Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 461-468 **https://doi.org/10.35925/j.multi.2020.2.51**

KONTÚRDETEKTÁLÁS DIGITÁLIS KÉPFELDOLGOZÁS SEGÍTSÉGÉVEL

Murányi Sándor
hallgató, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: sandormuranyi@yahoo.com

Absztrakt

A cikk áttekintést ad kontúrok detektálásáról digitális képfeldolgozás segítségével. Részletesen ismerteti a digitális képfeldolgozást és annak előnyeit, valamint egy konkrét területen való alkalmazását is bemutatja. A cikk során a Python nyelvet használjuk, az eredményesség érdekében az ehhez szükséges csomagokat is bemutatjuk. Ezt követően az élek, körök detektálására fogunk bemutatni egy módszert, melyhez a Hough-transzformáció alkalmazása szükséges. Kitekintünk a vékonyítási eljárásra is, amely segítséget nyújt a pontosabb eredmény elérésében. A bemutatott ismeretek segítségével a kontúrok detektálását fogjuk bemutatni egy példaprogram segítségével. A cikk végén röviden kitérünk alakjellemzők számítására is.

Kulcsszavak: *képfeldolgozás, kontúr, kontúrdetektálás, detektálás, Python*

Abstract

This article provides an overview of contour detection using digital image processing. It describes in detail the digital image processing and its advantages, as well as its application in a specific field. In this article, we use the Python language, and for the sake of efficiency, we also present the necessary packages. Next, we will present a method for the detection of edges and circles, which requires the application of the Hough transform. We also look at the thinning process, which helps to achieve a more accurate result. Then, with the help of the acquired knowledge, the detection of the contours will be demonstrated with the help of an example program. At the end of the article, we also briefly discuss the calculation of shape characteristics.

Keywords: *image processing, contour, contour detection, detection, Python*

1. Bevezetés

A számítógépek teljesítményének növekedésének és a mesterséges intelligencia térhódításának is köszönhetően a digitális képfeldolgozás alkalmazása az informatikában manapság nagyon elterjedt. Természetesen nem csak az informatikában, hanem különböző területeken is használják nap, mint nap a képfeldolgozás nyújtotta előnyöket.

Ilyen területek például a teljesség igénye nélkül:

- az egészségügy (CT, röntgen, gyógyszerészet),
- az űrkutatás,
- a meteorológia (időjárás-előrejelzés),
- az élelmiszeripar,
- a biztonságtechnológia.

Jelen esetben a képfeldolgozás segítségével kontúrok detektálására és meghatározására fogunk kitérni. Pontosabban egy olyan program működését mutatjuk be, amely a kontúrtulajdonságokat figyelembe véve megtalálja az eltéréseket egy referencia képen – legyen az a legapróbb, vagy akár egy relatív nagyobb eltérés. Mindezt Python programozási nyelven valósítjuk meg.

Továbbá célunk az is, hogy átlátható, illetve könnyen kezelhető legyen a programunk, mind a fejlesztő, mind a felhasználó számára, valamint ennek bemutatása.

2. Digitális képfeldolgozás

A bevezetésben már ismertettük a digitális képfeldolgozás elterjedését, valamint néhány felhasználási területét. A számítógépes képfeldolgozás nem más, mint egy környezetünkből származó képi információ számítógépes segítséggel való feldolgozása és kiértékelése. Ahhoz, hogy a számítógép fel tudja dolgozni a kívánt képet, digitalizálni kell. A digitalizálás logikailag tulajdonképpen három lépésből áll: a képkinyerésből/leképezésből, a mintavételezésből és a kvantálásból. Ezeket a lépéseket a különböző digitalizáló eszközök nem feltétlenül elkülönítve valósítják meg.

A digitalizálás alapvetően fontos egy digitális kép minőségének meghatározásában, illetve a további feldolgozás eredményessége miatt. Természetesen minél sűrűbb a mintavételezés és minél nagyobb a kvantálási szám, annál jobb lesz a digitális kép minősége is.

