

Pénzérme számláló alkalmazás

Fejlesztői és felhasználói dokumentáció

Bognár Milán Károly

Neptun kód: S3PH8C

Gépi látás (GKNB_INTM038)

2020/2021. tanév, őszi félév

Tartalomjegyzék

1) Fejlesztői dokumentáció.....	3
1.1. Feladat bemutatása, elvárt funkciók.....	3
1.2. Fejlesztői környezet, Python könyvtárak.....	4
1.3. Feladat elméleti háttere	5
1.3.1. Körök detektálása Hough transzformációval:	5
1.4. Feladat megoldása, program felépítése	7
1.4.1 HSL színtér	10
1.5. Tesztelés	12
1.6. Fejlesztési lehetőségek.....	13
2) Felhasználói dokumentáció	14
3) Források.....	17

1) Fejlesztői dokumentáció

1.1. Feladat bemutatása, elvárt funkciók

Féléves feladatomban egy olyan alkalmazás elkészítése volt a cél, amely képes felismerni pénzérmekeket az általunk készített képekről. A programnak fel kell ismernie a pénzérmekek típusait és meg kell határoznia, hogy a képen látható érméknek mennyi az összértékük. A pénzérmekek jellemzőit előre definiáljuk, csak az ilyen tulajdonságokkal rendelkező érméket kell felismernie az alkalmazásnak. A program a magyar forint típusú érméket képes felismerni (5 Ft, 10 Ft, 20 Ft, 50 Ft, 100 Ft, 200 Ft).

A programot Python nyelven készítettem el, kihasználva ennek a programozási nyelvnek az előnyeit, mint például a programozási nyelvhez készített csomagok, amelyek megkönnyítik a képfeldolgozást és a kép elemeinek elemzését.

Bemenet:

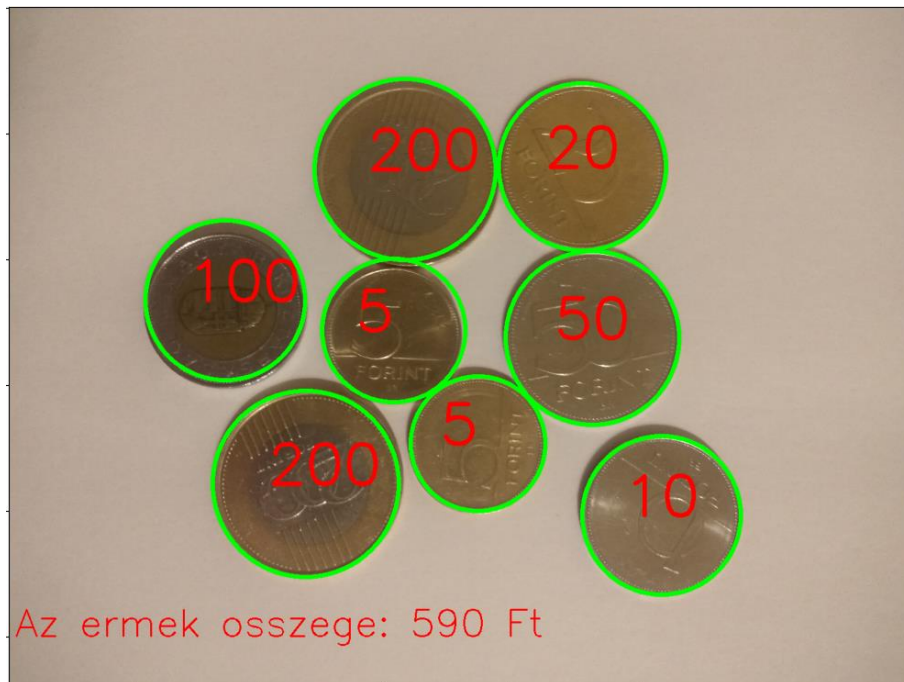
Általunk készített kép, amely magyar forint típusú pénzérmekeket tartalmaz, illetve fehér háttér előtt készült. A bemeneti kép lehetőleg minél kevesebb árnyékot tartalmazzon, mert az ronthatja a program sikerességét.



1.ábra: Bemeneti kép illusztrációja

Kimenet:

A kimeneti képen jelenjen meg a felismert pénzérték értéke külön-külön, illetve legyen kiírva a képre a megtalált pénzérték összértéke forintban. A kimeneti képeken zöld körökkel láthatjuk azt is, hogy az adott pénzértéket milyen pozícióban találta meg a program.



2.ábra: Kimeneti kép illusztrációja

1.2. Fejlesztői környezet, Python könyvtárak

A programot Visual Studio 2019 fejlesztői környezetben, Python programozási nyelven készítettem el.

