Mozgásvizsgálat fényképek alapján

OpenCV könyvtár
Python
Octave

Segédanyagok

OpenCV
https://docs.opencv.org/3.4/
OpenCV
https://www.programcreek.com/python/example/89361/cv2.Canny
OpenCV
http://www.bmva.org/bmvc/1989/avc-89-029.pdf
NumPy
https://github.com/OSGeoLabBp/tutorials/blob/master/hungarian/python/numpy.rst
Mintapéldák
https://github.com/OSGeoLabBp/tutorials/tree/master/english/img_processing/code

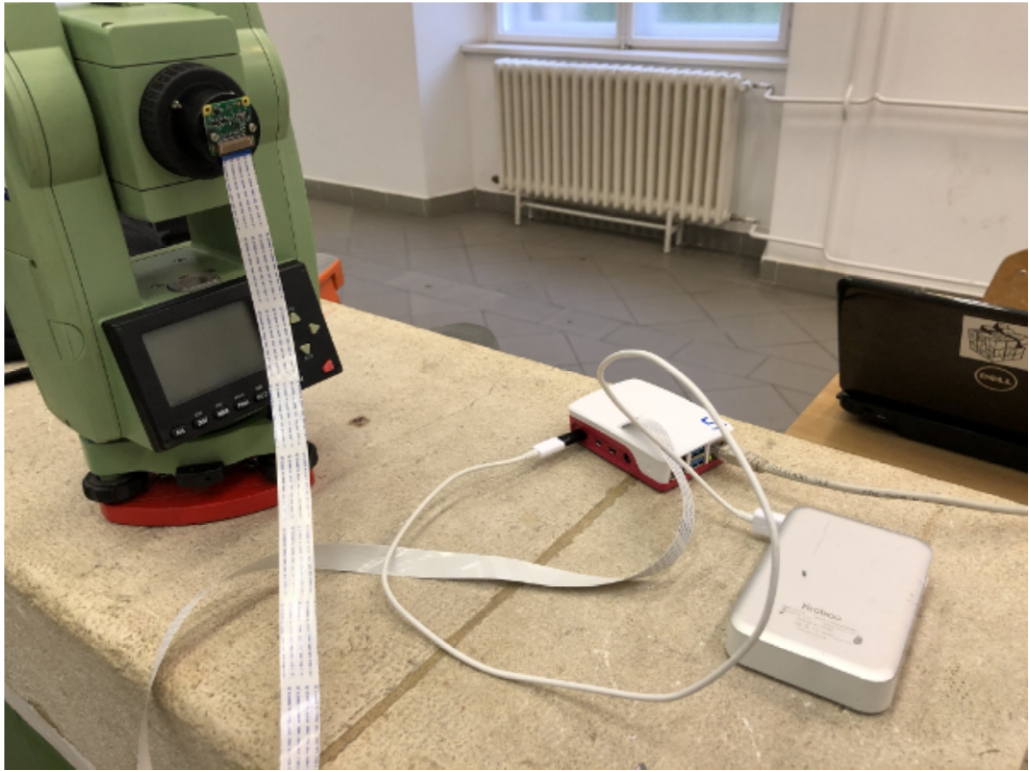
Alapelvek

Mozdulatlan fényképezőgéppel
készítünk felvételeket a mozgó tárgyról

A képek felbontását távcső
közbeiktatásával növelhetjük

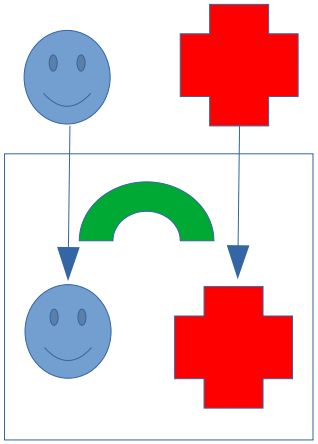
Kamera kalibráció

Megoldások
template matching –
képsíkkal párhuzamos mozgás kimutatása
pattern recognition –
elmozdulás és elfordulás



Mintakeresési módszerek

Mintaillesztés (template matching)