A számítógépes feldolgozásnak számos előnye lehet, ami minden esetben más és más tulajdonságból származik, jelen esetben például:

- A számítógép jobban terhelhető, ugyanazt a mérést képes órákon át elvégezni.
- A számítógép „gondolkodásmódja” objektív, így az azonos körülmények mellett mindig ugyanaz lesz az eredmény.
- A képfelismerés célja a képen rögzített valós tartalom felismerése, azonosítása, kiértékelése egy meghatározott séma, vagy adatbázis alapján. A cél az, hogy ezt automatizálva vigyük véghez.

2.1. Képfeldolgozó Python csomagok

Az alábbi képfeldolgozó Python csomagokat használhatjuk többek között:

- *Numpy („Numerical Python”):* Egy olyan kiegészítő csomag a Python programozási nyelvhez, mely a nagy, többdimenziós tömbök és mátrixok használatát támogatja egy nagy magas szintű matematikai függvénykönyvtárral. A többi csomag alapját képezi.
- *SciPy („Scientific Python”):* Numpy csomagra épülő, numerikus számítási algoritmusokat tartalmazó csomag.
- *MatPlotLib:* Numpy csomagra épülő, diagramrajzoló és vizualizációs csomag.
- *IPython („Interactive Python”):* Python parancssori képességeinek nagyfokú bővítése. Matlabszerű prototípus készítés érhető el a segítségével.
- *Scikit-image:* SciPy kiterjesztése további képfeldolgozó műveletekkel.
- *Pillow:* Számos képformátum betöltése és képek feldolgozása.
- *OpenCV:* Multiplatform, ingyenes függvénykönyvtár (BSD licensz). Több mint 2500 optimalizált algoritmus képfeldolgozás, számítógépes látás témakörben. C, C++, Python, Java interfészek, Android és iOS rendszeren is elérhető.

2.2. Geometriai jellemző kinyerése

Ebben a részben azt mutatjuk be, hogyan nyerhetünk ki egy képből strukturális leírást, valamint a kép tartalmát szeretnénk értelmezni. Ezt a már korábban említett képfeldolgozó csomagok segítségével fogjuk megtenni, többnyire OpenCV segítségével.

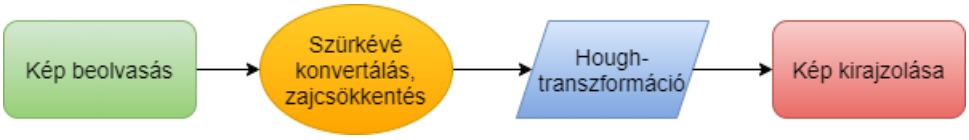
Célunk az, hogy a képből alakzatokat leíró geometriai jellemzőket nyerjünk ki, így pl.: élek, körök, négyszögek, objektum váz, objektumok határának leírása (kontúr), alakjellemzők számítása kontúr alapján.

A célunk eléréséhez a következő módszereket fogjuk felhasználni:

– Egyenesek, körök detektálása: Hough-transzformáció,
– Vékonyítás: Zhang–Suen- és Guo–Hall-módszerekkel,
– Kontúrreprezentáció: lánckód, poligon közelítés.

2.2.1. Élek, körök detektálása

Az egyenes és kör detekcióhoz Hough transzformációt fogunk használni, itt a kör vizsgálatot mutatjuk be. Végig megyünk egy bináris képen, majd minden objektumpontra megvizsgáljuk, hogy milyen középpontú és mekkora sugarú kör megy rajta keresztül, ezt pedig egy összegzőtömbben tartjuk nyilván. A vizsgálat után az összegzőtömbünk lokális maximumértékei adják vissza a detektált köreinket. A dimenziója ennek a tömbnek 3 lesz, mivel egy kört a középpontja és a sugara írja le.

