



Gépi látás

GKNB_INTM038

Feleletválasztós kérdőív kiértékelése *-Fejlesztési dokumentáció-*

Vincze Petra

Z8OPOU

Győr, 2020/2021/1

Tartalomjegyzék

1	Feladat leírása.....	3
2	Projekt elkészítése	4
2.1	Első rész – a kép előkészítése.....	4
2.2	Második rész – a kérdőív kiértékelése.....	5
3	Tesztelés	8
3.1	A tesztelés előkészítése	8
3.2	Eredmények	12
3.3	Felmerülő kérdések.....	13
4	Források.....	14

1 Feladat leírása

Féléves projektemben egy olyan programot készítettem, mely képes leellenőrizni, hogy egy kérdőív formailag helyesen lett-e kitöltve.

A program bemenetként kap egy képet, melyen kérdések és a hozzájuk kapcsolódó válaszlehetőségek szerepelnek. Minden válasz mellett megtalálható egy kör vagy egy négyzet alakzat, ezek jelzik, hogy az adott kérdéshez egy vagy egynél több helyes megoldás tartozik-e. Ahol kör szerepel, ott csak egy lehetőség van, ahol négyzet, ott több is elfogadható. A program feladata, hogy tájékoztassa a felhasználót arról, hogy az előbb leírt feltételeknek megfelelően töltötte-e ki a kérdőívet, illetve ha nem, akkor mely kérdések esetén hibázott.

2 Projekt elkészítése

2.1 Első rész – a kép előkészítése

A programot a kerdoiv.py fájlba, Python programozási nyelven és az OpenCV használatával készítettem el. Első lépésként importáltam a szükséges Python csomagokat, mint az imutils, numpy, argparse, cv2.

Az argparse lehetővé teszi, hogy parancssorból be lehessen olvasni a kiértékelendő képet, ehhez a --image kapcsolót kell használni, mely a bemeneti kép elérési útját tartalmazza. A betöltött képet szürkeárnyalatossá kell konvertálni és csökkenteni kell a magas frekvenciájú zajokat, majd a Canny él detektáló és a cv2.findContours függvény segítségével meg kell találni a teszt körvonalait. Ez abban az esetben hasznos, ha a kitöltött papírt valamilyen eszköz segítségével fotózzuk le, azaz nem csak a lap látszódik a képen. E nélkül később nem fogjuk tudni alkalmazni a four-point transzformációt. Feltételezzük, hogy a teszt lapja helyezkedik a betöltött kép fókuszpontjában, így az rendelkezik a legnagyobb területtel. Ha a megtalált kontúrokat csökkenő sorrendbe rendezzük, akkor a lista elején kell elhelyezkednie a lapot jelölő körvonalnak is. Ha emellé még a kontúrok közelítésével megvizsgáljuk, hogy azok hány csúccsal rendelkeznek, akkor az első, 4 csúccsal rendelkező kontúr fog a tesztlaphoz tartozni.

```

38 # konturok megkeresese az elterkepen
39 #a dokumentum konturjainak inicializalasa
40 cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
41     cv2.CHAIN_APPROX_SIMPLE)
42 cnts = imutils.grab_contours(cnts)
43 docCnt = None
44
45 #ellenorzes, hogy min egy kontur van
46 if len(cnts) > 0:
47     #konturok csokkeno sorrendbe rendezese
48     cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
49
50     # vegigmegyunk a rendezett konturokon
51     for c in cnts:
52         #konturok kozelítése
53         peri = cv2.arcLength(c, True)
54         approx = cv2.approxPolyDP(c, 0.02 * peri, True)
55
56         #ha 4 pontot talalunk, akkor beazonosítottuk a papírt
57         if len(approx) == 4:
58             docCnt = approx
59             break