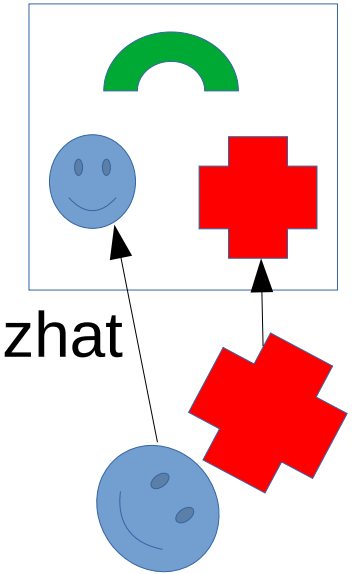
+

Mindig van találat
Egyszerű algoritmus
Nem kell speciális jelet feltenni

-

Hamis találat esélye nagyobb
Csak minimális elfordulás
Csak minimális méret változás

Alakfelismerés (pattern recognition)



+

Minta elfordulhat
Minta mérete megváltozhat
Irány becsülhető (3D)

-

Fényviszonyokra érzékenyebb
Speciális jelet kell elhelyezni

Template matching

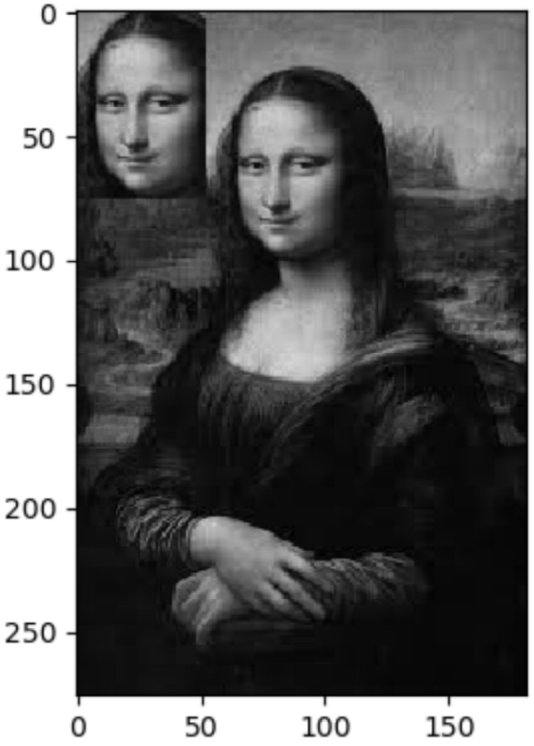
Egyik képben keressük meg egy másik kép legvalószínűbb előfordulását

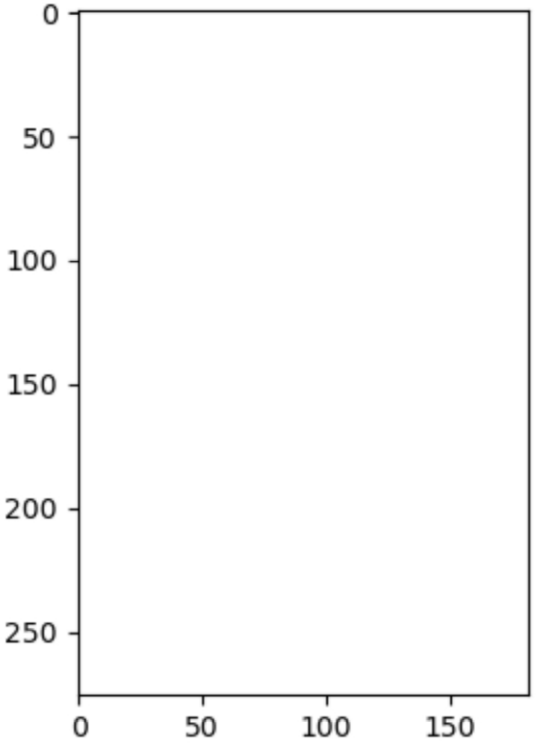
Vizsgált kép

Keresett minta

Mozgassuk a mintát a képen és
számoljunk statisztikát (az egymást fedő pixelek értékéből)
A hatékonyság érdekében konvertáljuk szürke árnyalatossá a képet

