Beadandó Dokumentáció  
Pénzérme számláló alkalmazás

Gépi látás  
(GKLB\_INTM038)

Szandai Gábor Zsolt

A5R58R

Tartalomjegyzék

[1. Bevezetés, megoldandó feladat kifejtése 3](#_Toc41813413)

[2. Megoldáshoz szükséges elméleti háttér rövid ismertetése 4](#_Toc41813414)

[3. A megvalósítás terve és kivitelezése 8](#_Toc41813415)

[4. Tesztelés 11](#_Toc41813416)

[5. Felhasználói leírás 14](#_Toc41813417)

[6. Irodalomjegyzék 15](#_Toc41813418)

# Bevezetés, megoldandó feladat kifejtése

Pénzérme számláló alkalmazás:  
Előre definiált jellemzőkkel rendelkező pénzérméket tartalmazó fényképről megállapítani, hogy mennyi a képen látható érmék összes értéke.

A MNB honlapján megtalálhatók a magyar érmék fő paraméterei.   
Az érmék felismeréséhez szükséges paraméterek lehetnek az átmérő a szín, illetve oldalirányból a peremezés.

. táblázat A magyar érmék és paraméterei

|  |  |  |
| --- | --- | --- |
|  |  | Átmérő: 28,3 mm Szín: a külső gyűrű aranysárga, a belső rész ezüstfehér színű  Pereme: szaggatottan recés (receszám: 72) |
|  |  | Átmérő: 23,8 mm Szín: a külső gyűrű ezüstfehér, a belső rész aranysárga színű  Pereme: recés (receszám: 170) |
|  |  | Átmérő: 27,4 mm Szín: ezüstfehér  Pereme: sima |
|  |  | Átmérő: 26,3 mm Szín: aranysárga  Pereme: recés (receszám: 130) |
|  |  | Átmérő: 24,8 mm Szín: ezüstfehér  Pereme: szaggatottan recés (receszám: 70) |
|  |  | Átmérő: 21,2 mm Szín: aranysárga  Pereme: sima |

# Megoldáshoz szükséges elméleti háttér rövid ismertetése

Az elméleti háttér főforrása az internet, abból is leginkább az OpenCV dokumentációs oldala, célja a gyakorlati részben alkalmazott legfontosabb módszerek hátterének kifejtése.

**OpenCV:**Egy kiegészítő könyvtár, ami képfeldolgozó funkciókat tartalmaz.

**Kép:**A digitális kép két fő részre bontható, az egyik a fej/fejléc, ami tartalmazza a kép formátumát, palettáját, méretét, tehát a képet általánosságban leíró jellemzőket. A másik az adatrész, ami az egyes képpont adatait tartalmazza mátrix formában.

Számunkra a képek egyik fontos jellemzője az adattípus:

<bit-depth>{U|S|F}C(<number\_of\_channels>)

U: unsigned integer típus

S: signed integer típus

F: float típus

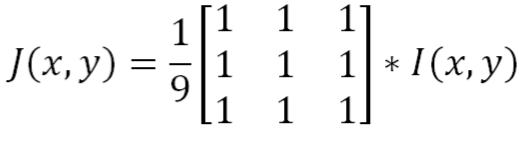
Például: 8UC3 egy 8-bites unsigned integer matrix 3 csatornával.

A csatornák a képpontok különböző információit tárolják, például egy RGB képnél a három csatorna a vörös (red) a zöld (green) és a kék (blue) szín képpontokként felvett értékét tárolja.   
A szürkeárnyalatos képek csak egy csatornával rendelkeznek.

**Színtartomány konvertálás:**Színes képet képesek vagyunk egy színárnyalatosra (szürkévé) konvertálni, ilyenkor az RGB/BGR során alkalmazott 3 csatorna egy csatorna lesz, és pixelekként átszámolja ennek az egy csatornának az értékét. pl.: RGB -> Gray: Y = 0.299\*R+0.587\*G+0.114\*B.