```

Ezután a felhasználó segítségével van szükség, ugyanis a four-point transzformációt csak abban az esetben lehet végrehajtani, ha a betöltött kép nem scannelt, vagy online, azaz nincs eredetileg is felülnézetben. A felhasználónak az 1-es gombot kell megnyomnia, ha szükség van a transzformációra és a 0-s billentyűt, ha nincs.

```

61 #szuksege van-e four point perspective transzformacioara
62 kepe = input("Ha a kerdoiv valamilyen eszkozzel lett lefotozva, nyomja meg az 1-es
63
64 #ha lefotozott papir a kep
65 if kepe == '1':
66     #alkalmazunk four point perspective transzformaciot a felulnezei kepert
67     #az eredeti kepen
68     paper = four_point_transform(image, docCnt.reshape(4, 2))
69     #a szurkearnyalatos kepen
70     warped = four_point_transform(gray, docCnt.reshape(4, 2))
71
72     #Otsu metodussal alakitsuk at a kepet binaris kepe
73     #elkulonul a hatter es a kerdesek, valaszok
74     thresh = cv2.threshold(warped, 0, 255,
75                             cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
76
77 elif kepe == '0':
78     #Otsu metodussal alakitsuk at a kepet binaris kepe
79     #elkulonul a hatter es a kerdesek, valaszok
80     thresh = cv2.threshold(gray, 0, 255,
81                             cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
82
83 else:
84     print('Hibas gomb')

```

A kép feldolgozásának utolsó lépéseként vagy a transzformált vagy a szürkeárnyaltos képet bináris képpé kell alakítani Otsu módszerrel, hogy elkülönüljenek a kérdések és a válaszok a háttértől.

2.2 Második rész – a kérdőív kiértékelése

Ahhoz, hogy a teszt kitöltésének helyességét meg tudjuk vizsgálni, szükség van néhány változóra. Kell egy változó, ami azt mutatja meg, hogy hányadik kérdésnél tartunk, kezdőértéke 1 ('question'), egy, ami kérdésenként a jelölt válaszok számát tartalmazza és 0-ról indul ('answers'), illetve egy logikai változóra is szükség van, amit akkor állítunk át igazról hamisra, ha az adott kérdésnél nem megfelelő számú válasz van jelölve ('helyes'). Valamint egy 's' nevű változó segítségével fogjuk megvizsgálni, hogy milyen alakzat található az adott kérdés mellett.

Az 's' egy Shape osztályt jelöl, amit az alakzat.py fájlban hoztunk létre. Ehhez az osztályhoz tartozik egy detect függvény, melynek, ha megadunk különböző kontúrokat, a korábban is használt közelítő módszerrel meg tudja róluk állapítani, hogy hány csúcsuk van, ezáltal pedig, hogy körrel vagy négyzetről van-e szó.

```

1  # szükséges csomagok importálása
2  import cv2
3
4
5  class Shape:
6      def __init__(self):
7          pass
8      def detect(self, c):
9
10         # initialize the shape name and approximate the contour
11         shape = "unidentified"
12         peri = cv2.arcLength(c, True)
13         approx = cv2.approxPolyDP(c, 0.04 * peri, True)
14
15         if len(approx) == 4:
16             shape = "negyzet"
17         else:
18             shape = "kor"
19
20         return shape

```

A program a bináris képen újra megkeresi a kontúrokat, majd ezek közül a képarány segítségével kiválasztja azokat, amelyek a válaszlehetőségekhez tartozó alakzatokat jelölik és a 'questionCnts' nevű listához adja őket.

A továbbhaladáshoz ismét a felhasználó segítségére van szükség, ugyanis meg kell adni, hogy hány válasz tartozik egy-egy kérdéshez. A program csak azokat a kérdőíveket képes megvizsgálni, ahol minden kérdéshez azonos számú válasz tartozik.

Ezután egy for ciklusban újra megvizsgáljuk kontúrokat. Léptéknek azt a számot állítjuk be, amit a felhasználó megadott. A cikluson belül létrehozunk egy 'negyzet' nevű logikai, illetve egy 'hanyadik_valasz' nevű változót, mely kezdőértéke 1 lesz. A cikluson belül elindítunk még egy ciklust, amiben először azt vizsgáljuk meg, hogy a hanyadik_valasz változó milyen értékkel rendelkezik. Ha 1-gyel, akkor meghívódik az 's' változó detect függvénye és eldől, hogy az adott kérdésnél négyzet vagy kör szerepel-e a válaszok mellett. Mivel egy kérdéshez csak egy fajta válaszlehetőség tartozik, ezért elég a vizsgálatot csak minden kérdés első válaszánaál megtenni.

```

137 #vegigmegyunk a tombon, akkor leptekek amekkorat a felhasznalo megadott
138 for (q, i) in enumerate(np.arange(0, len(questionCnts), answersCount)): #
139     #kerdesek rendezese balrol jobbra
140     cnts = contours.sort_contours(questionCnts[i:i + answersCount])[0] #
141
142     #logikai valtozo inicializalasa a valaszokhoz tartozo alakzat megallapitasa
143     negyzet = False
144     #valtozo inicializalasa ahhoz, hogy tudjuk, a kerdeshez tartozo elso
145     hanyadik_valasz = 1
146
147     #vegigmegyunk a konturokon
148     for (j, c) in enumerate(cnts):
149
150         #megnezzuk, hogy az alakzat negyzet-e vagy sem
151         #eleg csak minden kerdesnel az elso alakzatot megvizsgalni
152         while hanyadik_valasz == 1:
153
154             alakzat = s.detect(c)
155             #print (alakzat)
156             if alakzat == "negyzet":
157                 negyzet = True
158
159             hanyadik_valasz += 1
160