A Python nyelvhez találhatunk csomagokat, amelyek megkönnyítik a képfeldolgozás lépéseit számunkra. Ezek közül az alábbiakat használtam:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from tkinter import filedialog
import tkinter as tk
```

3.ábra: A programban felhasznált Python csomagok listája

1.3. Feladat elméleti háttere

A pénzérme felismerés feladat első részében az érmék pozícióját kell meghatároznunk. A pénzérme kör alakú objektumok, tehát a képen meg kell keresnünk a nekünk megfelelő méretű kör alakzatokat. Ezt megtehetjük többféleképpen is. Amennyiben szeretnénk minden egyes lépést magunk megcsinálni, és előre megírt, beépített függvény nélkül megoldani a feladatot, akkor a lépések a következők lennének:

- Szürkeárnyalatossá alakítás
- Gauss szűrő alkalmazása a bemeneti képen
- Éldetektálás: Canny éldetektor segítségével
- Megadott sugarú körök készítése, amelyekkel végig iterálunk a kapott képen, és megkeressük, hogy vannak-e ilyen sugarú körök
- Amennyiben találunk egyezést, az adott kör középpontjának koordinátáit eltároljuk, illetve a sugár méretét is elmentjük

Ez a módszer nem biztos, hogy a leghatékonyabb, illetve elég sok helyen hiba kerülhet a kódba. Az előbb felvázolt módszer mintájára találhatunk az OpenCV könyvtárban egy függvényt, amely megkeresi nekünk a köröket a képen. A függvény neve: `cv2.HoughCircles`

1.3.1. Körök detektálása Hough transzformációval:

A körök detektáláshoz szükségünk van egy transzformációra a képtérből a Hough térbe. Ez a transzformáció megadja az egy ponton átmenő összes egyenes egyenletét. A képtérben egy pontot x és y koordinátákkal jelölhetünk. Ehhez a képponthoz tartozó görbe a Hough térben a következő: $r = x * \cos(\varphi) + y * \sin(\varphi)$

A kör detektálása kétféleképpen történhet:

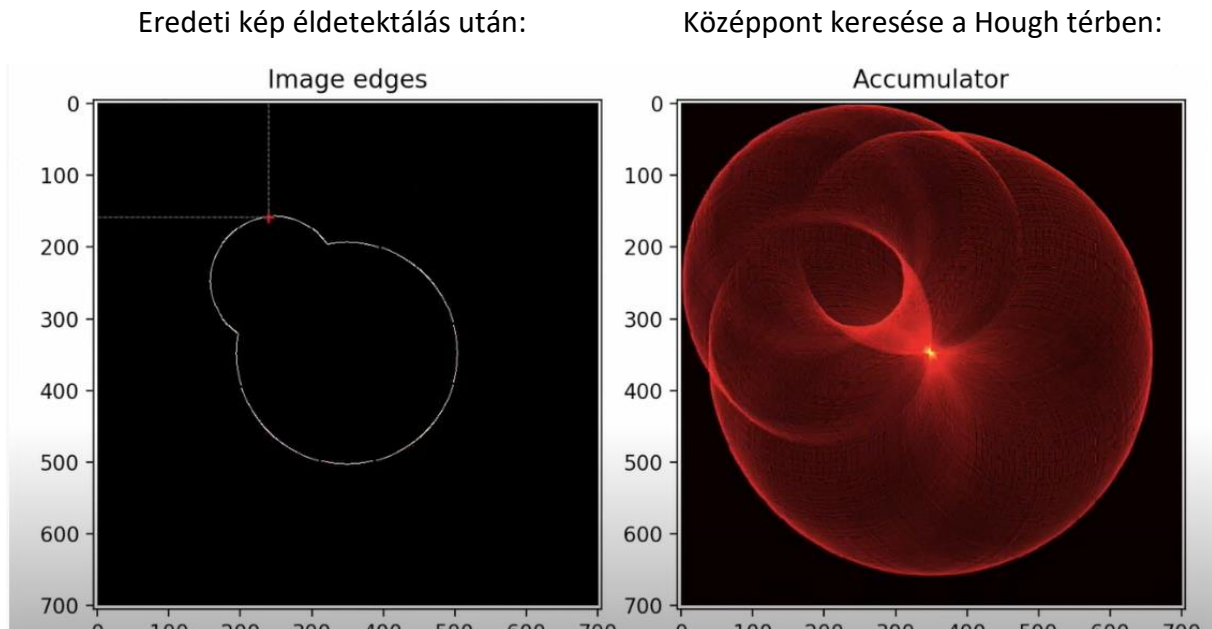
- Ismert sugarú köröket keresünk (például ipari környezetben használnak ilyeneket)
- Ismeretlen sugarú köröket keresünk (Egyéb, például mobilos felismerő applikációk esetén)

Általános körök esetén a Hough tér 3 dimenziós lesz (amikor nem ismert a keresett kör sugara). Amennyiben ismert a sugár, akkor 2 dimenziós térről beszélünk.

Kör egyenlete a képtérben: $ax^2 + by^2 = r^2$ ahol (a,b) a kör középpontja, r a sugár

Abban az esetben, ha a sugár rögzített, akkor a Hough tér 2 dimenziósra csökken. Az eredeti kör minden egyes pontjára (x, y) meghatározhat egy (x, y) középpontú kört r méretű sugárral. Az összes ilyen kör metszéspontja a paramétertérben megegyezik az eredeti kör középpontjával. A 4. ábrán ezt a keresési folyamatot láthatjuk. A bal oldali képen látjuk az eredeti képet éldetektálás után. A forrásként megjelölt videóban nagyon jól szemléltetik, hogy a megtalált él mindegy egyes képpontján végigmegy az algoritmus. Ha a Hough térben minden ugyanazon pontokból (x,y) készítünk r sugárral köröket, akkor egy olyan ábrát kapunk, ami a jobboldali képen látható.

Az eredeti kör középpontját úgy kapjuk meg, hogy egy összegzőtömbben nyilvántartja az algoritmus, hogy mely ponton megy át a legtöbb körív, és ahol ennek a tömbnek a maximuma van, ott lesz a kör középpontja.



4.ábra: kör detektálása a Hough térben, abban az esetben, ha ismert a keresett kör sugara

Megjegyzés: A jobb oldali ábrán láthatjuk, hogy a Hough térben a keresett középpontot ott találhatjuk meg, ahol a legtöbb kör metszi egymást. Az ábrán ennek a helye jól kivehető.

forrás: <https://www.youtube.com/watch?v=Ltqt24SQQoI>

A `cv2.HoughCircles` függvény használatakor nagyon fontos, hogy a `minRadius`, `maxRadius` paramétereket megfelelően állítsuk be, mert ha rosszul vannak ezek megadva, akkor számunkra felesleges -túl kicsi vagy túl nagy- köröket is találni fog a függvény.

Ha a keresett körök sugara nem ismert, akkor 3D-s Hough térben kell dolgoznunk. Ebben az esetben a módszer ugyanaz, de a sugár mindig növekedni fog.

1.4. Feladat megoldása, program felépítése

A feladat első felében a képen meg kell keresnünk a kör alakzatokat, amelyekről később eldöntjük, hogy milyen pénzérme található az adott pozícióban.

Ehhez a következő lépéseket használtam:

- **Kép átméretezése**

A legelső művelet a beolvasott képpel az volt, hogy átméreteztem. Erre azért van szükség, mert a mai technológiával már nagy felbontású képeket vagyunk képesek készíteni, de ennél a feladatnál nem kell, hogy a képek felbontása túl nagy legyen. Ha túl nagy a kép felbontása, az még akár hátráltathatja is a munkánkat, mivel sokkal részletesebb és ezáltal a feldolgozása lassabb lehet. A képet 1440 képpont szélesre méreteztem át.

- **Kép szürkeárnyaltossá alakítása**

Az OpenCV a színcsatornákat BGR sorrendben tárolja, ezt először átalakítom RGB sorrendre. Ezután, elkészítem a kép szürkeárnyaltos másolatát. Erre azért van szükség, mert a cv2.HoughCircles függvény egy szürkeárnyaltos képet vár paraméterként.



5.ábra: szürkeárnyaltossá alakított kép

- **Gauss szűrő alkalmazása**

A harmadik műveletben az átméretezett és szürkeárnyaltos képen Gauss-szűrőt alkalmazok. A körök megkeresésénél nem fontos a részletesség, az éleket szeretnénk detektálni.



6. ábra: Gauss-szűrő alkalmazása

- **Kör alakzatok keresése a képen, HoughCircles függvény segítségével**

A kör alakzatokat a feljebb már ismertetett Hough transzformáció segítségével fogjuk megkeresni. Erre az OpenCV-ben a cv.HoughCircles függvény van a segítségünkre. A függvény visszaad egy eredménytömböt, amelyben eltárolta a keresett körök középpontjainak x és y koordinátáit, és a sugár hosszát. ezt láthatjuk a konzolon kiírva.