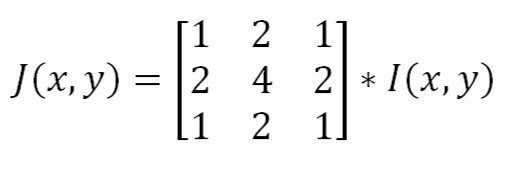
**Blur:**Élsimítással/szűréssel, a célunk a zajok megszűntetése azonban a szűrés mértékével csökken a kép élessége is. A szűrésre több kidolgozott módszer is a rendelkezésünkre áll:

*Átlagoló:*Átlagolja a kernel méretében (3\*3 - egyes rádiusz) lévő képpontokat. Leggyakoribb alak:



. egyenlet Átlagoló szűrés

*Gaussian Blur:*Gauss szűrőt használva simítja a képet. A közelebbi szomszédok nagyobb súllyal szerepelnek, a szórás adja meg a súlyozás csökkenésének gyorsaságát a távolság függvényében. A kernel mérete pozitív egész páratlan szám. Leggyakoribb alak:



. egyenlet Gaussian szűrő

*Medián szűrő:*Veszi a kernel területén lévő képpontok mediánját és a kernel középső elemének értéke a mediánnal lesz egyenlő. A kernel mérete itt is pozitív egész páratlan szám.

**Éldetektálás:**Az élek detektálása a képfeldolgozás egy központi kérdésköre, célja az objektumok beazonosítása, azok határvonalának meghatározásával. Alapja a feltevés, hogy ott van él, ahol az egyes képpontok között nagy az intenzitáskülönbség.   
Éleket az első vagy a másodrendű deriváltak számításával tudunk keresni. Mivel az éldetektálás nagyon érzékeny a zajokra, ezért általában első lépésként a képen szűrést hajtunk végre.

*Canny*1986ban fejlesztette ki John F. Canny. Az eljárás eredményeként egy bináris kép keletkezik, a képpontok tartalmazzák az információt, hogy az adott pont él vagy sem.

Lépései:

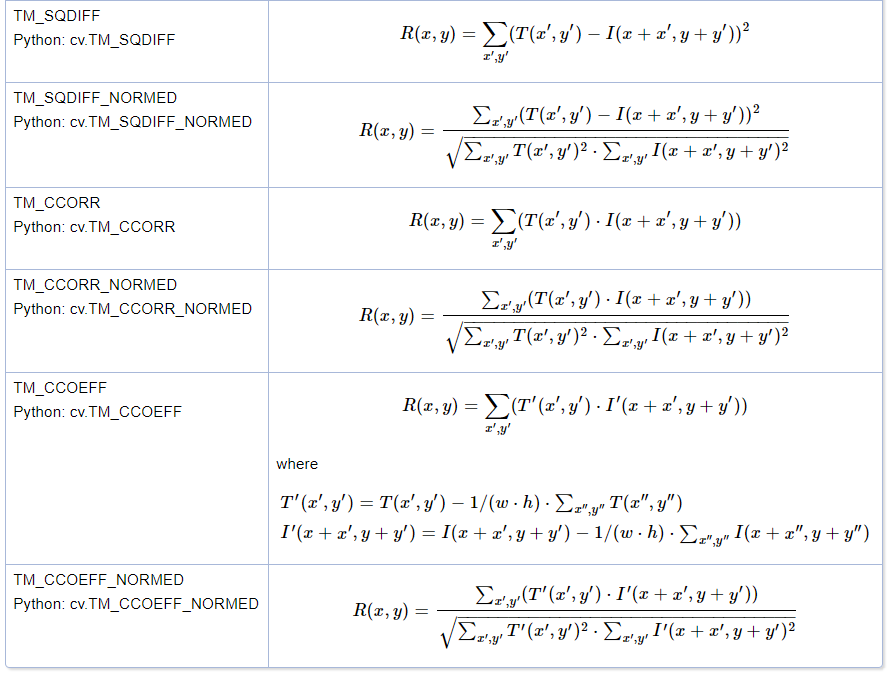
* Gauss simítás
* Differencia számítás
* Nem-maximum vágás
* Kettős küszöbölés
* Hiszterézis küszöbölés

**Hough kör transzformcáió:**Körök detektálására alkalmas eljárás. A kör detektálásának első lépése az éldetektálás (Canny). Megállapíthatjuk vele a kör középponti koordinátáit (x,y) és a kör rádiuszát (r).