Mintaillesztés (template matching)



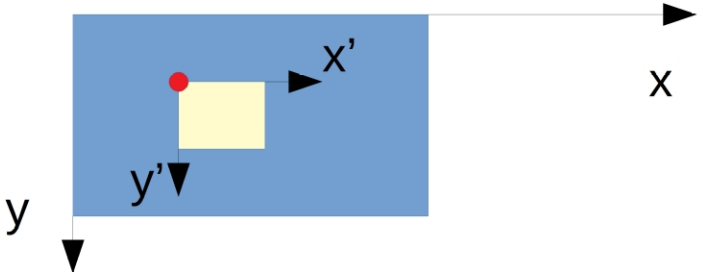


$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \quad \text{min}$$

$$= \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad \text{min}$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')) \quad \text{max}$$

$$= \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad \text{max}$$



Megvalósítás Python-ban

```
def img_correlation(img, templ):
    """ find most similar part to templ on img
         returns upper left corner of templ in img and statistic
    """
    rows, cols = img.shape
    trows, tcols = templ.shape
    row = col = None
    mins = trows * tcols * 255**2
    t = self.templ.astype(int)
    for i in range(rows - trows):
        i1 = i + trows
        for j in range(cols - tcols):
            j1 = j + tcols
            s = np.sum(np.square(t - img[i:i1, j:j1]))
            if s < mins:
                mins = s
                row = i
                col = j
    return (col, row, s)
```

Főprogram:

```
import cv2
import numpy as np
import sys

img = cv2.imread(sys.argv[1], cv2.IMREAD_GRAYSCALE)
templ = cv2.imread(sys.argv[2], cv2.IMREAD_GRAYSCALE)
print(img_correlation(img, templ))
```

Megvalósítás Octave-ban

```
#! /bin/octave
% image template matching minimizing absolute difference
pkg load image
args = argv();          % command line params into a cell array
if nargin > 1
  tnam = args{1};       % first parameter the template image
  inam = args{2};       % second parameter the image
else
  tnam = "mona_temp4.png";
  inam = "monalisa.jpg";
end
img = rgb2gray(imread(inam));   % read and convert to grayscale
temp = rgb2gray(imread(tnam));  % read and convert to grayscale
[n, m] = size(img);
[nt, mt] = size(temp);
min_dif = nt * mt * 255;
pos = [0, 0, min_dif];
for i = 1:n - nt
  for j = 1: m - mt
    dif = sum(sum(abs(img(i:i+nt-1, j:j+mt-1) - temp)));
    if dif < min_dif
      pos = [i, j, dif];
      min_dif = dif;
    end
  end
end
printf("%d %d %d\n", pos(1), pos(2), pos(3));
```

