyes irányokban tapasztalt eltérést, illetve a hibavektor euklideszi hosszát láthatjuk.

	1.				2.			
	X	Y	Z	LEN	X	Y	Z	LEN
min	0,16	0,34	1,91	2,23	0,61	0,22	17,29	17,53
max	2,22	3,33	19,33	19,47	2,58	2,82	21,46	21,59
avg	1,12	1,18	7,84	8,24	1,17	1,54	18,83	18,98

	3.				4.			
	X	Y	Z	LEN	X	Y	Z	LEN
min	0,23	0,07	0,03	0,24	0,22	0,12	0,14	0,50
max	3,37	0,47	1,26	3,63	0,83	0,88	0,99	1,28
avg	1,17	0,23	0,41	1,28	0,52	0,48	0,50	0,96

	5.				6.			
	X	Y	Z	LEN	X	Y	Z	LEN
min	0,31	1,13	0,68	2,10	0,01	0,01	0,15	0,15
max	2,51	3,24	17,82	18,28	0,96	1,21	8,01	8,13
avg	1,23	2,11	9,09	9,76	0,50	0,57	2,45	2,73

	7.			
	X	Y	Z	LEN
min	0,04	0,01	0,02	0,05
max	0,60	2,15	5,02	5,47
avg	0,36	0,65	1,46	1,75

1. táblázat. A különböző algoritmusok helymeghatározásának hibaértékei. A mennyiségek centiméterben értendők.

Megfigyelhető, hogy a legtöbb algoritmus a Z irányban (ez a kamerából kifele mutató irány) a legpontatlanabb. Ez azért van, mert a helytelen párosítások a perspektív érzékenység miatt ebben az irányban sokkal nagyobb hibát okoznak, mint a másik két irányba.

Az is megfigyelhető, hogy a 2. módszeren kívül mindegyik algoritmusnál a legkisebb hiba kellően alacsony, ám a maximális hiba nem elfogadható.

Látható, hogy a maximális hiba nagyon nagy (20 cm-es nagyságrend) az 1. 2. és 5. algoritmusnál ebből arra a következtetésre jutottam, hogy a 7.5 módszer nem alkalmas a probléma megoldására, ugyanis ez a módszer közös ezekben az algoritmusokban.

Látható, hogy a 7.4.3 módszer sokkal jobban teljesített, mint más szűrési módszerek.

Az is észrevehető, hogy a 3. algoritmus is jól teljesített, ez valószínűleg a 7.6.3 módszernek köszönhető. A módszer robusztusságát támasztja alá, hogy a 6. és 7. algoritmusok közül a 7. adta a jobb eredményeket, ez az algoritmus is ezt a módszert használja.

9. Értékelés, továbbfejlesztési lehetőségek

Az algoritmusok kellően jól teljesítettek, különösképpen a RANSAC elven működők, és a homográfia-alapú szűrést használók. Ehhez valószínűleg az is hozzájárult, hogy a megfigyelt tárgy kellően részletgazdag. Egy egyszínű tárggyal nem működtek volna ilyen jól az algoritmusok.

Továbbfejlesztési lehetőség, hogy több hasonló nézőpontból készült képen nagyobb adathalmazt hozzunk létre. Így sokkal több párosítást találhatunk a képek között, hiszen kisebb a projektív torzítás, ez alapján pedig nagyobb adathalmazból több helyes adatot tudunk kinyerni, így pontosabban tudjuk a pozíciót is meghatározni.

10. Forrásjegyzék

1. Lowe, D. G., “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004

2. Bay, H. and Tuytelaars, T. and Van Gool, L. “SURF: Speeded Up Robust Features”, 9th European Conference on Computer Vision, 2006

3. Marius Muja, David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, 2009

4. Emami, S. and Levgen, K. (2012). Mastering OpenCV with Practical Computer Vision Projects

5. Martin A. Fischler & Robert C. Bolles (1981). ”Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”

6. Gao, Xiao-Shan; Hou, Xiao-Rong; Tang, Jianliang; Cheng, Hang-Fei (2003). ”Complete Solution Classification for the Perspective-Three-Point Problem”