Gépilátás beadandó

2020/2021/2

Feladat: Panorámakép készítése

Készítette: Smauzer Richárd

Neptunkód:CQWANZ

Bevezetés

Panorámakészítő programot választottam féléves feladatnak. Ez a program több kép darabot képes összeilleszteni a közös sarokpontok(jellemzőpontok) alapján.

Képes képek darabolására(ha a mappában 1db kép van) majd az összevágott képet a sarokpontok által összeilleszteni, és a bemeneti kép és a kimeneti kép(panorámakép) összehasonlítására(Template Matching).

Az összevágott képeket egy (Cutted_Images) mappába tárolja és ebben a mappában lévő képeket használja fel a panorámakép készítéséhez, majd át kell esnie a vizsgálaton amivel a Template Matching segítségével % pontosságban meg tudjuk adni hogy a legenerált kép mennyivel tér el a bemeneti képtől.

Elméleti háttér

```python
import cv2
import os
import shutil


mainFolder = 'Images'
myFolders = os.listdir(mainFolder)
print(myFolders)

for folder in myFolders:
    path= "C:/Users/smauz/Desktop/Panoramakep/Images/"+ folder +"/Cutted_images"
    if os.path.exists(path):
        shutil.rmtree(path)

    path = mainFolder + '/' + folder +'/Panorama.jpg'
    if os.path.exists(path):
        os.remove(path)
#-------------------------
```

Beimportálom a szükséges packageket, itt érdemes megemlítenem, hogy a program ujrafuttatásának automatizálása érdekében, a feldarabolt képek mappáját (Cutted_images) és a kimeneti képet (Panorama.jpg) törölni kell, ez csak akkor fut le, ha megtalálja a mappát és a filet.

```python
    #-----------------------------

    #0 átmenettel
    #cropped1 = image[0: int(image.shape[0]), 0:int(image.shape[1]/2)]
    #cropped2 = image[0: int(image.shape[0]), int(image.shape[1]/2):int(image.shape[1])]

    #cv2.imshow("first",cropped1)
    #cv2.imshow("secound",cropped2)
    #cv2.waitKey(0)

    #-----------------------------

    #1/5 átmenet ezt az értéket a tesztek során növelni ill. csökkenteni fogjuk
    #képek 2 vágása
    cropped1 = image[0: int(image.shape[0]), 0:int(image.shape[1]/2+image.shape[1]/5)]
    cropped2 = image[0: int(image.shape[0]), int(image.shape[1]/2-image.shape[1]/5):int(image.shape[1])]

    #-------------------------
```

Ezen a képen látható az a programrészlet, ami alapján a képdarabokat elkészítjük. *cropped1* a kép baloldaláról haladok a feléig majd hozzáadom a kép 1/5-dét. Majd a *cropped2*-nél ugyan ez csak jobbról haladunk visszafelé és adjuk hozzá a kép 1/5-dét igy megkapjuk az átfedést. Átfedés segít a képek összeillesztésében hiszen az átfedésben lévő jellemzőpontok megtalálható mind 2 képen.

Matematikai háttér

Jellemzőpontok (sarokpontok keresése)

Jellemzőpontok segítségével össze tudjuk fűzni a képeket.

Feltételezzük, hogy a kamera az optikai tengelye körül forog, a transzformációs csoportokból a képek áteshetnek a speciális homográfiai csoportokhoz. A paramétereket egyes kameráknál a forgási vektor határozza meg $\theta = [\theta_1, \theta_2, \theta_3]$ és a fokális hosszát az *f* határozza meg. Ez megadja a homográfiát páronként $\tilde{u}_i = H_{ij}\tilde{u}_j$ ahol =>

$$=> H_{ij} = K_i R_i R_j^T K_j^{-1} \quad (1)$$

és $\tilde{u}_i, \tilde{u}_j$ a homográfiai képpontok ($\tilde{u}_i = s_i[u_i, 1]$, ahol az u_i egy 2 dimenziós képpont).

Meghatározzuk a 4 paraméteres camera modellt

$$K_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

és (exponenciális ábrázolással a forgatáshoz)

$$R_i = e^{[\theta_i]x}, [\theta_i]x = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix} \quad (3)$$

Ideális esetben képjellemzőket használ, amelyek változatlanok a transzformációs csoportok alatt.

Habár, ha kis változás van a kép pozíciójában

$$u_i = u_{i0} + \left.\frac{\delta u_i}{\delta u_j}\right|_{u_{i0}} \Delta u_j \quad (4)$$

vagy egyenértékűen

$$\tilde{u_i} = A_{ij}\tilde{u}_j \qquad ahol \quad A_i \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

van affin transzformáció, linearizálással kapjuk meg a homográfiát (u_{i0}). Ez azt jelenti, hogy minden kis képdarab átesik az affin transzformáción.