***Mintaillesztés (Template Matching):***A mintaillesztés egy olyan eljárás, aminek célja megtalálni egy mintaképen szereplő információt/objektumot egy nagyobb képen. Ennek érdekében a mintaképet többször ráilleszti a képre, így „végig csúsztatja” a mintát és összehasonlítja a képen, a mintaképpel azonos méretű területtel. Az eredménye egy szürkeárnyalatos kép, ami megmutatja, hogy melyik pixel mennyire hasonlít az adott pixel a mintához. Fajtái:

* Square Difference Matching Metchod (SQDIFF) és NORMED
* Correlation Matching Method (CCORR) és NORMED
* Correlation Coefficient Matching Method (CCOEFF) és NORMED

. táblázat Templatematching képletei



Az SQDIFF során a végeredmény a képek közötti különbség, ezért a minél kisebb eredmény mutatja a nagyobb hasonlóságat, 0 jelenti a tökéletes egyezést. Az „SQDIFF\_Normed” értéke 0-1 között mozoghat, a 0 itt is a tökéletes egyezést jelenti.

A CCORR/CCOEFF módszer során a képek közötti hasonlóságot mutatja, így a nagyobb szám jelent jobb egyezést. „Normed” során a CCORR 0 és 1 között mozog, 1 jelenti a tökéletes egyezést, míg a CCOEFF -1 és 1 között.

**ORB:**A képeket geometriai információk alapján hasonlítjuk össze. Ehhez fontos két fogalmat tisztázni az egyik a kulcspont, a másik a leíró.

Kulcspont: adott régió nevezetes pontja, pl. sarok, vagy mintázat

Leíró: a kulcspont lokális környezetét leíró, adott hosszúságú vektor

Az ORB egy a sok fajta kulcspont detektáló eljárások közül. FAST eljárást alkalmazza piramis alakzatban a kulcspontok detektálására, kiválasztja a legerősebbeket FAST vagy Harris pontozás alapján, megkeresi az orientációjukat és kikalkulálja a leírót BRIEF eljárást alkalmazva.

Több lépésből álló művelet:

1. ORB létrehozása  
   Paraméterei:   
   nfeature: kulcspontok maximális száma

scaleFactor: piramis szintjeinek méreti aránya  
nlevels: piramis szintjeinek száma,

edgeThreshold: határérték kulcspont detektálásra,

firstlevel: forrás kép a piramis melyik szintjén legyen

WTA\_K: BRIEF leíró pontjainak a száma

score\_Type: pontozási rendszer Harris vagy Fast

patchSize: feature által letakart terület

fastTreshold: fast határa

1. Lépés: Kulcspontok és Descriptorok meghatározása, a képen és a mintaképen is  
   detectAndCompute parancsal.   
   Paraméterei: bemeneti kép; maszk; kulcspontokat tároló vektor; leírókat tároló kép.
2. Lépés: Descriptorok összehasonlítása  
   Típusai: BrutForce (BF); BF\_HAMMING; BF\_HAMMINGLUT; BF\_L1;BF\_SL2.

# A megvalósítás terve és kivitelezése

Fejlesztői környezetnek a visual studio 2017-et választottam, kiegészítve az OpenCV könyvtárral.  
**Mintaképek elkészítése:**

A program nem igényli, hogy különálló mintaképek már rendelkezésre álljanak, amennyiben az „Image1.jpg” nevű kép, egymás mellett növekvő sorrendben tartalmazza mind a 6 érmét, akkor a program saját maga képes elkészíteni a mintaképeket. A mintaképek ilyenkor mentésre kerülnek, a programmal azonos mappában, a következő alkalommal felhasználhatók vagy újak is készíthetőek.