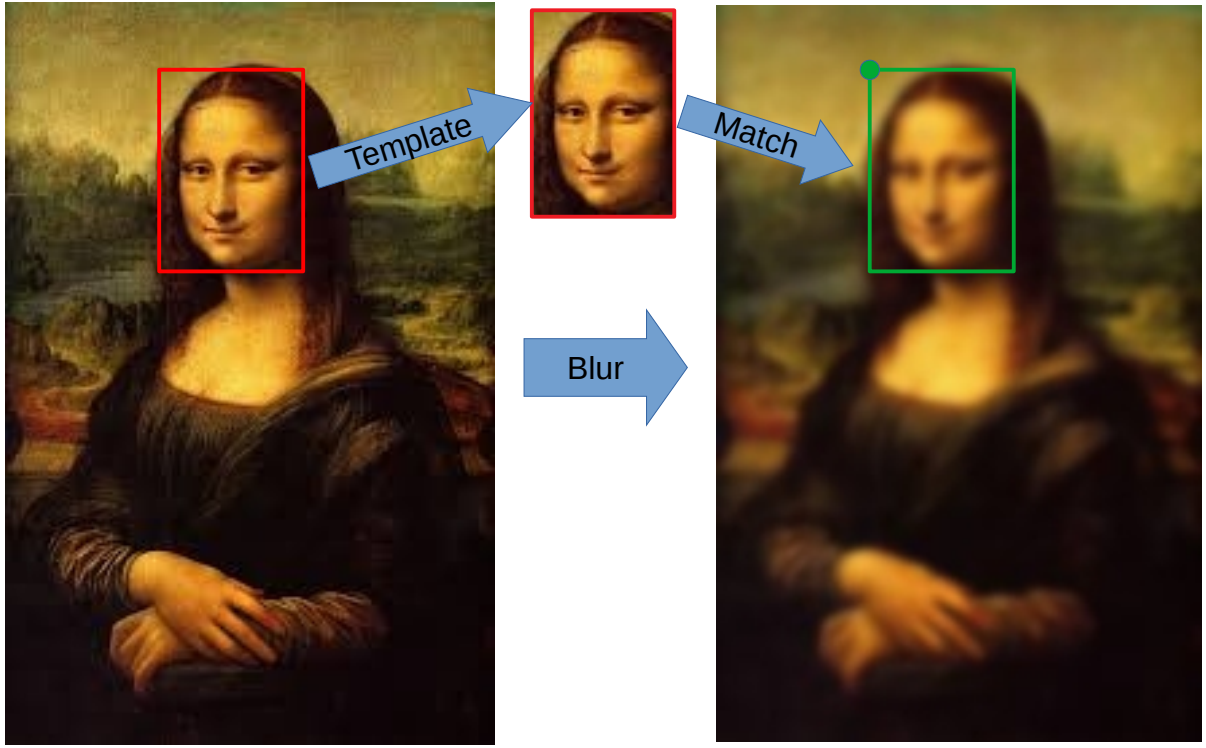
template_match.m
Statisztika:

$$\sum_{x',y'} |T(x',y') - I(x+x',y+y')|$$

Futtatás

Használat a parancssorból:

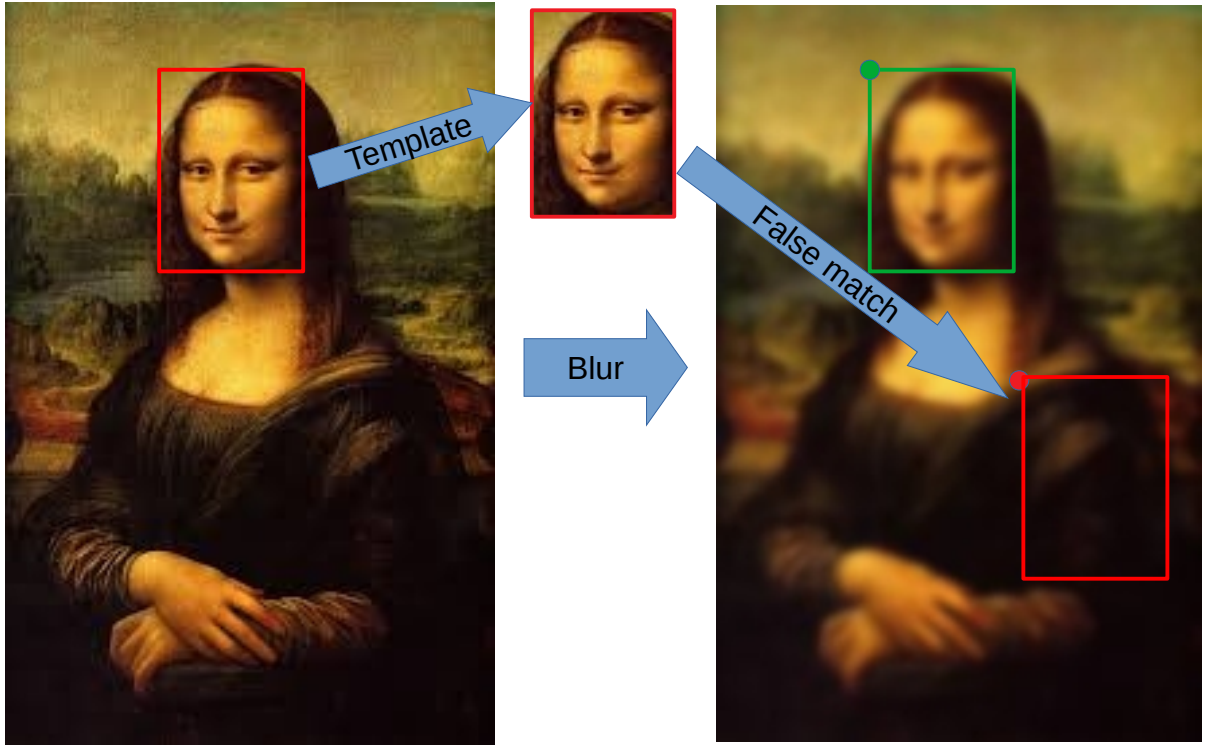
```
python3 img_corr.py monalisa.jpg mona_temp4.png
(57, 25, 39491287)
python3 img_corr.py monalisa_blur.jpg mona_temp4.png
(57, 25, 39483412)
```



