Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Irányítástechnika és Informatikai Tanszék

Sárosi Márk

Objektumkövető algoritmus fejlesztése kiterjesztett valóság rendszerekhez

KONZULENS

Szemenyei Márton (IIT)

BUDAPEST, 2016

DIPLOMATERVEZÉSI FELADAT

Objektumkövető algoritmus fejlesztése kiterjesztett valóság rendszerekhez

Az utóbbi évtized során számos kutatási és fejlesztési projekt kapcsolódott alternatív felhasználói felületek kifejlesztésére. Ezek közül különös fontossággal bírnak a virtuális- és kiterjesztett valóságot alkalmazó megoldások, amelyek lehetővé teszik különböző virtuális objektumokkal történő interakciót.

Az interakció megvalósításához azonban minden esetben szükség van valamilyen beviteli módszerre (gesztus, kesztyű, mutatóeszköz, stb), ami az esetek túlnyomó többségében az alkalmazás során használt különféle valós tárgyak egyidejű követését igényli. Számos esetben a követendő tárgyak előre ismeretlenek, így a követést megelőzően szükséges azokról egy a követéshez használt modell megalkotása.

A diplomatervezés során a hallgató feladata egy olyan algoritmus fejlesztése és értékelése, amely képes tetszőleges valós tárgyak esetén a követéshez szükséges információk kinyerésére, és ezek alapján a követés elvégzésére.

A hallgató feladatának a következőkre kell kiterjednie:

- Tanulmányozza át a téma releváns szakirodalmát. Vizsgálja meg, hogy más műhelyek milyen megoldásokat alkalmaznak.
- Készítsen rendszertervet egy megoldásra, amely alkalmas valós tárgyak követési modellek megalkotására, valamint egyszerre több objektum követésére.
- Készítse el a rendszert, valamint egy demonstráló alkalmazást, amely a követés eredményét vizualizálja.
- Tesztelje az algoritmust pontosság, robusztusság és valósidejűség szempontjából.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Sárosi Márk**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

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 hitelesített 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.

Kelt: Budapest, 2016. 05. 29.

.....
Sárosi Márk

Összefoglaló

A mai modern világban a képfeldolgozás által használt technológiák a mindennapjaink részévé váltak, hiszen az objektumkövetés és az alakzat felismerés már a gépjárművekben található parkolást figyelő vagy végző rendszerek is alkalmazzák és akkor még meg sem említettem a haditechnikában vagy az űrtechnológiában használt követő és felismerő rendszereket.

A diplomaterv témám pozícióbecslést pontosabban objektumkövetést alapul vevő virtuális- és kiterjesztett valóságot alkalmazó megoldást alkalmazó program elkészítése.

1. Bevezetés

Témaválasztásomat meghatározta, hogy az egyetemi tanulmányaim során betekintést nyerhettem a képfeldolgozás témakörébe és az én megítélésem szerint nagyon nagy kihívásokat tartalmaz még ez a tématerület, valamint a véleményem szerint rendkívül hasznos eljárásokat lehet ezzel a gépi látás témakörrel létrehozni, gondol itt az orvostechnológiában elterjedt képalkotó rendszerekre, de akár az autóiparban megvalósított funkciókra.

1.1 Fejlesztőkörnyezet

A Visual Studio egy olyan fejlesztőkörnyezet, melyet a Microsoft támogat. Ez a fejlesztőkörnyezet több programozási nyelvben történő fejlesztést is lehetővé tesz, a nyelvek száma az idő előrehaladtával egyre csak bővül. Jelenleg az F#,C, C++, C# és a Visual Basic programozási nyelveket valamint a XML-t támogatja. [2]

1.2 C++ nyelv

A C++ egy általános célú, magas szintű programozási nyelv. Támogatja a procedurális, az objektumorientált és a generikus programozást, valamint az adatabsztrakciót. Napjainkban szinte minden operációs rendszer alá létezik C++ fordító. A nyelv a C programozási nyelv hatékonyságának megőrzése mellett törekszik a könnyebben megírható, karbantartható és újrahasznosítható kód írására, ez azonban sok kompromisszummal jár, erre utal, hogy általánosan elterjedt a mid-level minősítése is, bár szigorú értelemben véve egyértelműen magas szintű.[3]