A program három fő lépésből fut le:

* Előkészület
* Érme azonosítás
* Kiértékelés

**Előkészület:**

Az előkészület során a program megkeresi a képen látható köröket:

1. Átméretezés: input két átméretezése az eredeti méretarányok megtartása mellett egy kb. 500\*500-as méretre. Ezt azért tartom szükségesnek, hogy a különböző felbontású input képek is nagyjából hasonlóan viselkedjenek a beállított paraméterekkel.
2. Színtranszformáció: az átméretezett képet szürkeárnyalatossá tesszük
3. Szűrés: szűrést hajtunk végre, Gaussian szűrővel, Kernel mérete 9\*9.
4. Körkeresés: HoughCircles paranccsal végrehajtjuk a Hough kör transzformációt, majd a kapott körök központi koordinátáit, illetve sugarát mentjük egy vektorba.

**Érme Azonosítás:**

A feladat megoldására három módszert választottam ki:

* Színcsatornás összehasonlítás
* Mintaillesztés
* ORB

A megvalósítás során ez kiegészült az ORB + Színcsatornás vizsgálat kombinációjával, mint negyedik különálló módszer.

A módszerekben azonos, hogy mindegyik esetben az előkészületek során megtalált körökön egy „for” ciklussal megyünk végig és azon belül is egy „for” ciklussal pörgetjük végig az érméket. Így minden egyes kört hasonlítunk minden egyes érmével. Végeredményként megkapjuk „D” értékét, ami jelenti számunkra az eltérést/egyezőséget a talált kör és az érme között.

*Színcsatornás összehasonlítás:*Végig megyünk a képen sorok és oszlopokként és minden egyes pixelnél megvizsgáljuk, hogy mekkora a kör és a minta között az egyes színcsatornákban tárolt értékek különbsége. Minél alacsonyabb annál inkább hasonlítanak, így a 0 jelenti a tökéletes egyezést.

for (int y = 0; y < kor.rows; y++)

for (int x = 0; x < kor.cols; x++)   
db = abs(korforgatott.at<Vec3b>(y, x)[0] - tempresized.at<Vec3b>(y, x)[0]);  
dg = abs(korforgatott.at<Vec3b>(y, x)[1] - tempresized.at<Vec3b>(y, x)[1]);  
dr = abs(korforgatott.at<Vec3b>(y, x)[2] - tempresized.at<Vec3b>(y, x)[2]);  
d = d + db + dg + dr;

*Mintaillesztés:*

A mintaillesztés során a képre ráhelyezzük a mintaképet és vizsgáljuk a két kép közötti különbségeket. Az OpenCV matchTemplate parancsával ezt megtudjuk tenni.  
Paraméterei: Kép, Mintakép, Kimenetikép, Módszer  
A programmal végig fogjuk próbálni a mintaillesztés mind a 6 változatát.  
Előre feltételezzük, hogy a módszer az orientációra kifejezetten érzékeny.

*ORB:*

Az előző fejezetben elég részletesen kifejtettem az ORB algoritmus lépéseit, a gyakorlatba is ezt ültetem át. Mindenképp érdemes lesz megvizsgálni az ORB objektum paramétereit, illetve a leíró párosítás különböző típusokra adott eredményét.

*ORB+Szín:*

Az ORB esetén minél nagyobb szám jelenti a jobb egyezést, míg a színcsatornás vizsgálat esetén a minél kisebb jelenti a nagyobb egyezést. Az eljárás alapötlete, hogy kisebb számot osztva egy nagyobbal kisebbet kapunk eredményül, így lehetne párosítani a két eljárást.

**Kiértékelés:**

A kiértékelés nem más, mint az azonosított érmék feliratozása a képen, és az érmék összegének kiíratása a kép bal alsó sarkában, majd a kép mentése a programmal azonos mappában, egymás után 1-10-ig sorszámban.