```
      X      Y      R
[[ [ 472  758  147]
   [ 628  256  143]
   [ 342  474  131]
   [ 912  254  133]
  [1038  808  126]
   [ 930  516  147]
   [ 744  696  109]
   [ 608  512  109]]]
Megtaláltam 8 db penzermet!
```

7. ábra: A cv2.HoughCircles segítségével eltároltuk a körök pozícióját és sugarát

Miután megtaláltuk a pénzerméket (köröket) a képen, és ezeknek a pozícióját eltároltuk, ki kell elemeznünk, hogy a kép adott szeletében milyen típusú érme található.

- **Pénzérme típusának meghatározása**

Körök területének átlagos színintenzitása

A pénzérmék egyik meghatározó tulajdonsága, az összetételük, amelyek különböző színeket eredményeznek. Például a 10 forintos és az 50 forintos érme is 75% rézből és 25% Nikkelből épül fel. Ennek eredményeképpen mind a két érme ezüstös/ szürkés színű. A képeknek van egy olyan tulajdonságuk, hogy leírhatók mátrixok segítségével és minden egyes képpontot egy számhármassal tudunk jellemezni, amelyek megadják a színcsatornák értékeit. Ezt kihasználva a pénzérmék területét kivágtam a képből, majd ennek a területnek vettem az összes képpontját és ezek színintenzitását átlagoltam. Így megkaptam minden érmére egy értéket, amely jellemzi az adott pénzérmét. Ezzel a módszerrel jól el lehet különíteni a 10 Ft-os és az 50 Ft-os érméket a többitől, mert azok színei sötétebbek ezekhez viszonyítva.



8. ábra: 10 forintos érme területe a képen

A pénzérmék területének átlagos színintenzitását az alábbi függvénnyel számoltam ki:

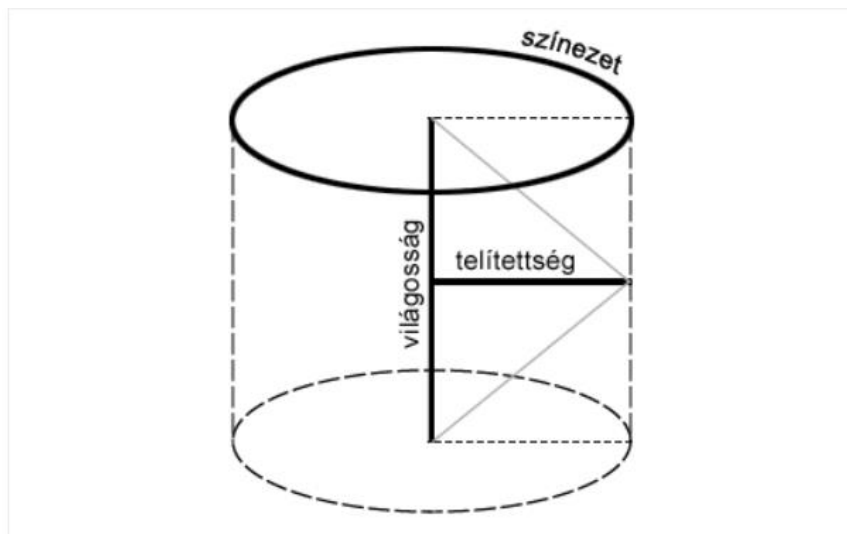
```
def averageIntensity(img, circles):
    av_values = []
    for coordinates in circles[0,:]:
        r=coordinates[2]
        coin= img[coordinates[1] - r:coordinates[1] + r, coordinates[0] - r:coordinates[0] + r]
        coin_hls = cv2.cvtColor(coin, cv2.COLOR_RGB2HLS) # másik színrendszer használata
        avg_row = np.average(coin_hls, axis=0)
        avg_hls = np.average(avg_row, axis=0)
        av_values.append(np.around(avg_hls[0])) # hue = avg_hls[0]
    print (av_values)
    return av_values
```

9. ábra: Kódrészlet: átlagos színintenzitás kiszámolása

A pénzérmék szín alapú kiválogatása RGB színtérben nem volt sikeres számomra, mert az átlagolás után nagyon hasonló értékeket kaptam, így például volt, hogy a 20 Ft-os érmét és a 10 Ft-os érmét összekeverte a program. Ez azért volt lehetséges, mert méretük és a kapott átlagos színintenzitásuk is hasonló volt. Erre a problémára azt a megoldást találtam, hogy a kivágott képrészletet át kellett konvertálni egy másik színtérbe. A színterek a színek ábrázolására használható virtuális térbeli koordináta-rendszer, ahol az egyes színek tulajdonságait azok koordinátái fejezik ki.¹ Az általam választott másik színtér a HLS lett végül. Ebben a színtérben az adott színeket ugyancsak 3 értékkel tudjuk leírni, de itt a 3 érték a következőkre utal:

¹ <https://hu.wikipedia.org/wiki/Színtér>

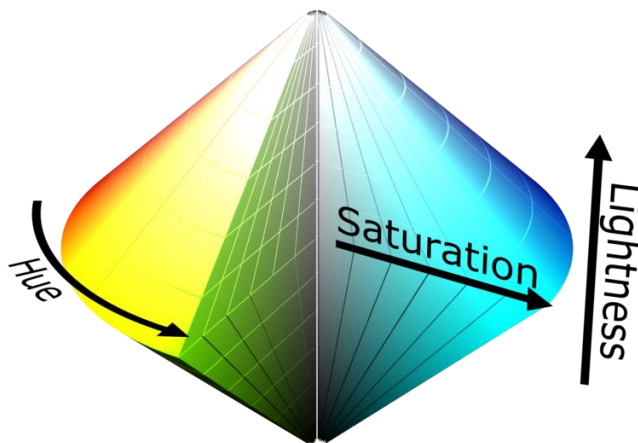
1.4.1 HSL színtér



10.ábra: Hengerkoordináta rendszer

forrás: http://www.szinkommunikacio.hu/12_05.htm

A HSL színtér (gyakran nevezik HSB színtérnek is) egy hengerkoordináta-rendszer, nagyon hasonló HSV színtér modelljéhez. Ezeket a modelleket nagyon gyakran használják képfeldolgozásnál. A HSL modell alakja kettőskúp.

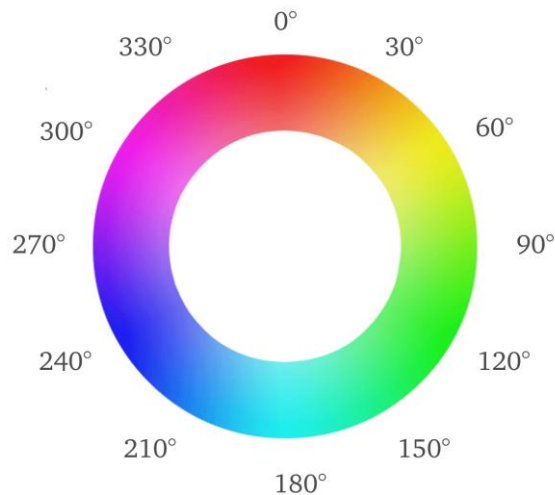


11.ábra: HSL modell: látható a kettőskúp alakzat

forrás: https://commons.wikimedia.org/wiki/File:HSL_color_solid_dblcone.png

Hue: színérték

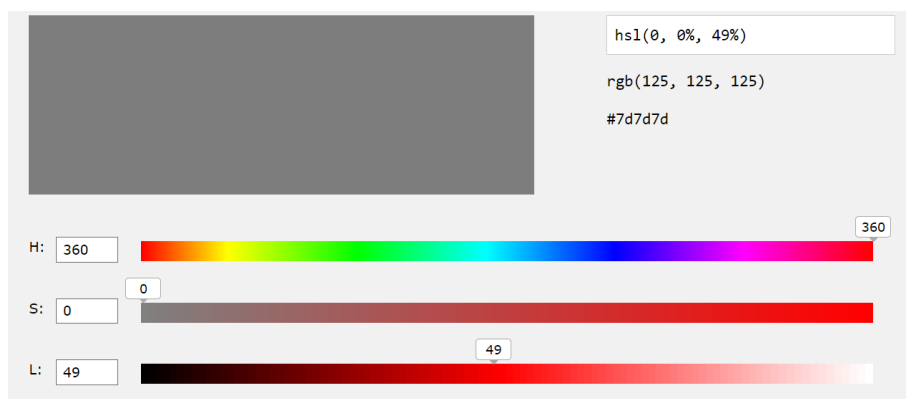
A HSL modellben az első paraméter a Hue, azaz a színérték. Ahogy azt a fenti ábrán is láthatjuk, a színértéket a henger főkörén fut körbe, minden színhez egy szöveget tudunk rendelni, és azzal megadni a kívánt színt. Általában a 0°-nál található a piros szín, 120°-nál a zöld, és 240°-nál a kék szín.



12. ábra: Színkör, színérték (hue) olvasható le erről az ábráról
forrás: <https://www.sarasoueidan.com/blog/hex-rgb-to-hsl/>

Saturation: telítettség

A telítettség értékét egy nullától százig tartó skálán adhatjuk meg. A telítettség sugárirányban változhat. A telítettség értéke mutatja meg, hogy egy árnyalat hol helyezkedik el a tiszta színezet és a (vele azonos világosságú) szürke között. Ha a telítettség 0, akkor a függőleges tengelyen van az érték, ez pedig a szürke színt fogja nekünk eredményezni, bármilyen értéket is adtunk meg a Hue-nál.



13. ábra: Láthatjuk, hogy a piros szín lett kiválasztva, de mivel a telítettség értéke = 0, ezért szürke színt kaptunk végül. A saturationt jelölő második csúszkán láthatjuk, hogy a piros szín intenzitása miként változik 0-tól 100-ig

forrás: https://www.w3schools.com/colors/colors_hsl.asp

Lightness: világosság (Lightness helyett gyakran a Brightness szót használják)

A világosság értékét a függőleges tengelyirányban adhatjuk meg, ahogy azt a 10. ábra is mutatja. A világosság értéke is 0-tól 100-ig terjed, sokszor százalékban adhatjuk meg. A következő ábrán láthatjuk, hogy az alsó pontja a tengelynek a teljesen fekete szín, a felső pontja pedig a teljesen fehér szín. Amennyiben a világosság értéke 0, nem számít a szín és a telítettség értéke, mindig feketét kapunk.

Kör sugara:

A pénzérték meghatározásához újra elő kell vennünk a megtalált körök sugarát, hiszen a másik fontos tényező, amely meghatároz egy pénzérték, az a mérete. Az 5 és 200 forintos érmék mérete kiugró a többi közül, hiszen az első a legkisebb érme, a második pedig a legnagyobb érme mind közül.