1.3 OpenCV

Az OpenCV pontosabban kifejteve az Open Source Computer Vision egy programozási funkciókat megvalósító könyvtár, melynek célja elsősorban a valósidejű számítógépes látás támogatása. Az OpenCV C++ nyelvben íródott és a fejlesztése a mai napig is ebben a nyelvben történik, habár több különböző

nyelvben - például Python, Java, C# - is lehet használni az OpenCV által nyújtott szolgáltatásokat. Ebben a könyvtárban nagyon sok a képfeldolgozás során hasznos funkció található, jelen munkám során a videó kezelő, kép kezelő illetve a matematikai funkcióit használtam legtöbbet.[4]

1.4 Visual Struvcture from Motion System

A rövidebb nevén VSFM egy grafikus felhasználói interfésszel rendelkező háromdimenziós rekonstrukcióra alkalmas program. A program futási gyorsaságára nem lehet panasz, mivel a gépek többmagos tulajdonságát kihasználja, valamint a képjellemzők keresését és a párosításokat párhuzamosan végzi a több mag rendelkezésre állása által. [5]

1.5 Scale-invariant feature transform

A legtöbbször csak SIFT nevezett algoritmus a gépi látás és a képfeldolgozás során gyakran használt képleíró és képjellemző detektálásra alkalmas algoritmus. [6]

2. Megvalósítás

1.6 Tesztobjektum kiválasztása

Az első lépésben egy tesztobjektumot választottam ki, melyről videót készítettem, hogy a későbbiekben meg tudjam majd alkotni a követéshez szükséges modellt. A tárgy kiválasztása során figyeltem arra, hogy a tárgy ne legyen túlságosan szimmetrikus,- mint például egy focilabda -, illetve, hogy kellően megfelelő mintázatokkal rendelkezzen. A jellegzetes mintázatok, valamint az, hogy egy kevésbé szimmetrikus tárgyat választottam, a tárgyról kapott képjellemzők számát növelte meg, ami pedig a modell elkészítését könnyítette meg.

A választásom így egy olyan téglalap alapú dobodra esett, amit körberagasztottam csomagoló papírral. Hozzáteszem, hogy a tárgyat tartó alap kiválasztása során érdemes volt, egy olyan alapot - jelen esetben asztalt - kiválasztani, melynek színe és mintázata nagyban eltér az adott tárgyétól, mivel így a képjellemzők kinyerése során illetve a modell rekonstrukciónál csak minimálisan zavarnak be az alaphoz tartozó képjellemzők.

1. ábra: Jellegzetes mintázatokkal rendelkező tesztobjektum

1.7 Videó illetve kép készítés a modellalkotáshoz

A megfelelő tárgy kiválasztása után több képet is készítettem a tárgy minden oldaláról, mely fényképeket betöltöttem a Visual Structure from Motion System nevű programban, továbbiakban csak VSFM-ként fogok rá hivatkozni. Ez a program többek között a betöltött képekről képjellemzőket készít, illetve a képek közti párosításokat is elvégzi, továbbá rekonstruálja a meglévő képekből és adatokból az objektumot. Mindezeken túl az objektumhoz tartozó adatokat kimenti egy általunk megadott helyre.

A képek közti párosítás művelet alatt azt kell érteni, hogy a különböző képeken lévő azonos képjellemzőket a program felismeri és így össze tudja párosítani azokat. Olyan mintha, egy asztalról több képet készítünk és az asztal sarka jellegzetes pont, tehát a képeken a sarkokról kapott képjellemző adatokat össze tudja párosítani a rendszer, megtudja határozni, hogy melyik képeken hol van a sarka az asztalnak.