```

Utána készítünk egy maszkot, ami az éppen aktuális válaszlehetőséget mutatja, és ezt alkalmazzuk a bináris képen. Majd összeszámoljuk a nem 0 pixeleket és az értéket eltároljuk a 'total' változóban. Ha a 'total' értéke egy bizonyos értéknél nagyobb, vagy azzal egyenlő, akkor a felhasználó megjelölte a választ a képen és az 'answers' értékét növeljük eggyel.

```

#keszitsunk maszkot, ami az éppen aktualis valaszlehetoseget mutatja
mask = np.zeros(thresh.shape, dtype="uint8")
cv2.drawContours(mask, [c], -1, 255, -1)

#alkalmazzuk a maszkot a binaris kepen
#szamoljuk ossze a nem-nulla pixeleket a teruleten
mask = cv2.bitwise_and(thresh, thresh, mask=mask)
total = cv2.countNonZero(mask)

#adjunk meg egy erteket ami felett valoszinuleg jelolve van a valaszlehetoseg
#print (total)
if total >= 290:
    answers += 1

```

A ciklus végén megvizsgáljuk, hogy az alakzatnak megfelelő számú válasz volt-e jelölve, és ha nem, akkor erről visszajelzést küldünk a felhasználónak. Végül a 'question' értékét eggyel növeljük, az 'answers'-nek pedig újra 0 értéket adunk.

```

Ha a kerdoiv valamilyen eskozzel lett lefotozva, nyomja meg az 1-es billentyut, ha szkennelve van, nyomja meg a 0-as bi
llentyut: 0
Egy kerdeshez tartozo valaszok szama: 5
A 2. kerdes hibasan lett kitoltve

```

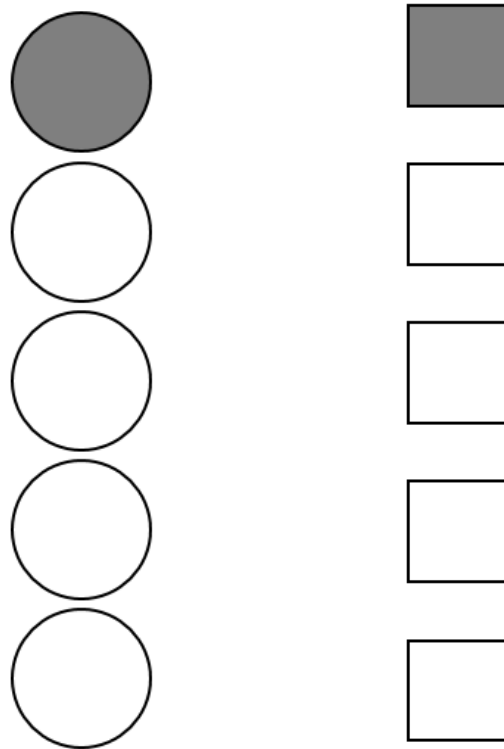
Feleletválasztós kérdőív kiértékelése

```
Ha a kerdoiv valamilyen eszkozzel lett lefotozva, nyomja meg az 1-es billentyut, ha szkennelve van, nyomja meg a 0-as billentyut: 0
Egy kerdeshez tartozo valaszok szama: 5
A kerdoiv helyesen lett kitolelve!
```


3 Tesztelés

3.1 A tesztelés előkészítése

A program megfelelő működését a fejlesztés során folyamatosan teszteltem. A feladat legfontosabb része az volt, hogy a program meg tudja különböztetni egymástól a kör és a négyzet objektumokat. Így a teszteket először olyan képekkel csináltam meg, ahol a hangsúly azon volt, hogy milyen alakzatok szerepelnek a képeken, nem pedig azon, hogy ezek közül melyek különböznek a többitől. Ezeket a teszteket a következő típusú képekkel hajtottam végre:

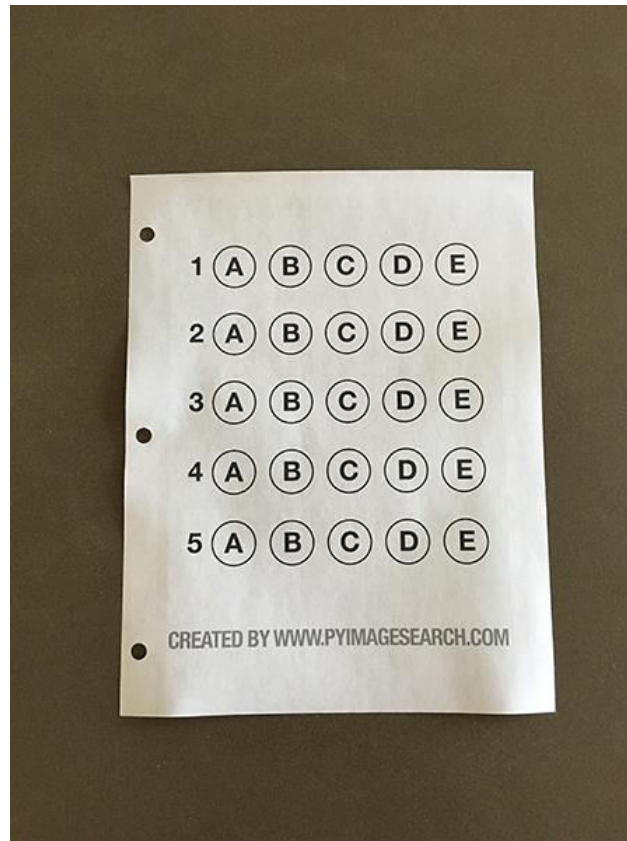


A kész programot összesen 27 képpel teszteltem. A teszteléshez használt képek 4 csoportba oszthatóak. Az első csoportba tartoznak azok a képek, melyeken egy papíralapú teszt fényképe látható. Erre a típusra azért volt szükség, hogy a four-point transzformáció helyes működését is lehessen tesztelni.

A másik három csoportba olyan számítógéppel szerkesztett képek találhatók, melyek hasonló sémára épülnek, a különbség köztük a kérdések és a hozzájuk tartozó válaszok számában mutatkozik. Ezekkel a képekkel főleg azt teszteltem, hogy a program helyesen el tudja-e különíteni egymástól az egy kérdéshez tartozó válaszokat.

Mind a négy képcsoport esetében fele-fele arányban végeztem el a teszteket helyesen, illetve hibásan kitöltött képekkel.

A teszteléshez használt eredeti képek, melyeket képszerkesztővel különböző módokon kitöltöttem:



1. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz
- ☐ válasz
- ☐ válasz

2. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz
- ☐ válasz
- ☐ válasz

1. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

2. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

3. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

4. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

1. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

2. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

3. Kérdés

- ☐ válasz
- ☐ válasz
- ☐ válasz

Példa a kitöltött kérdőívekre:

1. Kérdés

☐

válasz

☒

válasz

☐

válasz

☒

válasz

☐

válasz

2. Kérdés

☐

válasz

☒

válasz

☐

válasz

☒

válasz

☐

válasz

1. Kérdés

☒

válasz

☒

válasz

☐

válasz

2. Kérdés

☐

válasz

☒

válasz

☐

válasz

3. Kérdés

☒

válasz

☒

válasz

☐

válasz

4. Kérdés

☐

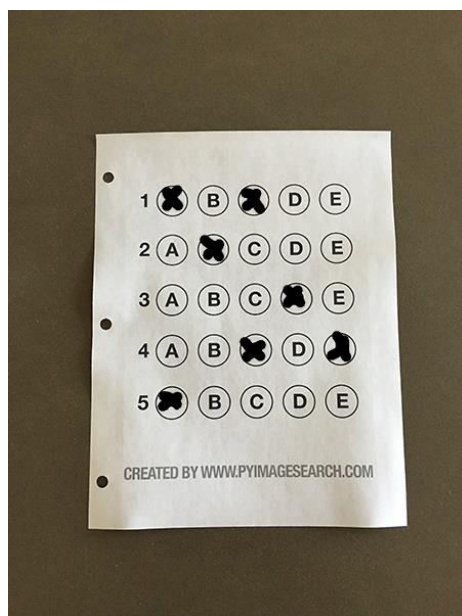
válasz

☒

válasz

☐

válasz



3.2 Eredmények

Az alakzatok helyes felismerésére vonatkozó tesztek 100%-os eredményt hoztak, azaz minden esetben helyes eredményt kaptam. Így kijelenthetjük, hogy az elkészült program bármilyen helyzetben meg tudja különböztetni egymástól a kört és a négyzetet.

A kérdőív helyes kiértékelésére vonatkozó tesztek kb. 92%-os eredményt hoztak. Azaz a 27 képből 25-öt helyesen értékelt ki a program, 2 kép esetében viszont hiba merült fel. Mindkét esetben a hiba a kérdőív kitöltésének módjából származott.

Az első esetben a probléma az alakzat jelölésének módjával volt. A program nem köti ki, hogy satírozással vagy x-szel kell-e bejelölni az alakzatot, a lényeg csupán annyi, hogy a jelölés kellően határozott legyen. Ugyanis ha a nem 0 pixelek vizsgálatánál a kapott szám nem éri el a 290-et, akkor a program nem tekinti jelöltnek az adott választ. Azért 290 lett megadva értéknek, mert a tesztekben az derült ki, hogy ennél kisebb csak nagyon ritkán lesz egy jelölt válasznál a nem 0 pixelek száma.

A hibásan javított kép:

1. Kérdés

- ☒ válasz
- ☐ válasz
- ☒ válasz
- ☐ válasz
- ☐ válasz

2. Kérdés

- ☐ válasz
- ☒ válasz
- ☐ válasz
- ☒ válasz
- ☐ válasz

A második esetben a probléma a jelölés színével volt. Különböző képekkel vizsgáltam, hogy lehet-e feketétől eltérő színnel is kitölteni a kérdőívet. A tesztek alapján ez csak akkor lehetséges, ha a választott szín kellően sötét. Így amikor egy halványsárga színnel teszteltem, a program nem érzékelte, hogy melyik válaszokat jelöltem be és rossz eredményt adott.

Ez a kép a következő volt:

1. Kérdés

- ☒ válasz
- ☒ válasz
- ☐ válasz

2. Kérdés

- ☐ válasz
- ☒ válasz
- ☐ válasz

3. Kérdés

- ☐ válasz
- ☒ válasz
- ☐ válasz

Ezen kívül akkor is hiba lépett fel a tesztelés során, amikor a felhasználó nem megfelelő adatot írt be az egy kérdéshez tartozó válaszlehetőségek számának. Ekkor ugyanis nem tartozott ugyanannyi válaszlehetőség minden kérdéshez, ezt az esetet pedig nem tudja kezelni a program.

3.3 Felmerülő kérdések

A tesztelés során felmerültek bizonyos kérdések, mint például, befolyásolja-e egy hosszabb válaszlehetőség a program helyes futását? A válasz az, hogy nem, mivel a program az egyes kérdésekhez megadott válaszlehetőségek felismerésében kizárólag arra támaszkodik, hogy a felhasználó milyen számot adott meg a válaszok darabszámának.

Egy másik felmerült kérdés, hogy mi történik akkor, ha a kérdőív elkészítése során egy kérdéshez kör illetve négyzet alakzattal jelölt válasz is tartozik, mert a kérdőív készítője hibázott. Itt azt kell mondanom, hogy annak függvényében vizsgálja a program az adott kérdést, hogy az első válaszlehetőség mellett melyik szimbólum szerepel. Vagyis a program lefutását ez nem befolyásolja.

Harmadikként azt a kérdést kell megemlítenem, hogy a program képes-e olyan kérdőívek kiértékelésére, ahol a kérdésekhez nem azonos számú válasz tartozik? Sajnos a program erre nem képes. Bár készült egy ilyen változata is a feladatnak, a tesztek eredményei nem bizonyultak megfelelőnek ahhoz, hogy kijelenthessük, a program minden ilyen esetben megfelelően értékeli ki a kérdőívet.

4 Források

- [1] <https://www.pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/>
- [2] <https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/>