Ebben a szakaszban a célkitűzés az, hogy megtaláljuk az összes olyan sarokpontokat, amik megtalálhatóak mind a 2 képen(átfedés)

Előző feladatban az volt a cél, hogy a jellemzőpontokat megtaláljuk a képeken, amiből elég sok van.

RANSAC ALGORITMUS (Random minta konszenzus)

RANSAC algoritmust használjuk, hogy kiválassza azokat a homográfiai pontokat, amik azonosak a képek között.

Minimális random mintavételezést használ, hogy kiszámítsa a kép transzformációt. Tekintettel arra, hogy valószínűleg vannak közös jellemzőpontjai a kepék között, amit p_i jelölünk, a valószínűségé, hogy találunk korrekt transzformációt az n próbák után az

$$p(H\ is\ correct) = 1 - (1 - (p_i)^r)^n \qquad (6)$$

egyenlet segítségével nagy számú próbálkozások után nagyobb a valószínűségé, hogy megtaláljuk a korrekt homográfiai pontokat.

Valószínűségi modell a képillesztés ellenőrzéséhez

Adott képhez, az összes jellemzőpontját felhasználva az átfedés területén n_f és az *inliners*-ek száma n_i. Ez megmutatja, hogy korrek/inkorrekt a képek egyezése.

Ezt a bináris változó mutatja meg m ∈ {0,1}, jellemzőpont $f^{(i)} \in \{0,1\}$ inlier/ outlier értékeket összegezzük a Bernoulli függvénnyel, az összes inlier érték Binomiális.

$$p\left(f^{(1:n_f)}|m = 1\right) = B(n_i; n_f; n_1) \qquad (7)$$

ahol p1 az a valószínűlege, hogy a képegyezés korrekt

$$p\left(f^{(1:n_f)}|m = 0\right) = B(n_i; n_f; n_0) \qquad (8)$$

ahol p0 az a valószínűlege, hogy a képegyezés inkorrekt

Binomiális eloszlással számolva

$$B(x, n, p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x} \qquad (9)$$

$$p\left(f^{(1:n_f)}|m=1\right) = \frac{p\left(f^{(1:n_f)}|m=1\right)p(m=1)}{p\left(f^{(1:n_f)}\right)} \quad (10)$$

$$= \frac{1}{1+\frac{p\left(f^{(1:n_f)}|m=0\right)p(m=0)}{p\left(f^{(1:n_f)}|m=1\right)p(m=1)}} \quad (11)$$

Elfogadjuk a képegyezést, ha $p\left(f^{(1:n_f)}|m=1\right) > p_{min}$

$$\frac{B(n_i;n_f;p_1)p(m=1)}{B(n_i;n_f;p_0)p(m=0)} \begin{matrix} >(accept) \\ <(reject) \end{matrix} \frac{1}{\frac{1}{p_{min}}-1} \quad (12)$$

Miután páros egyezések megtörténtek a képek között, panorámaszekvenciákkal összekapcsoljuk a képeket.

Template matching (Mintakeresés), Kereszt-korreláció

Erre azért van szükségünk mivel, ha egy képet feldarabolunk és össze akarjuk illeszteni (jellemzőpontok összeillesztésével), akkor a bemeneti képet össze kell hasonlítanunk a kimeneti képpel, hogy mekkora az eltérés a legenerált képe és a valós kép között.

Erre a feladatra a legegyszerűbb mód a kereszt-korreláció. Legyen a I a kimeneti kép és a W a bemeneti kép, a feladatban az összevágni kívánt kép a bemenetikép, és a kimeneti kép pedig az, amit a bemeneti feldarabolt képek össze illesztése után kapunk. A CC (x, y) szám megadja, hogy az I(x, y) pozícióra mennyire jól illeszkedik rá a keresett W bemeneti kép

$$CC(x,y) = \frac{\sum(W(x',y')I(x+x',y+y'))}{\sqrt{\sum(W(x',y')^2 \sum I(x+x',y+y')^2)}}$$

Megvalósítás terve és kivitelezés

Megvalósítás terén szükségesnek érzem azt, hogy megemlítsem az eszközt amivel a féléves feladatomat elvégeztem. Először a Pycharm nevű alkalmazás próbálkoztam viszont a program nem volt felhasználóbarát mivel az új verzió ismételten meghiúsította a program működését. Ám emlékezve az órán használt alkalmazásra, aminek a neve **Thonny**. Megelégedve a könnyű kezelésével nekiláttam a féléves feladatom megvalósításához.