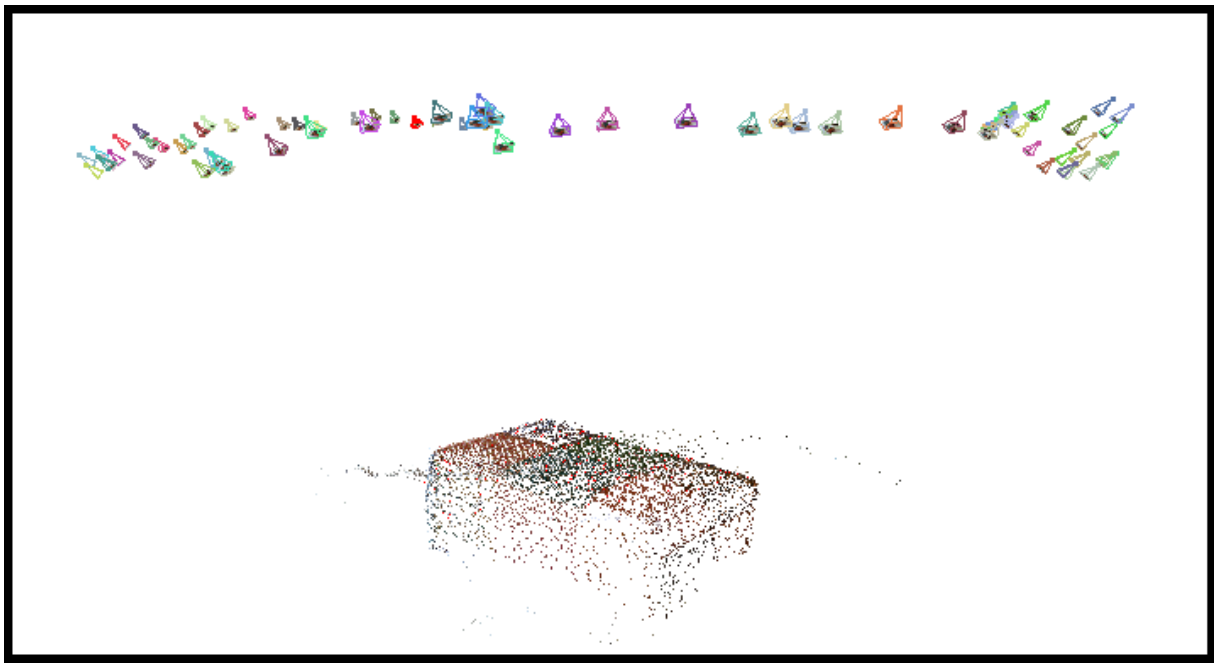
1.8 Háromdimenziós rekonstrukció

A VSFM-be betöltött képeken lévő tárgyról a program egy pontfelhőt készít, ami megfelelő ellenőrzési pont a felhasználó számára, hogy megvizsgálja, hogy az általa kiválasztott tárgy megfelelő-e a követésre, valamint, hogy kellő számú képpel rendelkezik-e a program a pontfelhő létrehozásához. Az első próbálkozásaim során a program által készített pontfelhő nem hasonlított az általam kiválasztott tárgyra, mivel a programba betöltött képek száma csupán tíz és húsz között volt.

Ezek után arra a következtetésre jutottam, hogy a betöltött képek számát meg kell, hogy növeljem, azaz több kép készítésére volt szükség. Azonban az ötven vagy száz kép elkészítése helyett, inkább egy 360 fokos videót készítettem a tárgyról. A videóból pedig a Free Video to JPG Converter nevű ingyenes program segítségével képeket mentettem le. A programban egyszerűen meg lehetett adni, hogy mely videóból, hány darab képet szeretnénk kinyerni.

A megfelelő pontfelhő megalkotása érdekében a videóból ötven, hetvenöt illetve száz képet nyertem ki. A tárgyról alkotott képek száma már elégségesnek bizonyult egy megfelelő pontfelhő létrehozásához, azonban a kellő számú képek segítségével létrehozott pontfelhő mégsem hasonlított kellőképpen az általam kiválasztott objektumra.

Az alkalmas pontfelhő létrehozását, úgy értem el, hogy az objektumtól teljesen eltérő alapra helyeztem el a tárgyat, és így készítettem el a videót, amiből a képeket kinyertem. Megjegyezném még, hogy a tárgyról készített videót érdemes nappal kellően sok fény mellett elkészíteni, más esetekben a program nem adja vissza kellően a várt adatokat, illetve kevesebb képjellemzőt talál meg.