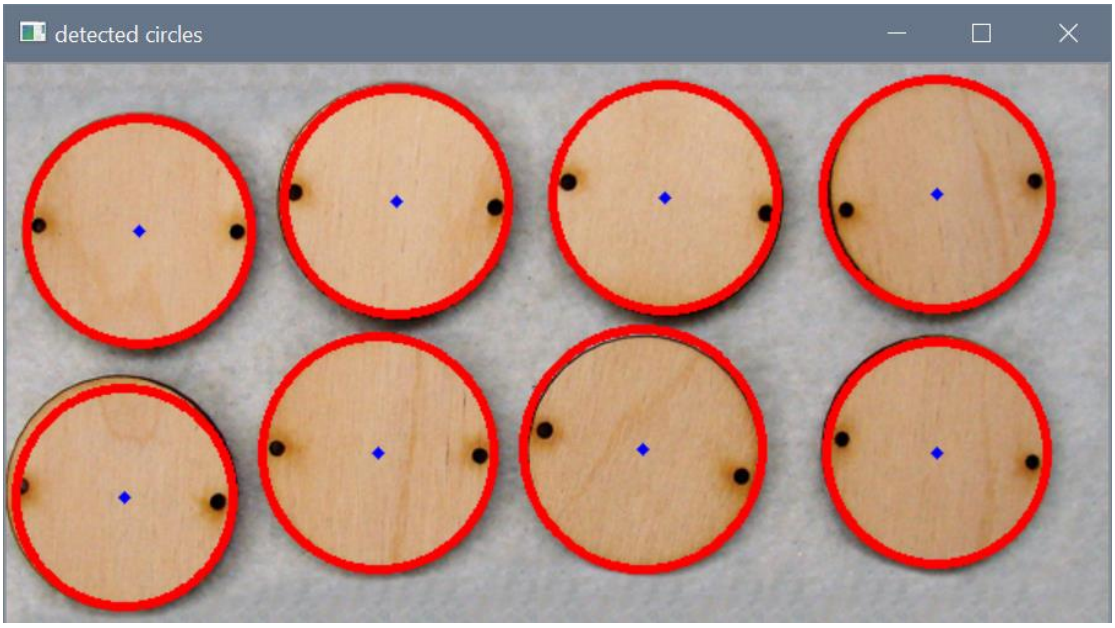


1. ábra. *Folyamatdiagram*

Egy rövid példaprogram és annak eredménye:

```
1 import sys
2 import cv2
3 import numpy as np
4
5
6 def main(argv):
7     ## [Beolvasás]
8     default_file = "szgy_fa4.jpg"
9     filename = argv[0] if len(argv) > 0 else default_file
10
11     # A kép beolvasása
12     src = cv2.imread(filename, cv2.IMREAD_COLOR)
13
14     # Megvizsgáljuk, hogy a képet sikerült-e beolvasni
15     if src is None:
16         print ('Nem sikerült megnyitni a képet!')
17         print ('Usage: hough_circle.py [image_name -- default ' + defa-
18 ult_file + '] \n')
19         return -1
20     ## [Beolvasás]
```

```
21
22      ## [Szürkévé_konvertálás]
23      # Szürkévé konvertálja
24      gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
25      ## [Szürkévé_konvertálás]
26
27      ## [Zajcsökkentés]
28      # Zajcsökkentés a pontosabb kör detektálás miatt
29      gray = cv2.medianBlur(gray, 5)
30      ## [Zajcsökkentés]
31
32      ## [houghcircles]
33      rows = gray.shape[0]
34      circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, rows / 16,
35                                 param1=200, param2=30,
36                                 minRadius=20, maxRadius=100)
37      ## [houghcircles]
38
39      ## [Kirajzolás]
40      if circles is not None:
41          circles = np.uint16(np.around(circles))
42          for i in circles[0, :]:
43              center = (i[0], i[1])
44              print(center, i[2])
45              # kör közepe
46              cv2.circle(src, center, 1, (255, 0, 0), 3)
47              # kör körvonala
48              radius = i[2]
49              cv2.circle(src, center, radius, (0, 0, 255), 3)
50      ## [Kirajzolás]
51
52      ## [Kirajzolás]
53      cv2.imshow("detected circles", src)
54      cv2.waitKey(0)
55      ## [Kirajzolás]
56
57      return 0
58
59
60  if __name__ == "__main__":
61      main(sys.argv[1:])
```



2. ábra. *Detektált körök és azok kirajzolása*

2.2.2. Vékonyítás

Mivel bináris objektumokkal dolgozunk, így szükségünk lehet az objektumok vázára, középvonalára. Sajnos kevés beépített lehetőséget biztosít számunkra az OpenCV. Így vékonyításra a contrib csomagban található `thinning()` függvénnyel van lehetőség, ezen belül is kétféle implementált algoritmus található.