**Kiegészítő lehetőségek:**

A képek között biztos, hogy különbség van az érme fényessége, mérete és orientációja között, ezért a következő módszereket alkalmaztam, hogy ezeket a különbségeket csökkentsem:

*Fényerő csökkentése:*

Az átméretezett kép minden csatornájának az értékét csökkenti 30%-al, az így kapott kép sötétebb, célja, hogy a csillogó felületek által keletkezett eltérést minimalizálja. Amennyiben a képen, akkor a mintaképeken is alkalmaztam a sötétítést.

double alpha = 0.3;Mat new\_image = Mat::zeros(atmeretezett.size(), atmeretezett.type());

for (int y = 0; y < atmeretezett.rows; y++) {  
for (int x = 0; x < atmeretezett.cols; x++) {for (int c = 0; c < atmeretezett.channels(); c++) {  
new\_image.at<Vec3b>(y, x)[c] = saturate\_cast<uchar>(alpha\*atmeretezett.at<Vec3b>(y, x)[c]);

A sötétítés könnyedén kikapcsolható, amennyiben alpha értékét 1-re állítjuk.

*Átméretezés:*

GaussianBlur(kor, kor, Size(3, 3), 2, 2);

Hogy az átméretezés okozta zajokat minimalizáljam szűrőt állítottam be.

*Orientáció:*

Amennyiben az eltérést az okozná, hogy az érmék elforogtak a mintaképen lévő állapothoz képest akkor forgassuk vissza.

for (angle = -360; angle <= 0; angle += 10) { Mat r = getRotationMatrix2D(Point(kor.rows / 2, kor.cols / 2), angle, 1.0) warpAffine(kor, korforgatott, r, Size(2 \* (radius), 2 \* (radius)));

A for ciklus 0 és 360 között körbeforgatja a képet a megadott lépték szerint, az elforgatott kép mentődik és az összehasonlítást alapja az elforgatott kép.

A forgatás könnyedén kikapcsolható amennyiben a kezdő és a végértéket azonosra vesszük.

# Tesztelés

A teszteléshez 10 db képet használtam. A mintaképek minden esetben az egyes számú képről kerültek kivágásra.

*Előkészület:*   
a körök megtalálásának legnagyobb hátráltató tényezője az árnyékok voltak, illetve a 100 és 200-as belső körei. Megfelelő paraméterbeállításokkal azonban sikerült megtalálni azt a beállítást, amivel már nagy pontossággal meg tudta határozni az érmék körvonalát.   
Mivel a 100 és 200 Ft-os érme belseje is tartalmaz egy extra kört, így kénytelen voltam megadni egy minimum átmérőt, ezzel pedig már be tudtam állítani úgy a határértékeket, hogy sem ezek a belső körök, sem az árnyékok nem befolyásolták jelentős mértékben az érmék körvonalának felismerését.  
A legnagyobb érdeme a módszernek, hogy a 9. és 10. képen szereplő átfedésben lévő érmék körvonalát is egész pontosan felmérte.

*Érmék felismerése:*Az összes képet tekintve összesen 34 db érmét próbáltam beazonosítani négy fajta módszerrel. A kiértékelés során 3 kategóriába soroltam a találatot:

* Zöld: az érmét sikeresen azonosította
* Sárga: nem sikerült a pontos azonosítás, de az érme színe legalább hasonló (párokat alkotva 5-20 / 10 – 50 / és a 100 – 200 Ft-os érmékkel)
* Piros: egyáltalán nem sikerült azonosítani.

*Színcsatornás:*

1. ábra Színcsatornás összehasonlítás eredménye

A színcsatornás összehasonlítás esetén próbáltam a fényerőt lecsökkenteni 30%-ra, remélve, hogy közelebb jutok a megoldáshoz, és mivel harmadoltam az egyes képpontok értékét így természetesen „D” értéke is csökkent, de az érmék azonosítása nem lett pontosabb.  
Megpróbáltam az érméket forgatni, de pontosabb eredményt értem el, ha nem forgattam az érméket.   
Az átméretezések után, mind a mintakép, mind a képből származó képet szűrtem a zajok megszűntetésének érdekében, sajnos javulást ez sem eredményezett.  
Továbbá megpróbáltam a mintakép és a talált érme közötti méretkülönbségre egy megfelelő korrekciós tényezőt találni, de sajnos szintén sikertelenül.