A megvalósítás első része a képdarabok elkészítése (telefonnal), majd külön mappákba (folderekbe) való elhelyezése. Mivel igyekeztem felhasználóbaráttá tenni a programot igy képes több panorámakép elkészítésére. Először is ugye szükségem volt az **opencv** és az **os** plugin segítségére. Ezeket beimportáltam a program elején. A mainfolder megadása után a program a mainfolderben található mappákat kiíratva belelép és a benne lévő képeket összefűzi a stitcher.stitch() segítségével majd az aktuális mappában legenerálja a végeredményt ha sikerül. Ehhez szükségem volt egy stitcher nevű változóra hogy behívjam a stitcher osztálynak a függvényét ezt a **cv2.Stitcher.create()** el valósítottam meg.

Továbbá a program képes arra is, hogy megvizsgálja a mappa tartalmát úgy hogy ha 1 képet talál akkor szétdarabolja és a képrészleteket elmenti egy (Cutted_images nevü) mappába, és az ebben lévő képeket igyekszik összeilleszteni ami függ az átfedések méretével. Majd a kimeneti (Panoráma) képet összeveti a bemeneti(szétvágott képpel) és % pontossággal megadja hogy mekkora az eltérés a 2 kép között

Tesztelés működő képekkel

Eredménye a tesztelésnek:

Jelentése az eredményen látható szimbólumoknak:

\+ -> összefűzés két kép között
= -> kimeneti végeredmény

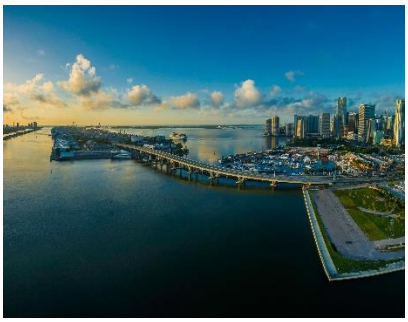
1 DB kép 2 vágása 2/5 átfedéssel


+

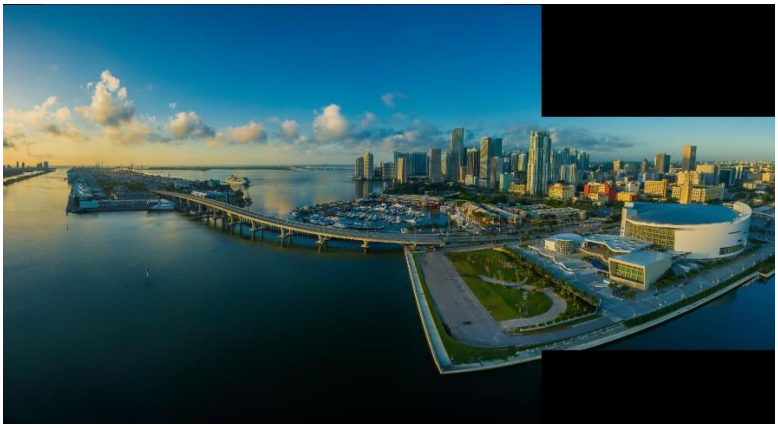
=

2 db 0° képből elkészített panorámakép nem látszik deformáció a közös pontoknál, jól látható és áttekinthető végeredmény. 98%-ban egyezik a kimeneti kép a bemeneti képpel

1) 0°-os szögben elkészített képek (internetről)


+

=



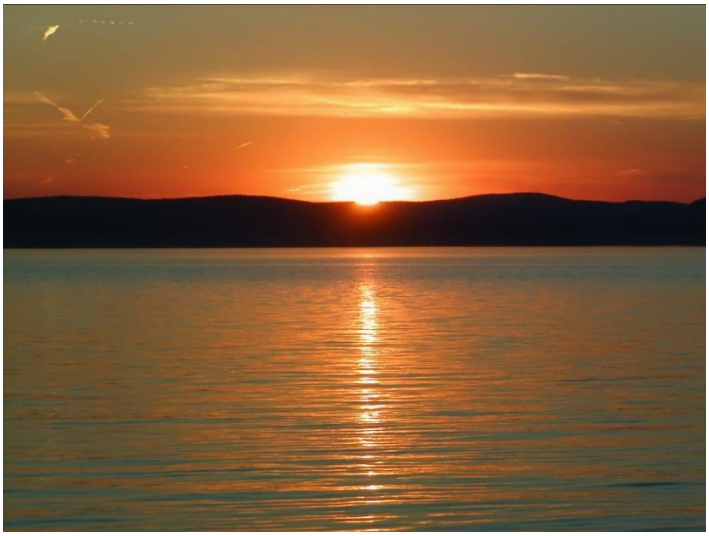
2 db 0° képből elkészített panorámakép itt már megfigyelhető a deformáció a képek szélénél és a közös pontok össze illesztve nem szépek. Valószínűleg azért, mert a kép összeillesztése közben el lett forgatva egy kicsit és igy ha összeilleszti a két elforgatott képet akkor deformálás jelenik meg a közös össze illesztett pontoknál

2) 2 db 35°-os szögben készített fekete-fehér képekkel(telefonnal készítve).