Használata:

```
dst = cv2.ximgproc.thinning(src, [dst, [thinningType]])
```

Az src a bemeneti, 8-bites egész típusú, egy csatornás bináris kép, amelyben a 255 értékek jelzik az objektumpontokat.

A thinningType kétféle értéket vehet fel:

`cv2.ximgproc.THINNING_ZHANGSUEN` és `cv2.ximgproc.THINNING_GUOHALL`.

2.2.3. Kontúrok detektálása

Bináris képeken az objektumok körvonalának meghatározására, illetve kontúrok kirajzolására az OpenCV biztosít lehetőséget. Szürkeárnyalatos vagy színes képek esetén a bináris bemenet előállításához előzetesen küszöbölést vagy szegmentálást kell végezni a képen. Nem kapunk hibát, ha nem végezzük el, viszont az OpenCV binárisként kezeli a szürkeárnyalatos képet (0: háttérpont, >0: objektumpont). Azok az objektumpontok lesznek kontúrpontok, amelyeknek van háttérpont szomszédja.

Kontúrok képmátrixba rajzolására az OpenCv egy segédfüggvényt biztosít számunkra:

`findContours()`

Kontúrok poligonnal való közelítésére a következő függvénnyel van lehetőségünk, megadott pontossággal. Mivel kevesebb definiáló pontból fog állni, így a kontúrunk simább lesz, de ez felveti annak a lehetőségét, hogy nem illeszkedik tökéletesen az eredeti körvonal pontokra:

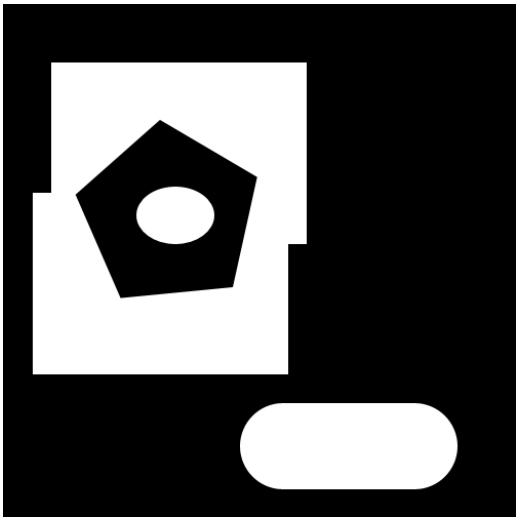
```
approxCurve = cv2.approxPolyDP(curve, epsilon, closed[, approxCurve])
```

Példaprogram

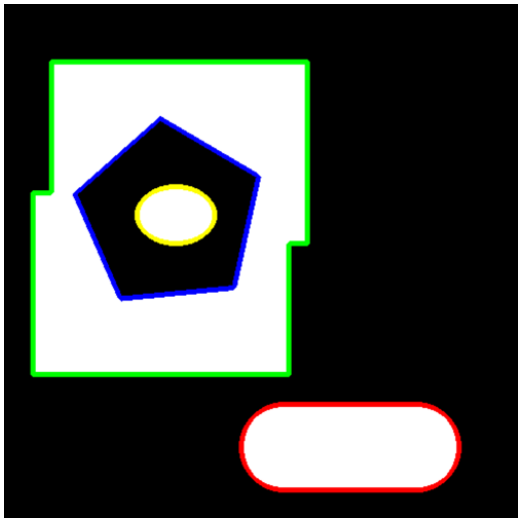
Miután a tesztképet betölti, küszöböli azt, majd a kontúrkeresést végzi el. Ezután a megtalált kontúrokat a képmátrixba rajzolja egyenként.

```
 1 import cv2
 2
 3 line_colors = [(0, 0, 255), (0, 255, 0), (255, 0, 0), (0, 255, 255),
 4 (255, 0, 255), (255, 255, 0), (255, 255, 255)]
 5
 6 # Kép beolvasás
 7 im = cv2.imread('szgy_kontur1.png')
 8 imgray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
 9 ret, thresh = cv2.threshold(imgray, 127, 255, 0)
10
11 # Kontúrok keresése
12 contours = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
13
14 # Kontúrok kirajzolása
15 for cntrIdx in range(0, len(contours)):
16     cv2.drawContours(im, contours, cntrIdx, line_colors[cntrIdx % 7], 3,
17 cv2.LINE_4)
18     cv2.imshow('Contours', im)
19     cv2.waitKey(0)
20
21 cv2.destroyAllWindows()
```

