Szkenner alkalmazás - Fejlesztői Dokumentáció

Megoldandó probléma: Egy képen található papír alapú dokumentum felismerése, szkennelése és javítása.

A probléma megoldásához a programban következő sorrendben fogjuk a képeket "manipulálni".:

-Szürkeárnyalat konverzió Canny-detektorhoz

-Gauss szűrő alkalmazása Canny-detektorhoz

-Canny Detektor alkalmazása

-Vesszük a megtalált kontúrélek végpontjait (4 pont a mi esetünkben)

-Sorrendben eltároljuk őket egy tömbben

-Kiválasztjuk a kívánt dokumentum dimenziót (egyenlő oldalú dokumentum, A4 álló, A4 fekvő)

-A kiválasztott dimenzióhoz "hozzátranszformáljuk"/kifeszítjük a beszkennelt képet.

-Utolsó lépésben pedig élesítjük a képet és megmutatjuk a felhasználónak az összes lépésben elvégzett műveletek eredményét külön felugró képeken.

A fejlesztéshez Thonny 3.2.3-ját használtham Python 3.7.5-el.

Python csomagok:

-numpy

-opencv-contrib-python (OpenCV 4.1.1.26 verziója)

Alapvetően 2 python fájl fog kelleni a működéshez. Az egyik a ***szkenner.py*** a másik pedig a ***fuggvenyek.py***.

Alapértelmezésben egy print metódus található a szkenner.py fájlban aminek a sztring paraméterét módosítva az idézőjelek között fogjuk tudni megadni a beszkennelni kívánt kép nevét.:

print(kepinput("A4_fekvo_1.jpg"))

A program lelke azonban a ***fuggveny.py*** fájlban található.

Itt 5 függvénnyel találkozhatunk, rövid leírásuk.:

- def kepinput(kepneve_kiterjesztessel)

A program itt végzi el az összes kép"manipulációt" és adja vissza az eredményt. Az az összes többi függvény ebbe a függvénybe lesz meghívva.

- def elesitett(dst)

A kép élesítéséhez használt függvény.

- def konturrendezo(anegypont)

A megtalált végsőkontúrokat rendezzük egy tömbben a következő sorrendben:

bal-felső -> jobb-felső -> jobb-alsó -> bal-alsó

- def aktualiskepszama()

A program kiírja a shell-re, az aktuális kép(ek) kontúrpontjainak helyzetét, pixelben. Ez a funkció arra is jó, hogy leellenőrizzük a programunk pontosságát összevetve a manuálisan vett kontúrpontokkal.

- def kep_atmeretezes_aranyosan(kep, szelesseg = None, magassag = None, inter = cv2.INTER_AREA)

Ez a függvény arányosan fogja átméretezni az általunk megadott pixel értékhez a kép többi részét.

Tesztelés

A letöltött mappában lévő **01_Teszt_Kepek nevű** mappában, található mind a 3 fajta méretű dokumentáció. Célszerű ezekbe a mappákba bemásolni a szkenner.py és a fuggvenyek.py nevű fájlt a gyorsabb tesztelés érdekében. Valamint található a főkönyvtárban egy hivasok.py nevű python fájl is. Ebben a fájlban az összes (ebben a mappában lévő)képhez megtalálható a print függvény így csak kikell onnan másolni bele a szkenner.py nevű fájlba és már indulhat is a tesztelés.

Ebben a mappában lévő 30 darab képhez készítettem egy összevető ellenőrzést is amit a **02_Konturpontok_Osszehasonlitasa** nevű mappában található Kontúrpontok összehasonlítása.pdf-ben lehet megtalálni.

Ebben a mappában lévő összes képen egy fekete pixellel megjelöltem azokat a (x,y) koordinátákat amiket a pdf-ben leírtam. Így nagyításnál könnyen látszanak a dokumentum sarkai amiket manuálisan vettem összehasonlítási pontnak. Így a pdf-ben található 2 hasábban a bal oldalon a program által kidobott koordináták szerepelnek míg a jobb oldalon az általam kézzel felvett kontúrpontoknak vélt pixelek.

Valamint mellékeltem a főkönyvtárban található **03_Extra_Teszthez_Kepek nevű** mappában további 86 darab képet amivel lehet tesztelni a program működését és helyességét.

Összesen 116 képet mellékeltem és mind a 116 helyes működést produkál.

