Structure from Motion from Two Views

1. fejezet

Algoritmus

A Structure from motion (SfM) folyamat segítségével 3D rekonstrukciót hajthatunk végre egy képpár segítségével.

1. Két kép közötti ritka ponthalmazok megfeleltetése (pontmegfeleltetés): az első kép sarkainak azonosítása a *detectMinEigenFeatures* függvénnyel, majd azok követése a második képre a *vision.PointTracker* segítségével.

2. Az esszenciális mátrix becslése *estimateEssentialMatrix* használatával.

3. Kamera elmozdulásának kiszámítása *estrelpose* függvénnyel.

4. Két kép közötti sűrű ponthalmazok megfeleltetése (pontmegfeleltetés): több pont kinyeréséhez újra kell detektálni a pontokat a *detectMinEigenFeatures* függvény segítségével a *'MinQuality'* opciót használva. Ezt követi a sűrű ponthalmaz követése a második képre a *vision.PointTracker* használatával.

5. Az illeszkedő pontok 3D helyzeteinek meghatározása a *triangulate* segítségével (háromszögelés).

2. fejezet

Kód magyarázata

2.1. Képpár betöltése

1. *fullfile(string1, string2, ...)* = az argumentumként kapott stringekből összeállít egy elérési útvonalat, pl.:

   ```
   path = fullfile('myfolder', 'mysubfolder')
   path = 'myfolder\mysubfolder\'
   ```

   toolboxdir(toolbox) = visszaadja az argumentumként kapott toolbox abszolút elérési útvonalát.

2. *imageDatastore(path)* = létrehoz egy ImageDatastore objektumot a kapott elérési útvonallal meghatározott képekből. Az ImageDatastore objektum segítségével egy mappában található összes képet össze lehet gyűjteni egy változóba (de alapból nem lesz az összes kép egyszerre betöltve).

3. *readimage(datastore, n)* = betölti az n. képet a megadott datastoreból.

4. *figure* = létrehoz egy új, üres ábra ablakot.

5. *imshowpair(image1, image2, 'montage')* = a meghatározott két képet egymás mellé helyezi a legutolsó ábrán.

6. *title('string')* = hozzáad egy címet a legutolsó ábrához.

2.2. A Camera Calibrator alkalmazás segítségével előre kiszámolt kamera paraméterek betöltése.

1. *load(file_ name.mat)* = betölti egy korábban elmentett workspace adatait a jelenlegi workspace-be. A workspace egy ideiglenes tároló amely a MATLAB elindítása óta létrehozott változókat tárolja. Alapértelmezetten a MATLAB ablak jobb oldalán látható. A workspace-t el lehet menteni, így a benne tárolt változókat később vissza lehet tölteni a MATLAB-ba.

2.3. Lencse által okozott torzítás eltávolítása.

1. *undistortImage(image, intrinsics)* = a második argumentumként megadott kamera paramétereket felhasználva eltűnteti a kamera lencséje által okozott torzítást a megadott képről.
   A kamera kalibrációja során kapott kamera paramétereket és a torzítási együtthatókat felhasználva kiszámítjuk a bemeneti kép minden pixelének eredeti pozícióját. Az egyes pixelek pozícióját az alábbi torzítások módosítják:

   - **Radiális torzítás** = kiváltó oka, hogy a lencse szélén áthaladó fény jobban törik, mint a lencse közepén környezetében áthaladó fény. Ez kiszámolható:

     $$x_d = x_u(1 + k_1 r^2 + k_2 r^4)$$
     $$y_d = y_u(1 + k_1 r^2 + k_2 r^4)$$

     (Ahol x_u, y_u = torzulásmentes koordináták; x_d, y_d = torzított koordináták; k_1, k_2 = radiális torzítási együtthatók; $r^2 = x_u^2 + y_u^2$)
   - **Tangenciális fordítás** = előfordul, ha a kameraszenzor és a lencse nem állnak tökéletesen párhuzamosan. Ez kiszámolható:

     $$x_d = 2p_1 x_u y_u + p_2(r^2 + 2x_u^2)$$
     $$y_d = 2p_2 x_u y_u + p_1(r^2 + 2y_u^2)$$

(Ahol x_u, y_u = torzulásmentes koordináták; x_d, y_d = torzított koordináták; p_1, p_2 = tangenciális torzítási együtthatók; $r^2 = x_u^2 + y_u^2$)

Az egyes pixelek korrigált helyének kiszámítása nem egész számú értékeket is előállít. Mivel a nem egész szám nem lehet pixel koordináta, ezért bilineáris interpolációt is végre kell hajtani. A bilineáris interpoláció során, a legközelebbi négy szomszédot felhasználva először lineáris interpolációt hajtunk végre az egyik irányba (pl. az x tengely mentén), majd pedig a másik irányba (az y tengely mentén):

$$out_P = I_1(1-\Delta X)(1-\Delta Y)+I_2(\Delta X)(1-\Delta Y)+I_3(1-\Delta X)(\Delta Y)+I_4(\Delta X)(\Delta Y)$$

(Ahol I_1, I_2, I_3, I_4 = a szomszédos négy koordináta intenzitása az eredeti, torzított képen; $\Delta X, \Delta Y$ = a nem egész értékű koordinátákkal rendelkező vizsgált pixel és a vizsgált pixelhez legközelebb eső, egész értékű koordinátákkal rendelkező szomszédai közötti távolság; out_P = végeredményként kapott pixel intenzitás)

A szomszédos pixelek efféle súlyzott átlagolásával, az interpoláció eredményeképp egy pixel intenzitás értéket kapunk, amely a legközelebbi egész érték koordinátával rendelkező pixel intenzitása lesz.
Az előállított, torzítatlan képen néhány pixel (leginkább a kép szélein) nem rendelkezik megfelelő pixel párral az eredeti, torzított képről (ezek azok a területek, ahol az eredeti képből nincs információ). Ezek a pixelek alapértelmezetten 0 értéket kapnak (feketék lesznek).

2.4. Pontmegfeleltetés a képek között.

1. *detectMinEigenFeatures(grayImage, MinQuality=0.1)* = a minimum sajátérték algoritmust (Shi & Tomasi, Minimum Eigenvalue Algorithm) használva keresi meg a kép sarokpontjait (a sarokpont jelen esetben olyan pixeleket jelentenek, amelyek éles változást mutatnak a környező pixelekhez képest). Szürkeárnyalatos képet vár argumentumként, ezért

a képet még előtte az *im2gray* függvénnyel szürkeárnyalatosra változtatjuk. A *MinQuality* argumentum a detektált sarokpontok minőségét határozza meg. Az értékének [0, 1] tartományból választhatunk. Magasabb érték, jobb minőségű, viszont kevesebb sarokpontot is eredményez.

3. fejezet

Forrás

- https://www.mathworks.com/help/vision/ug/structure-from-motion-from-two-views.html
- https://www.mathworks.com/help/vision/ref/undistortimage.html?s_tid=doc_ta
- https://www.mathworks.com/help/visionhdl/ug/image-undistort.html
- https://e-learning.ujs.sk/pluginfile.php/23441/mod_resource/content/1/01-ProjektivKamera.pdf
- https://www.mathworks.com/help/vision/ref/detectmineigenfeatures.html