A beolvasott képünk és az eredmény:



3. ábra. *Beolvasott bináris kép*



4. ábra. *Detektált kontúrok kirajzolása*

2.2.4. Alakjellemzők számítása kontúr alapján

Mivel a `findContours()` függvény eredményként a kontúrok egy listáját adja vissza, viszont a feldolgozó függvények egy-egy kontúrral dolgoznak, így észszerű ciklussal bejárni a lista elemeit.

- Kontúr által közrezárt terület: Előjeles eredményt akkor kapunk a kontúr körüljárási irányától függően, ha az oriented paraméter True. Alapértelmezett a False.
  `retval = cv2.contourArea(contour, oriented)`
- Kontúr hossza: A closed értéke legyen True.
  `retval = cv2.arcLength(contour, closed)`
- Konvexitás vizsgálat, logikai eredménnyel. `retval = cv2.isContourConvex(contour)`
- Legkisebb területű (forgatott) befoglaló téglalap: Eredményként megkapjuk a téglalap középpontjának koordinátáját, a méreteit (magasság és szélesség), és az elforgatási szögét. Lebegőpontos típusúak az értékek.
  `rect_center, rect_size, rect_angle = cv2.minAreaRect(contour)`

3. Összefoglalás

A cikk során bemutatott eljárásokkal jól látható eredményeket sikerült elérni, a kontúrok detektálása, meghatározása és kirajzolása volt a cél, amit sikerült megvalósítani. Ezeket a technikákat számos területen lehet alkalmazni, természetesen kibővítve, más paraméterekkel.

Levonható az a következtetés, hogy viszonylag egyszerűen lehet a Pythonban megvalósítani a kontúrokkal való műveleteket.

4. Köszönetnyilvánítás

A cikkben ismertetett kutatómunka az EFOP-3.6.1-16-2016-00011 jelű Fiatalodó és Megújuló Egyetem – Innovatív Tudásváros – a Miskolci Egyetem intelligens szakosodást szolgáló intézményi fejlesztése” projekt részeként – a Széchenyi 2020 keretében – az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Irodalom

[1] van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T. the scikit-image contributors (2014). Scikit-image: image processing in Python. *Peer J.*, 2, e453. **https://doi.org/10.7717/peerj.453**

[2] Lindblad, T., Kinser, J. M. (2013). *Image processing using pulse-coupled neural networks*. Springer-Verlag, Berlin, Heidelberg, ISBN 978-3-642-36876-9. **https://doi.org/10.1007/978-3-642-36877-6**

[3] Chityala, R., Pudipeddi, S. (2021). *Image processing and acquisition using Python*. Taylor & Francis Group, LLC, Boca Raton, FL, ISBN: 9780367198084. **https://doi.org/10.1201/9780429243370**

[4] Gouillart, E., Nunez-Iglesias, J., van der Walt, S. (2016). Analyzing microtomography data with Python and the scikit-image library. *Adv Struct Chem Imag*, 2, 18. **https://doi.org/10.1186/s40679-016-0031-0**

[5] Canty, M. J. (2014). *Image Analysis, Classification and Change Detection in Remote Sensing: With Algorithms for ENVI/IDL and Python*. Third Edition, CRC Press, ISBN 1466570385. **https://doi.org/10.1201/b17074**

[6] Minichino, J., Howse, J. (2015). *Learning OpenCV 3 computer vision with Python*. Packt Publishing Ltd, ISBN 1785289772, 9781785289774.
[7] Czap, L. (2007). *Képfeldolgozás*. Miskolci Egyetem.
[8] Downey, A. B., Elkner, J., Meyers, C. (2002). *How to think like a computer scientist*. ISBN-10:0971677506.