2. ábra: VSFM által készített rekonstrukció

A rekonstrukciós képen látható pontfelhő kellőképpen hasonlít az általam kiválasztott tárgyra. A háttérben található pontok az alaphoz tartozó képjellemzők, melyeket egy később bemutatott módszerrel kiszűrtem, mivel ezek a képjellemzők nem az általam kiválasztott tárgyhoz tartoznak.

A pontok kiszűrésére pedig azért volt szükség, hogy a későbbiekben ne okozzanak gondot a számítások során. A pontfelhőn kívül az adott szögből készített kameráknak a pozíciója is látszik a képen.

1.9 A VSFM program által szolgáltatott információk

A VSFM programból kinyert adatok tartalma és formátuma igen fontos, mivel a pozícióbecsléshez szükséges információkat ezekből a fájlokból tudjuk kinyerni. A számításokhoz három fontos fájltípus szükséges. Minden képhez tartozik egy .sift kiterjesztésű fájl, melyben az adott képen található képjellemzők vannak eltárolva.

Ennél a fájltípusnál szükség volt kiterjesztés konvertálást csinálni, pontosabban az ilyen típusú fájlokat beolvastam mint .sift fájl majd kiírattam őket szöveges formátumban. Így már számomra is látható volt, hogy a képekhez tartozó képjellemzőknek az adatait milyen formában menti el a program, megjegyzem a VSFM dokumentációjában is leírták, hogy milyen információkat milyen sorrendben ment el a program, de én úgy gondoltam, hogy az a biztos, hogyha én is meggyőződöm a saját szememmel a kiírt adatok formátumáról.

A szöveges fájlokban elsőként a képhez tartozó képjellemzők száma volt definiálva, ezután következett egy pozíciót meghatározó mátrix, valamint egy képjellemzőt leíró mátrix, illetve minden .sift fájl végén egy lezáró integer szám áll.

Következő lényeges fájl a párosításokat tartalmazó adatsor, melyet a VSFM program automatikusan szöveges fájl típusként ment el. Ebben az adatállományban párosával szerepelnek a képek, utánuk pedig a két képen megtalálható azonos képjellemzők száma, valamint a képjellemzők sorszáma és koordinátája.

```
1 C:\Users\Mark\Desktop\Diplomaterv\FromStudio\image10.jpg
2 C:\Users\Mark\Desktop\Diplomaterv\FromStudio\image11.jpg
159
0 1071.48 460.908 1 1009.84 423.475
7 569.636 362.321 5 503.492 384.755
4 964.78 124.118 6 891.088 96.4897
9 805.654 181.758 8 721.225 167.109
15 309.398 391.693 14 229.6 442.941
17 478.754 438.971 16 420.616 475.697
22 289.27 564.375 19 217.236 622.19
26 574.305 486.798 25 531.292 519.308
31 766.578 578.002 26 745.35 595.738
29 702.67 543.144 27 674.46 565.027
32 638.312 516.71 30 603.148 544.241
33 638.312 516.71 31 603.148 544.241
21 902.255 394.117 32 847.927 377.973
```

3. ábra: Párosítások fájl tartalma

Végül a harmadik fontos fájl típusban, melynek .nvm a kiterjesztése találhatók azok a paraméterek melyekből a későbbiekben össze lehet állítani a különböző mátrixokat, melyek a pozícióbecsléshez szükségesek.

1.10 Program készítés a pozícióbecsléshez

Miután a VSFM program segítségével megalkottam az alkalmas pontfelhőt illetve lementettem a modellalkotáshoz szükséges adatokat, elkezdtem elkészíteni Visual Stúdió fejlesztőkörnyezetben egy programot, amely első lépésben elvégzi az eddig bemutatott lépéseket. A programozás során megvalósított lépéseket nagyban segítette a képfeldolgozás során gyakran használatos OpenCV.

A programban lévő függvények sorrendje megfelel a fentebb ismertetett lépésekkel. A programban a `numberOfImage` statikus változó értéke határozza meg, hogy az beolvasott videóból hány darab képet mentsen ki a program. Jelenleg ez a szám fixen 60, mivel ebben az esetben a tárgyról 6 fokonként áll rendelkezésemre kép. A kimentett képek .jpeg formátumúak, valamint a képek neve egytől hatvanig vannak sorszámozva.