A többi érme esetében a méretet és az átlagos színintenzitást is figyelembe kell venni, csak így tudjuk meghatározni az érmék értékét. A 100 Ft-os, és a 200 Ft-os érmét jól el tudjuk különíteni a többitől. A 10 és az 50 Ft-os is jól elkülöníthető, de mivel a színintenzitásuk hasonló, a kettő között csak a méret alapján tudunk dönteni. Az 5 Ft-os, 20 Ft-os is csoportosítható, de itt is meghatározó a méret. A tesztelésnél láthatjuk majd, hogy a legtöbb hiba ebből adódik.

A pénzérték detektálásánál a következő módszert alkalmaztam:

Az átlagos színintenzitás alapján 4 csoportra osztottam az érméket:

90, vagy annál nagyobb érték: 10 Ft vagy 50 Ft A kettő közül a méret dönt.

78 és 90 közötti érték: 100 Ft

60 és 78 közötti érték: 200 Ft

60 alatti érték: 5 Ft, 20 Ft

A kettő közül a méret dönt.

A kódban ezt a feladatot a CoinAnalysis nevű függvény végzi.

Kiírás:

Miután meghatároztuk a pénzérték értékét, kiírjuk az adott érme fölé az értéket, és a kép aljára az összértéket. A kimeneti képet feljebb már bemutattam (2. ábra).

1.5. Tesztelés

A programot 25 általam készített képen teszteltem. Ennek eredményeit a következő ábra tartalmazza.

Kép neve	Talált érmék száma	5 Ft	10 Ft	20 Ft	50 Ft	100 Ft	200 Ft	Ellenőrzés	Valós érték a képen	Eredmény	Megjegyzés
456.jpg	2	0	0	0	1	1	0	150	150	Megfelelt	
545.jpg	10	2	2	2	1	1	2	620	620	Megfelelt	
555.jpg	7	1	2	2	0	1	1	365	365	Megfelelt	
602.jpg	6	2	1	2	0	0	1	260	260	Megfelelt	
642.jpg	2	0	0	0	0	0	1	250	210	Nem felelt meg	10 Ft helyett 50 Ft : Közeli kép
645.jpg	1	0	0	0	0	0	1	200	200	Megfelelt	
658.jpg	3	0	0	1	0	0	0	120	25	Nem felelt meg	10 Ft helyett 50 Ft, 5 Ft helyett 20 Ft: Közeli kép
710.jpg	1	0	0	0	0	0	1	200	100	Nem felelt meg	100 Ft helyett 200 Ft
735.jpg	3	0	0	1	1	0	1	270	270	Megfelelt	
741.jpg	3	0	0	1	0	0	2	420	420	Megfelelt	
748.jpg	2	0	0	0	0	0	2	400	400	Megfelelt	
805.jpg	2	0	0	0	0	1	1	300	300	Megfelelt	
621.jpg	4	0	0	1	1	0	2	470	455	Nem felelt meg	5 Ft helyett 20 Ft
626.jpg	4	1	0	0	1	1	1	355	355	Megfelelt	
633.jpg	4	0	0	0	0	1	1	400	360	Nem felelt meg	10 Ft helyett 50 Ft
653.jpg	2	0	0	1	1	0	0	70	15	Nem felelt meg	10 Ft helyett 50 Ft, 5 Ft helyett 20 Ft: Közeli kép
703.jpg	3	0	0	0	0	0	1	300	120	Nem felelt meg	100 Ft helyett 200 Ft, 10 Ft helyett 50 Ft
707.jpg	2	0	0	0	0	0	1	250	110	Nem felelt meg	100 Ft helyett 200 Ft, 10 Ft helyett 50 Ft
755.jpg	3	0	0	0	0	0	1	240	225	Nem felelt meg	5 Ft helyett 20 Ft : 5 Ft-os árnyéka miatt, közeli kép
613.jpg	6	1	0	2	1	0	2	495	495	Megfelelt	
637.jpg	3	0	0	0	0	1	1	350	310	Nem felelt meg	10 Ft helyett 50 Ft
715.jpg	2	0	0	0	1	0	1	250	150	Nem felelt meg	100 Ft helyett 200 Ft
723.jpg	4	0	0	0	1	0	2	450	350	Nem felelt meg	Plusz kör felismerése a képen, kisméretű sugárral
729.jpg	3	0	0	1	1	0	1	270	255	Nem felelt meg	5 Ft helyett 20 Ft
800.jpg	3	0	0	1	0	1	1	320	305	Nem felelt meg	5 Ft helyett 20 Ft