Megvalósítás Octave-ban

Használat a parancssorból:

```
octave template_match.m mona_temp4.png monalisa.jpg
26 58 0
octave template_match.m mona_temp4.png monalisa_blur.jpg
142 107 183 !!!!! (26 58 21852) ????
```



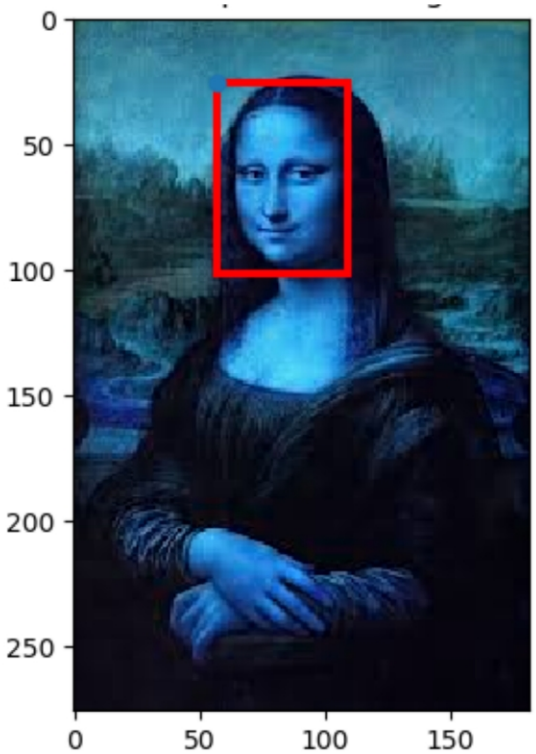
Megvalósítás Python, OpenCV

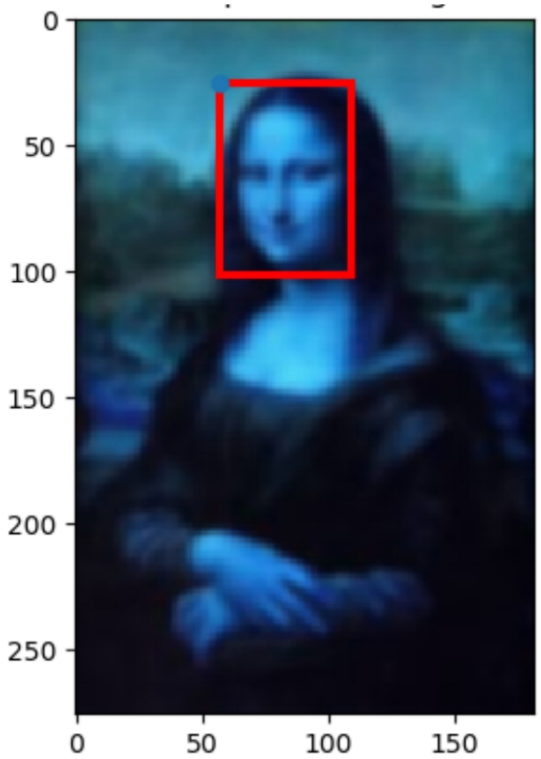
```
"""
    find template in image
"""
import cv2
import sys
import matplotlib.pyplot as plt
import matplotlib.patches as patches

if __name__ == "__main__":
    # selected method
    minv = 2    # index of min value
    maxv = 3    # index of max value
    methods = ((cv2.TM_SQDIFF_NORMED, minv), (cv2.TM_CCORR_NORMED, maxv),
        (cv2.TM_CCOEFF, maxv), (cv2.TM_CCOEFF_NORMED, maxv))
    method = methods[int(sys.argv[1])]
    # open template and convert to grayscale
    templ = cv2.imread(sys.argv[2])
    templ_gray = cv2.cvtColor(templ, cv2.COLOR_BGR2GRAY)
    # process image
    img = cv2.imread(sys.argv[3])
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    result = cv2.matchTemplate(img_gray,templ_gray, method[0])
    min_max = cv2.minMaxLoc(result)
    x = min_max[method[1]][0]
    y = min_max[method[1]][1]
    w, h = templ.shape[:2]
    print(min_max[method[1]][1]min_max[method[1]][0])
```