Irodalomjegyzék:

https://docs.opencv.org/4.1.1/

https://docs.python.org/3.7/

https://www.learnpython.org

https://opencv-python-tutroals.readthedocs.io/en/latest/

A végére pedig a megírt program kódszinten értelmezve:

```python
import numpy as np
import cv2

kep = 0

def kepinput(kepneve_kiterjesztessel):

    #Bekérjük a képet

    img = cv2.imread(kepneve_kiterjesztessel,1)



    #átméretezzük

    img = kep_atmeretezes_aranyosan(img, magassag = 700)



    #felhasználónak megmutatjuk egy felugró ablakban az átméretezett képet.

    cv2.imshow("image.jpg",img)



    #szürkeárnyalati konverió Canny detektorhoz

    szurkekep = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



    #felhasználónak megmutatjuk a konverzió után egy felugró ablakban a képet.

    cv2.imshow("szurkekep.jpg",szurkekep)



    #későbbi éldetektálást segítő Gauss szűrő alkalmazása Canny detektorhoz
    #5X5ös kernellel

    simitottkep = cv2.GaussianBlur(szurkekep, (5,5),0)



    #felhasználónak megmutatjuk a simítás után egy felugró ablakban a képet.

    cv2.imshow("simitottkep.jpg",simitottkep)



    # Canny detektor 35-60 as threshold-dal.

    eldetektaltkep = cv2.Canny(simitottkep,35,60)



    #felhasználónak megmutatjuk az éldetektált  képet egy felugró ablakban.

    cv2.imshow("eldetektaltkep.jpg",eldetektaltkep)



    #A kontúrokat egyenlő hierarchia szinten kezeljük (cv2.RETR_LIST),
    #valamint a kontúrvonalak végpontjait vesszük- (cv2.CHAIN_APPROX_SIMPLE)

    contours,hierarchy = cv2.findContours(eldetektaltkep,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)