14. ábra: Tesztelés eredménye

A tesztelés során a 25 képből 11 lett teljesen sikeres. Ez 44%-os arányt jelent. Ha a helyesen felismert érméket nézem, akkor 84-ből 61 db-ot sikerült helyesen felismernie, ez pedig 73%-os arány.

Leggyakoribb hibák:

A képek készítésénél nagyon fontos, hogy milyen távolságból, és milyen fényviszonyok között készülnek. A teszt eredményeiből látható, hogy a képek készítésénél közelebbi képeket is készítettem, amely miatt a program sikeressége nagyban lecsökkent. Az általam beállított paraméterek az 545.jpg kép alapján történtek. Ezek a paraméterek nem voltak megfelelőek a közelebből készült képek esetén

A kép közelsége okozta hibák:

5 Ft helyett 20 Ft és 10 Ft helyett 50 Ft:

A kódban a két érme között csak a méretbeli különbség dönt, de a közelebbi a kép, akkor a kisebb érme (5 Ft, vagy 10 Ft) sugarának mérete megnő, ezáltal a nagyobb méretű érmét (20 Ft, 50 Ft) ismeri fel a program.

Fényviszonyok okozta hiba:

100 Ft helyett 200 Ft: A két érme elkülöníthető a többitől az átlagos színintenzitás alapján, de ha kicsit megváltoznak a fényviszonyok, akkor a kettőt összekeverheti a program.

Plusz kör felismerése a képen: A program esetlegesen találhat a képen még köralakzatokat, amelyeknek ad egy plusz forintértéket is.

1.6. Fejlesztési lehetőségek

A program jelenleg csak akkor használható, ha tudjuk, mekkora a képeken a pénzérmék mérete, és milyen a színintenzitása. Ezek nélkül csak a köralakzatokat tudjuk felismerni a képen, amelyek esetlegesen pénzérmék lehetnek. Tehát ez csak fix helyzetben működőképes, például ipari környezetben lehetne elképzelni.

A program fejlettebb változata a következő lenne:

Bármely képen felismerné a pénzérméket: nem számít, hogy milyen távolságból készült a kép, illetve az sem befolyásolná, hogy milyen fényviszonyok között készült.

A program továbbfejlesztése sok megoldandó feladatot hordoz magában. Lehetne próbálkozni a minta kereséssel, így talán megoldható lenne a függetlenítés a külső tényezőktől.

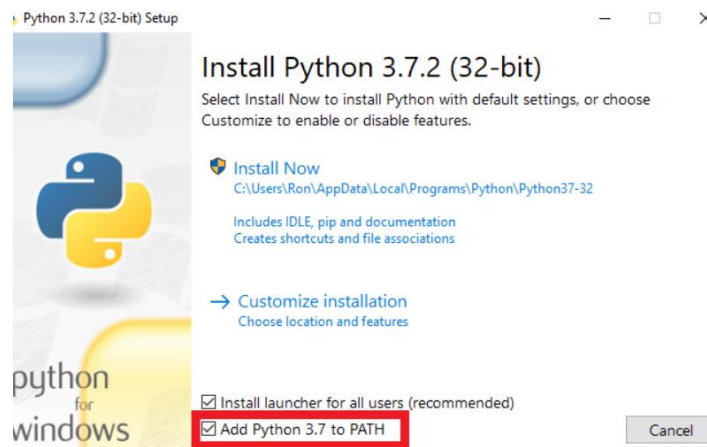
2) Felhasználói dokumentáció

Ebben a fejezetben ismertetem, hogy lehet működtetni az általam készített programot. A program futtatásához szükségünk lesz a Pythonra, illetve az ehhez tartozó csomagokra. Először azt mutatom be, hogy lehet futtatni a kódot parancssorból.

1.lépés:

Python letöltése és telepítése: <https://www.python.org/downloads/>

A telepítésnél figyeljünk arra, hogy a következő dolog kiválasztásra kerüljön, mert alapértelmezetten nem így van:



15. ábra: Python telepítésénél ki kell választanunk ezt az opciót

A telepítés sikerességét a következő módon ellenőrizhetjük:

Megnyitjuk a Parancssor alkalmazást, és kiadjuk a következő parancsot:

```
C:\Users\bogna>python --version
Python 3.9.0
```

16. ábra: Python verzió ellenőrzése Parancssor alkalmazásban

2. lépés: A futtatáshoz szükséges csomagok telepítése:

Le kell töltenünk az alábbi csomagokat: OpenCv, amely tartalmazza a NumPyt is, illetve a Matplotlib. Ezeket az alábbi parancsokkal tehetjük meg.

```
C:\Users\bogna>pip install opencv-python
Collecting opencv-python
C:\Users\bogna>pip install matplotlib
Collecting matplotlib
```

17. ábra: OpenCv és ezzel együtt a NumPy telepítése, majd a Matplotlib telepítése

3. lépés: Miután ezzel kész vagyunk, töltsük le a GitHubról a kódot, illetve a hozzá a teszt képet: https://github.com/milan674/penzerme_szamlalo

A helyesen futtatható kód a run.py elnevezésű. (Ez nem tartalmaz kommenteket)

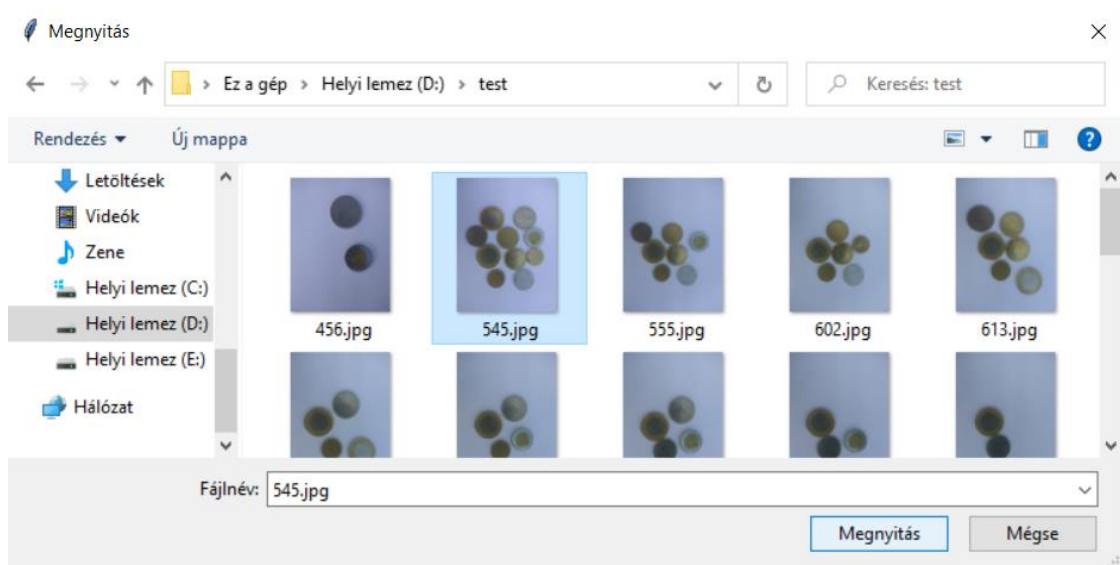
4.lépés: Fájl futtatása parancssorból: Nekem az egyszerűség kedvéért az Asztalon van a run.py fájl ebben az esetben. A parancssorban odanavigálok az első paranccsal, a második parancs pedig elindítja a programot.