OpenCV beépített függvényével

Használat parancssorból

```
python3 img_correlation.py 0 mona_temp4.png monalisa.jpg
25 57
python3 img_correlation.py 0 mona_temp4.png monalisa_blur.jpg
25 57
```





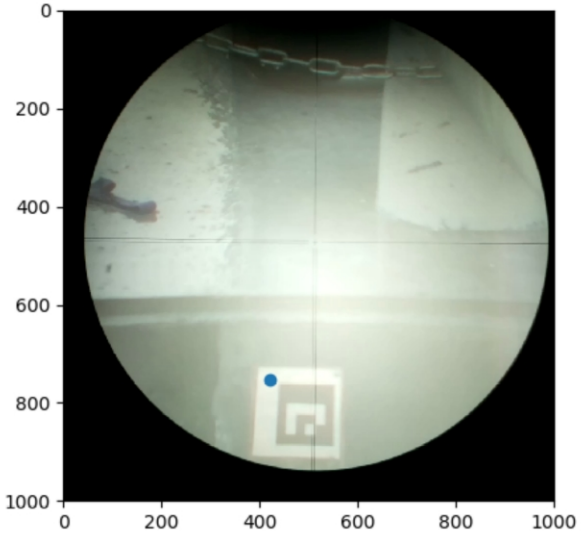
Kép Módszer	Alap X	 Y	Blur X	 Y
0	25	57	25	57
1	25	57	25	57
2	25	57	25	57
3	25	57	25	57

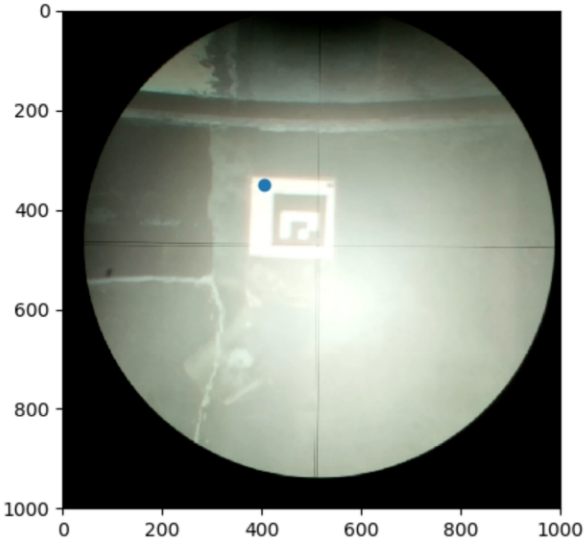
Futási idők másodp.	Octave	Python	C++*	Python OpenCV	C++ OpenCV
Mona 182 x 276	1.7	0.7	0.2	0.4	0.1
Lab1 ~2k x 3k	466	235	28	1.2	0.8
Lab ~5.5k x 4k	2880	703	109	2.8	1.8