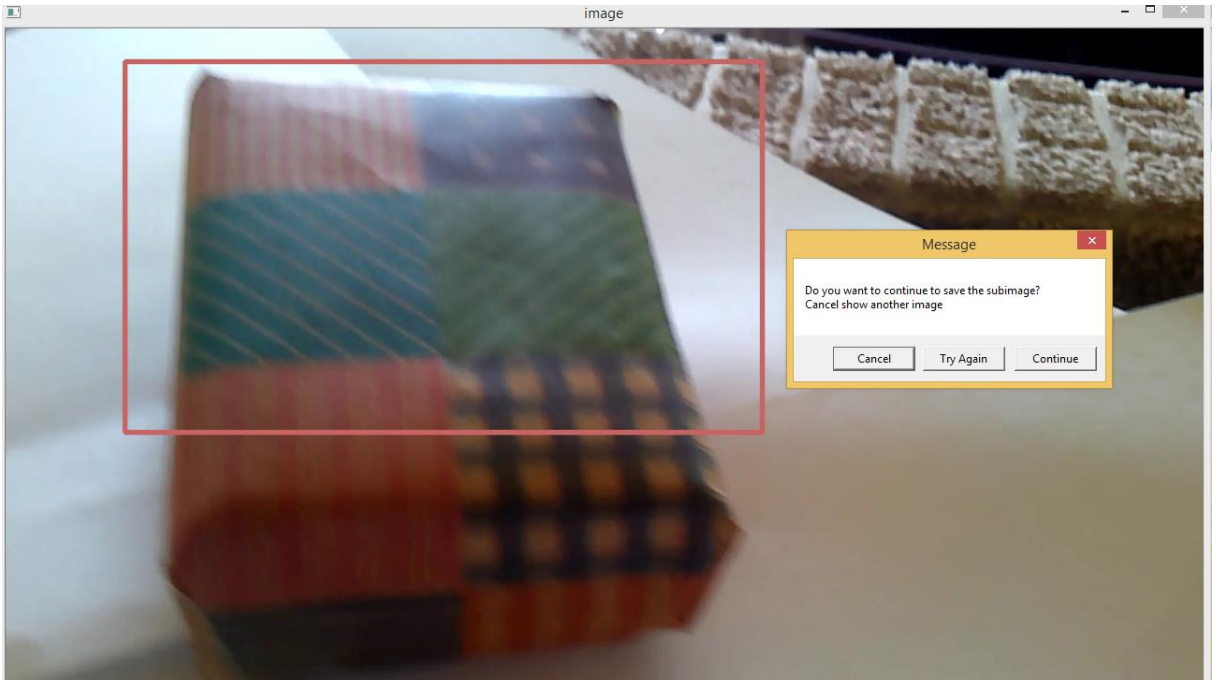
1.10.1 Fölösleges képjellemzők kiszűrése

A megfelelő számú képek kimentését követően a program megjelenít egy üzenetet, melyen értesíti a felhasználót, hogy négy képen be kell jelölnie az objektumot, melyet majd követni szeretne. Erre a lépésre azért van szükség, hogy csak azokkal a képjellemzőket vegye figyelembe a program, melyek a kiválasztott tárgyon vannak, a háttérben lévő képjellemzők nem fontosak illetve csak bezavarhatnak egyes esetekben, tehát ezzel a lépéssel szűrtem ki a fölösleges adatokat.

Az objektum kiválasztó függvény elkészítése során figyelembe kellett azt venni, hogy úgy kapjuk meg a legjobb képjellemzőket az adott tárgyról, hogyha a tárgy mind a négy oldaláról készítünk referenciaképet. A referencia képekhez tartozó képjellemzőkben már nem fognak szerepelni a háttérhez tartozó képjellemzők.

A felhasználónak több opciója is van egy tárgy kiválasztása során. Abban az esetben, hogyha a kép nem megfelelő, például homályos, a felhasználó kérhet egy másik képet, ami nagyvalószínűséggel már nem lesz homályos. Sőt ha esetleg a falhasználó hibásan jelölte be az objektumot a képen, akkor a program lehetőséget nyújt a kijelölés megismétlésére a Try Again gomb megnyomásával.