```
C:\Users\bogna>cd Desktop  
C:\Users\bogna\Desktop>run.py
```

18. ábra: Program futtatása parancssorból

Ekkor feljön egy ablak, amelyben kiválaszthatjuk a tetszőleges képfájlt.

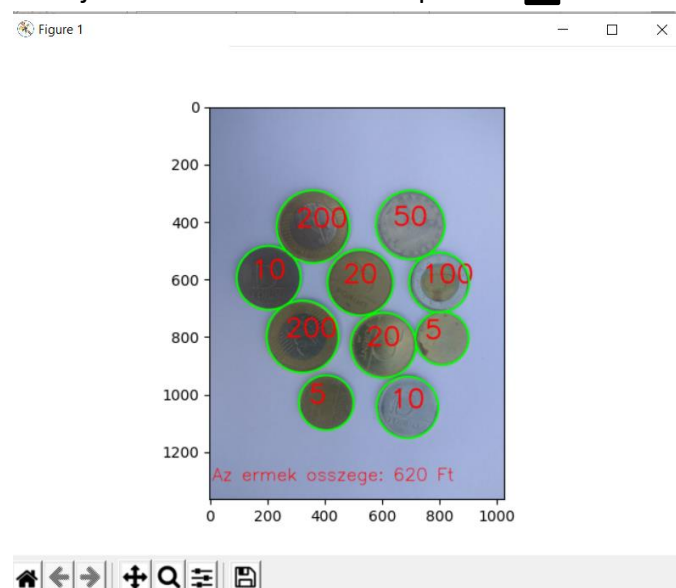
Megjegyzés: Arra figyeljünk, hogy az elérési út ne tartalmazzon ékezetes karaktert!



19. ábra: Kép elérési útjának kiválasztása

A fájl kiválasztása után a program lefut, és a kimeneti képet kell látnunk.

A mentés ikonnal el is tudjuk menteni a kimeneti képet.

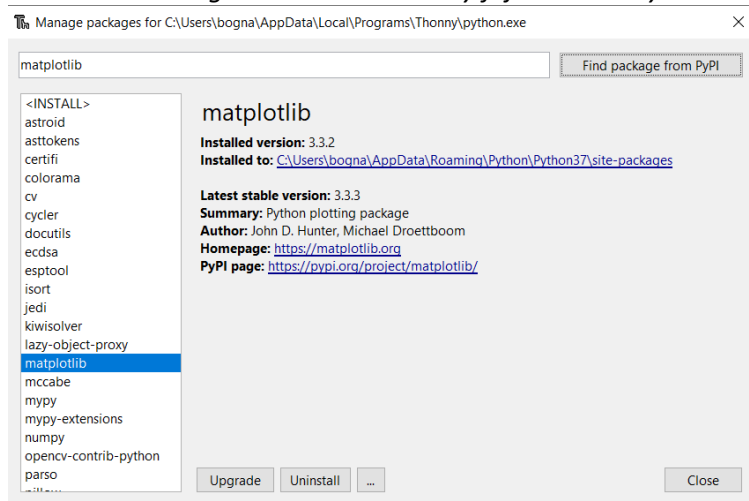


20. ábra: Kimeneti kép a program lefutása után

Másik módszer a futtatásra: Töltsünk le egy fejlesztői környezetet, például a Thonny nevűt. Letöltés: <https://thonny.org>

Telepítsük fel, és indítsuk el a run.py fájlt a Thonnyval. Ebben is telepítenünk kell a csomagokat, de ezzel könnyebb dolgunk van, mint a parancssorban. A címsorban felül kiválasztjuk a Tools menüt, és rákattintunk a Manage packages gombra. A következő ábrán a matplotlib csomag keresése látható. Ugyanezt kell tennünk az OpenCv-vel, és a NumPy-val is.

21.ábra: Csomagok letöltése a Thonny fejlesztői környezethez



Ha letöltöttük a csomagokat, akkor visszamegyünk a szerkesztő részre, és elindítjuk a programot az F5 gombbal, vagy a kis zöld ikonnal felül. A kimenet itt is ugyanaz lesz, mintha parancssorból indítottuk volna el.



21.ábra: Program indítása Thonny fejlesztői környezetből

3) Források

Hough-tér leírása:

https://hu.qaz.wiki/wiki/Circle_Hough_Transform

https://hu.qaz.wiki/wiki/Hough_transform

<https://www.youtube.com/watch?v=Ltqt24SQQoI>

Színterek leírása:

http://www.szinkommunikacio.hu/12_05.htm

<https://www.youtube.com/watch?v=Ceur-ARJ4Wc>

<https://www.sarasoueidan.com/blog/hex-rgb-to-hsl/>

https://www.w3schools.com/colors/colors_hsl.asp

<http://www.zmgzeg.sulinet.hu/ntk/inf9/graf/keg.pdf>

https://en.wikipedia.org/wiki/HSL_and_HSV#Lightness

<https://hu.wikipedia.org/wiki/Színtér>

<https://www.sarasoueidan.com/blog/hex-rgb-to-hsl/>