```

```
69
70      #rendezzük a megtalált kontúrokat(contours) területi nagyságuk szerint (key=cv2.contourArea)
71      # a legnagyobbal kezdve (reverse = True), mert alapból növekvő sorrendben rendez
72
73      contours = sorted(contours, key=cv2.contourArea, reverse = True)
74
75
76
77      #Meghívjuk a ciklust az összes kontúrra
78
79      for x in contours:
80
81          #visszaadja egy zárt (True) alaknak az ivhosszat (cv2.arcLength)
82          #0.01re állítva epsilon értékét: ami a maximum távolságot jelenti
83          #a kontúrtól, jelen esetben ez 1%
84
85          ivhossz = 0.01*cv2.arcLength(x,True)
86
87          #közelítő értéket ad vissza az ivhossz es a kontúrokból úgy, hogy
88          #egy zárt (True) közelítő kontúrvonalat keresünk
89
90          kozelito = cv2.approxPolyDP(x,ivhossz,True)
91
92          #visszaadja az elemek számát egy tárolóból
93
94          if len(kozelito) == 4:
95              vegsokonturok = kozelito
96              break
97
98
99
100
101     kozelito=konturrendezo(vegsokonturok) #részletesen leírva a függvény definíciójánál
102
103
104
105     #részletesen leírva a függvény definíciójánál
106     aktualiskepszama()
107     print(kozelito)
108
109
110
111     # A felhasználó itt állítja be (kiveszi a komment jelet: # , az elől a sor elől
112     # amilyen mérettel szeretne dolgozni.
113
114
115     ################################## Méret Állítás ##############################
116
117
118     #pts=np.float32([[0,0],[800,0],[800,800],[0,800]])  # egyenlő oldalú lap
119     #pts=np.float32([[0,0],[496,0],[496,702],[0,702]])  # A4-es méret álló
120     #pts=np.float32([[0,0],[702,0],[702,496],[0,496]])  # A4-es méret fekvő
121
122
123     op=cv2.getPerspectiveTransform(kozelito,pts)  #vesszük a kép felülnézetét
124
125
126     #dst=cv2.warpPerspective(img,op,(800,800)) # egyenlő oldalú lap
127     #dst=cv2.warpPerspective(img,op,(496,702)) # A4-es méret álló
128     #dst=cv2.warpPerspective(img,op,(702,496)) # A4-es méret felvő
129
130
131     ################################## Méret Állítás ##############################
132
133
134
135
136     #kontúrpontok megrajzolása
137
```

```
138     konturpontok = cv2.drawContours(img, vegsokonturok, -1, (102,255,178), 20)
139
140
141
142     #kontúrpontok megjelenítése a felhasználónak egy felugró ablakon keresztül
143
144     cv2.imshow("Konturpontok",konturpontok)
145
146
147
148     #A szkennelt kép megjelenítése a felhasználonak egy felugró ablakon keresztül
149
150     cv2.imshow("Szkennelt",dst)
151
152
153
154     #kép élesítése, részletesen a funkció definíciójánál
155
156     elesitett(dst)
157
158
159     cv2.waitKey(0)
160     cv2.destroyAllWindows()
161
162
163
164
165     # konvolúciót használunk egy élesítéshez használt kernel beállítással, a kép élesítéséhez
166
167 def elesitett(dst):
168     kernel = np.array([[0,-1,0], [-1,5,-1], [0,-1,0]])
169     im = cv2.filter2D(dst, -1, kernel)
170     cv2.imshow("Elesitett", im)
171
172
173
174
175
176     # bal-felső -> jobb-felső -> jobb-alsó -> bal-alsó sorrendben (x,y) koordinátákat felvesszük
ebben
177     # a sorrendben rendezzük.
178
179 def konturrendezo(anegypont):
180
181     # felvesszük a végső kontúrpontokat új "négyzet" 2d-s "4 soros, 2 adattagos" alakba
182
183     anegypont = anegypont.reshape((4,2))
184
185
186     #felveszünk egy új 2d-s tömböt float32 adattípussal és feltöljük nullákkal
187
188     rendezettnegypont = np.zeros((4,2),dtype = np.float32)
189     add = anegypont.sum(1)
190
191
192
193     # bal-felső pont meglesz az adott végsőkontúrok tömbbeli legkisebb x+y koordináták
szummájával
194
195     rendezettnegypont[0] = anegypont[np.argmin(add)]
196
197
198
199     # jobb-alsó pont meglesz az adott végsőkontúrok tömbbeli legnagyobb x+y koordináták
szummájával
200
201     rendezettnegypont[2] = anegypont[np.argmax(add)]
202
203
```

```
204
205     diff = np.diff(anegypont,axis = 1)
206
207     # jobb-felső pont meglesz az adott végsőkontúrok tömbbeli legkisebb x-y koordináták
különbségével
208
209     rendezettnegypont[1] = anegypont[np.argmin(diff)]
210
211
212     # jobb-felső pont meglesz az adott végsőkontúrok tömbbeli legnagyobb x-y koordináták
különbségével
213
214     rendezettnegypont[3] = anegypont[np.argmax(diff)]
215
216
217     # visszaadjuk a megadott sorrendben rendezett kontúrpontok koordinátáit
218
219     return rendezettnegypont
220
221
222
223
224
225 def aktualiskepszama():
226
227     # kiírja az aktuálisan feldolgozás alatt álló kép sorszámát, ha több kép lett megadva a
mainben
228     # akkor mindegyik képhez kiírja a "sorszámát" utána pedig hogy hol található a kontúrpontok
koordinátái
229     # ez a funkció segít összevetni a manuálisan meghatározott kontúrpontokhoz képest a program
pontosságát
230     # bal-felső -> jobb-felső -> jobb-alsó -> bal-alsó sorrendben (x,y) formátumban kiírja a
konzolra
231
232     global kep
233     kep += 1
234     print(str(kep) + ". kep konturpontjai:")
235
236
237
238
239
240 def kep_atmeretezes_aranyosan(kep, szelesseg = None, magassag = None, inter = cv2.INTER_AREA):
241
242     #függvény a kép átméretezéséhez arányosan
243     # felvesszük a meret nevű változót amibe az átméretezett kép méretét tesszük bele
244
245     meret = None
246
247
248     # vesszük a kép magasságát és szélességét
249
250     (m, sz) = kep.shape[:2]
251
252
253
254     # ha a paramétereknél nem adunk meg sem magasságot sem szélességet akkor
255     #az eredeti kép méretét kapjuk
256
257     if szelesseg is None and magassag is None:
258         return kep
259
260
261     # ha csak a magasság van megadva akkor kiszámolja a szélességet hozzá arányosan
262     # és eltárolja meret nevű változóban
263
264     if szelesseg is None:
265
266
```

```
267         r = magassag / float(m)
268         meret = (int(sz * r), magassag)
269 
270 
271     # egyébként a szélesség van megadva és kiszámolja a magasságot hozzá arányosan
272 
273     else:
274         r = szelesseg / float(sz)
275         meret = (szelesseg, int(m * r))
276 
277 
278     # újraméretezi a képet
279 
280     ujrameretezettkep = cv2.resize(kep, meret, interpolation = inter)
281 
282 
283     # visszaadja az újraméretezett képet
284 
285     return ujrameretezettkep
```