Viszont ha a felhasználó úgy ítéli meg, hogy megfelelően van bejelölve a tárgy akkor a Continue gomb megnyomásával haladhat a következő képre, ahol ugyanúgy ki kell jelölni a tárgyat. Az objektumkiválasztás lépését négyszer kell elvégeznie a felhasználónak mivel így fogunk majd rendelkezni mind a négy oldaláról egy-egy képpel. Azt a logikát, hogy a program lehetőleg pont a tárgy négy oldaláról mutasson képet a felhasználónak, egyszerű matematikai számításokkal végeztem el.



4. ábra: Rosszul kijelölt, homályos kép

Abban az esetben, amikor a felhasználó megfelelőnek ítélte meg az objektum kijelölését majd a Continue gombra kattintott, a program elmenti a képnek azon részét melyet a felhasználó kijelölt. Ezek lesznek a subimage-k, avagy referenciaképek, amik egytől négyig lettek számozva.

A tárgyat egy piros négyzettel lehet kijelölni, a piros négyzet létrehozásánál ügyelni kellett arra, hogy a felhasználó bármely sarokból megkezdhesse a kijelölést a képen, vagyis a koordináta számításoknál, a hibák és a futásbeli exception-ok elkerülése miatt, abszolút értékekkel kellett számolni.

Továbbá az elsőként kimentett referenciaképek még tartalmazták a kijelölő piros négyzetet is, ami ugye nem helyes, mivel az a része a képnek számunkra irreleváns. Egy kisebb módosítás után, melyben a kimentett képszélességből és magasságból levontam a vonal szélességét, már a piros keret nélküli referencia képeket mentette el a program.

Most már a program rendelkezik megfelelő képekkel valamint négy olyan referenciaképpel, amin az objektum szerepel nagyrészt. A VSFM rendelkezik

console parancsokkal, melyek segítségével könnyedén el lehet végezni azokat a lépéseket, amiket a felhasználói felületen is el lehet végezni, amely lépések végén a program kimentette a fent már említett képjellemző és párosításokat tartalmazó fájlokat.

1.10.2 Pozícióbecslés

A meglévő adatok már elég információt nyújtanak ahhoz, hogy elvégezhessem a pozícióbecslést. A .nvm kiterjesztésű fájlban minden képhez tartozik egy fókusztávolság, négy darab kvaternió koordináta, három kamera koordináta illetve egy torzítási együttható. A kvaternió egy háromdimenziós forgatást ír le négy darab számmal, ahol az x,y,z egy tengelyt ad meg, a w pedig a tengely körüli forgatásnak a szögét.

$$P = A\,[R\ t] \tag{1}$$

A (1) egyenletben a projekciós mátrix kiszámításának módját láthatjuk, miszerint az A kameramátrixot meg kell szorozni, az R rotációsmátrixxal és a t vektorral. A kameramátrix tartalmazza a belső paramétereket, míg a rotációs mátrixban pedig a külső paraméterek találhatóak. A projekciós mátrix kiszámítását el kell végezni minden egyes felhasznált képhez.

$$G = [P\ S] \tag{2}$$

$$G = U * \sum * V^T \tag{3}$$

A (2) egyenletből megkapjuk a G mátrixot, amit a projekciós mátrixból illetve a sift képjellemzők x,y koordinátáiból állítottam össze. A G mátrixot SVD felbontással több mátrixra bontottam fel.

Így V^T mátrix negyedik sorában szereplő koordináták lesznek a háromdimenziós keresett pozíciónk koordinátái, tehát ezzel a módszerrel meg tudtuk becsülni, hogy egy adott képen a tárgy a kamerához képes milyen

pozícióban van. Ezt a számítást is mind egyes képre el kell végezni. Miután már meg van az elmélet, már csak a megfelelő paraméterekkel kellett feltölteni a mátrixokat.