+ =

2 db fekete-fehér kép összeillesztve nem tapasztalható deformáció az összeillesztett képpontoknál, viszont itt már látszik, hogy a két kép 35° szögben készültek így a végső képen látható a kép szélénél, hogy ez miatt eldeformálódott.

3) 2/25-ös átfedéssel 2 vágott kép esetén



Template Matching *segítségével az eredeti képet összevetettem a kigenerált képpel és 99%-ban egyezik a kigenerált kép az eredetitől*

4) 2/15-ös átfedéssel 2 vágott kép esetén


+

=


97%-ban egyezik a kigenerált kép az eredetitől

Tesztelés zavaró tényezős képekkel, vagy több képdarabokkal történő tesztelés

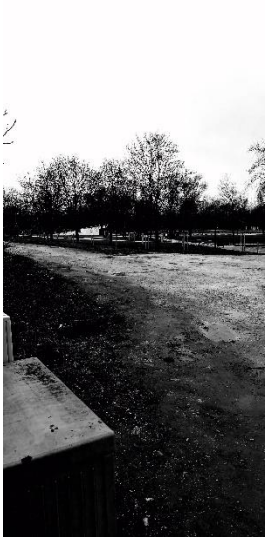
0 ráfedéssel

Ezen az oldalon fogom bemutatni azokat a teszteket, amelyek tartalmaznak sikertelen teszteket, és olyan több képdarabokkal rendelkező képek amik más formában jelentek meg (pl álló képek összefűzésénél).

<u>Jelentése az eredményen látható szimbólumoknak:</u>

+ -> összefűzés két kép között
= -> kimeneti végeredmény

1) 2 db 50°-os szögben készített normál és egy fekete-fehér képpel.(telefonnal készült képek)

 +  =

```
Images/4
Total no of images detected 2
Panorama Generation Unsuccessful
```

Az eredmény várható volt mivel 1. nem található közös pont. 2. nem, hogy nem található közös pont még a színek sem egyeznek. ***Javaslat:*** *A kép készítésekor figyelni kell hogy legyenek közös pontjai a képeknek és a stílus kiválasztása minden képen azonosnak kell lennie.*

2) 5db 25°-os szögben készített képek.

Látható hogy több képet is képes összeilleszteni pláne úgy hogy az első és az utolsó képdarab ismételten nem talalhato közös pontjai igy nem illeszkedik az első képpel, viszont a munka 75% képes volt legenerálni.

3) 2db kép, amin egy mozgó objektumot szeretnénk összefűzni.


+

=

```
Images/7
Total no of images detected 2
Panorama Generation Unsuccessful
```

Ezeken a képeken egy idős hölgyet kell nézni, akinek a helye mind a 2 képen változik, tehát a két kép készítése során mozgott igy nem talált közös pontot a két képen, aminek a segítségével össze tudná illeszteni a képeket.

4) 0 átfedéssel készült kép, 1 db képet kettévágunk a program segítségével.


+

=

```
Total no of images detected 2
C:/Users/smauz/Desktop/Panoramakep/Images/1/Cutted_images/test1.jpg
C:/Users/smauz/Desktop/Panoramakep/Images/1/Cutted_images/test2.jpg
Panorama Generation Unsuccessful
```

Látható, hogy 0 átfedéssel nem képes a közös pontok megtalálására és igy nem tud képet összefűzni

Felhasználói leírás

A program működéséhez szükséges az Opencv-python 4.4.0.46 , és az python 3.4 es verziójára.

Először is szükségünk van egy olyan filera amiben külön folderekben vannak elrendezve az összefűzendő képek. (Ezeket úgy kell elkészíteni, hogy az összefűzendő képeknek közös pontjaik legyenek). Ezeket célszerű számokkal elnevezni az átláthatóság szempontjából. Amint készen vagyunk a képek elhelyezésével a folderekben, akkor a program lefutása után látnunk kell egy result nevű képet, amin az összefőzött kép(panorámakép) látható. Ha ez megvan akkor további teendők nincs. A programnak működnie kell.

A mappába ha szeretnénk 1 darab képet beletenni és megnézni hogy a lefutás után milyen panorámaképet kapunk akkor megtehetjük, és a végén % pontossággal megkapjuk hogy mennyi az eltérés a 2 kép között.

Pl:.

Felhasznált irodalom

https://docs.python.org/3/library/os.html - OS

https://www.geeksforgeeks.org/append-extend-python/ - append

https://docs.opencv.org/master/d8/d19/tutorial_stitcher.html - stitcher

http://matthewalunbrown.com/papers/ijcv2007.pdf -Automatic Panoramic Image Stitching

https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/