*Templatmatching:*

2. ábra Templatematching eredménye

TM\_CCOEFF és a TM\_CCOEFF\_NORMED : teljesen eredménytelen, mindent 100-asnak vagy 200-asnak érzékel.

TM\_CCORR és a TM\_CCORR\_NORMED szintén mindent 100-asnak vagy 200-asnak érzékel.

TM\_SQDIFF és a TM\_SQDIFF\_Normed közel azonos eredményt produkáltak és az érméknek több mint a felét sikeresen azonosították. Ennél a módszernél azt vártam, hogy erősen orientáció függő és ez meg is mutatkozik majd, azonban a forgatással és forgatás nélkül is ugyanannyi érmét sikerült azonosítani.   
*ORB / ORB + Szín:*

3. ábra ORB módszer eredménye

Az ORB eljárás csak az esetek 29%-ban tudta rendesen eltalálni, hogy melyik érme látható a képen, a szín eltérésből származó adattal kombinálva ez a szám 41%-ra tudott javulni.

*Összegzés:*

Mindenképpen sikernek könyvelhető el, hogy az eredeti képen helyesen ismerte fel az érméket. A további képek során befolyásoló tényezők lehetnek az orientáció, a méretkülönbség, a fényességből adódó különbség. A korrekciós eljárások ellenére sajnos nem sikerült egyik módszert sem elég robusztussá tenni az érmék felismerése tekintetében. A legjobb eredményt a minta illesztéssel sikerült elérni, de azt sem találtam elég robusztusnak.

Véleményem szerint az érmék között nagyon kicsi a különbség, amit sok tényező erősen befolyásol, ezért ezek a módszerek nem igazán alkalmasak arra, hogy nagy biztonsággal felismerjék az érmékét.

További vizsgálat ként a HaarCascade és a gépi tanulás irányába mennék el. Talán megfelelő mennyiségű és minőségű képpel az érmékről lehetséges lenne az érmék megkülönböztetése több képen is.

# Felhasználói leírás

A program a .exe fileal indítható.  
*Forrásképek:*A képek, amelyeken az érméket szeretnék megkeresni Image1, Image2 … Image10 néven kell a .exe fileal azonos mappában szerepeljenek. A program futtatásához 10 képre van szükségünk.

*Mintaképek:*

A mintaképeket létrehozhatjuk manuálisán, illetve rábízhatjuk, a programra is, hogy automatikusan hozza létre.

*Manuális:*Ez esetben szüksége lesz a felhasználónak minden érméről egy képre, ezt megteheti úgy, hogy saját maga készít minden egyes érméről külön-külön egy-egy képet, vagy kivágja az érméket egy képről. Mentsük a mintaképeket .jpg kiterjesztéssel a program mappájába, a mintaképek neve pedig legyen:

* T5 – 5 forintos
* T10 – 10 forintos
* T20 – 20 forintos
* T50 – 50 forintos
* T100 – 100 forintos
* T200 – 100 forintos

Fontos, hogy az Image1 ilyenkor ne tartalmazzon pontosan 6 db érmét, mert így a mintaképek felülírásra kerülnének.

*Automatikus:*Amennyiben az automatikus létrehozás mellett döntünk, nincs más dolgunk, mint az Image1-nek olyan képet elmenteni, amin pontosan 6 db érme található, és az érmék balról jobbra növekvő sorrendben helyezkednek el egymás mellett. Ilyenkor a program elkészíteni helyettünk a fenti nevekkel a mintaképeket.  
*Kimenet:*A program végezetül 1-10-ig sorszámozva menti a képeket, amin megjelenik az egyes érmék értéke, illetve a kép bal alsó sarkában a képen szereplő érmék értékének összege.

# Irodalomjegyzék

1. <https://docs.opencv.org/3.4/index.html>
2. [http://www.inf.u-szeged.hu/~tanacs/pyocv/index.html#](http://www.inf.u-szeged.hu/~tanacs/pyocv/index.html)
3. <https://www.mnb.hu/bankjegy-es-erme/ermeink>
4. Tantárgy előadás fóliái