A kameramátrix egy felsőháromszög mátrix, melyben a fókusztávolság, principális pontok, amik a kép szélességének és magasságának a fele, illetve egy konstans 1-es szerepel. A pontosabb eredmény érdekében a fókusztávolság, az minden kameramátrixban megegyezik, mivel az összes fókusztávolság átlagát vettem alapul.

A t transzláció vektorba a kameraközéppont szerepel. A rotációs mátrixot pedig a kvaternió koordináták alapján számoltam ki. Míg az SVD felbontás elvégzésre az OpenCV-ben rendelkezésünkre áll a SVD::compute beépített függvény. [1]

Jelenleg ebben a stádiumban tart a munkám, pontos pozíció becslést még nem sikerült elérnem, mivel a tárolás és a mátrixok kezelése ilyen nagy mennyiségű kép esetén nem olyan egyszerű.

3. Tesztelés

A féléves munkám során a modell alkotása közben is igyekeztem tesztelni a már meglévő programot. Természetesen a tesztelés komolyabb része csak a program elkészülése után fog következni.

Azonban eddig is volt alkalmam tesztelni a lépéseket. A legjobb tesztelési lehetőséget a pontfelhő megalkotása adta. Az adott képekhez tartozó pontfelhőket ki tudtam értékelni, és meg tudtam határozni, hogy a tesztobjektum kiválasztása során a pontfelhő kevésbé helyes, hogyha a kiválasztott tárgy nagyon egyszínű, vagy ha kevés mintázat volt rajta, illetve hogyha nagyon szimmetrikus volt a tárgy.

Azt is meg tudtam állapítani, hogyha a tárgyal közel azonos színű alapot választok, és úgy készítem el a képeket vagy a videókat, akkor a pontfelhő tartalmazni fogja a háttérben található képjellemzőket. Viszont, hogyha a kiválasztott tárgyat egy fehér alapra helyezem, akkor elméletben a háttérről semmilyen vagy csak kevés képjellemzőt kap a VSFM program. Hiszen annál több információt kapunk a tárgyról minél pontosabb a pontfelhő.

A referenciaképek kiválasztása során is alkalmam volt tesztelni a rendszert, mivel több olyan alkalom is előfordult, hogy a program által megadott kép homályos volt. Ezt egyelőre úgy küszöböltem, ki, hogy a program egy másik képet fog felajánlani a felhasználónak, hogyha a felhasználó ezt igényli.

Következő tesztelési lehetőség a pozíció meghatározásánál lesz, mivel az adott mátrixokból és a matematikai műveletek segítségével, határozza meg a program a tárgy távolságát és én pedig le tudom majd mérni a kamera illetve a tárgy között lévő valós távolságot.

4. Továbbfejlesztési lehetőségek

Az eddig meglévő munkámban elsősorban a lehető legtöbb fölösleges adat kiszűrése lenne a cél. Vagyis jó lenne, hogyha az alap képjellemzőinek kiszűrését esetleg falhasználói beavatkozás nélkül meg lehetne valósítani, továbbá nem csak egyszínű alappal rendelkező videó készítés is egy reális továbbfejlesztési lehetőség.

A másik fontos fejlesztési lehetőség, hogy olyan programot készítsek, mely kevésbé érzékeny az objektum kinézetére, ezalatt azt értem, hogy nagyon szimmetrikus és kevés mintával rendelkező objektumok esetén is megfelelő legyen a rekonstruált modell illetve a kinyert adatok is kellőképpen valósak legyenek.

A követés során majd az objektumok kitakarását is valamilyen módon kezelnem kell majd, ez majd egy jövőbeli kihívás, de ez is mindenképpen egy fejlesztési lehetőség, valamint az ebből adódó egymás előtt lehaladó objektumok helyes követése is egy érdekes lehetőség a továbbfejlesztésre.

Mint a legtöbb program esetében jelen esetben is a gyorsaság és valósidejű működés biztosítása is egy megfelelő cél.

5. Irodalomjegyzék

[1] https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

[2] http://www.visualstudio.com/products/visual-studio-ultimate-with-MSDN-vs

[3]https://hu.wikipedia.org/wiki/C%2B%2B

[4] http://opencv.org/

[5] http://ccwu.me/vsfm/

[6] https://en.wikipedia.org/wiki/Scale-invariant_feature_transform