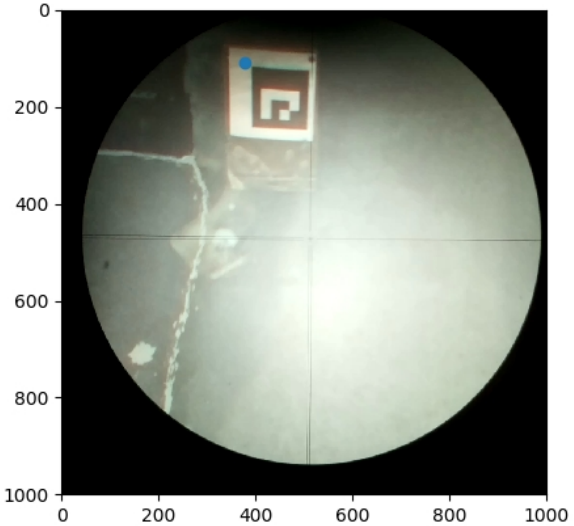
* Horváth Viktor Győző munkája

Mozgásvizsgálati alkalmazás

- Síkbeli mozgások kimutatása (elfordulás nélkül)
- Gyors mozgásnál videó feldolgozás
- Gyorsítási lehetőség – képrészlet feldolgozás (két kép között kis mozgás)
- https://github.com/zsiki/ulyxes/tree/master/camera
- Kamera kalibráció, 1 pixel mérete a valóságban







Élkiemelés, detektálás

- Szűrők (kernel) alkalmazása (gimp)

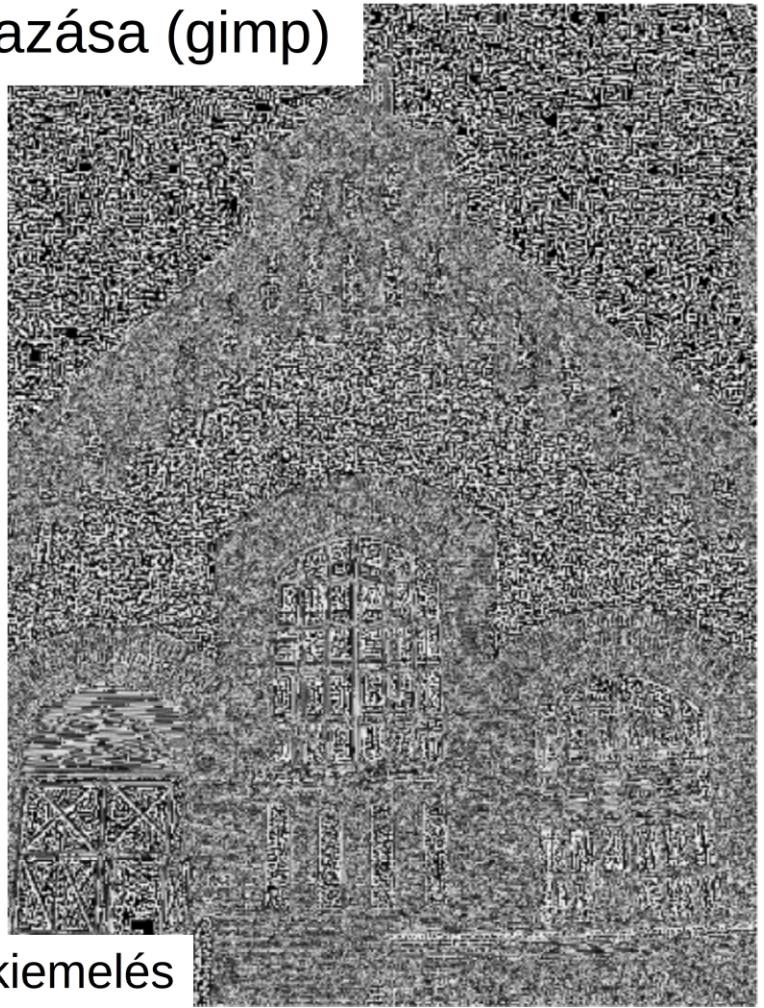
[[1/9, 1/9, 1/9],
[1/9, 1/9, 1/9],
[1/9, 1/9, 1/9]]

Low-pass szűrők – zaj eltávolítás

Élkiemelés, detektálás

- Szűrők (kernel) alkalmazása (gimp)

[[-1, -1, -1],
[-1, 8, -1],
[-1, -1, -1]]

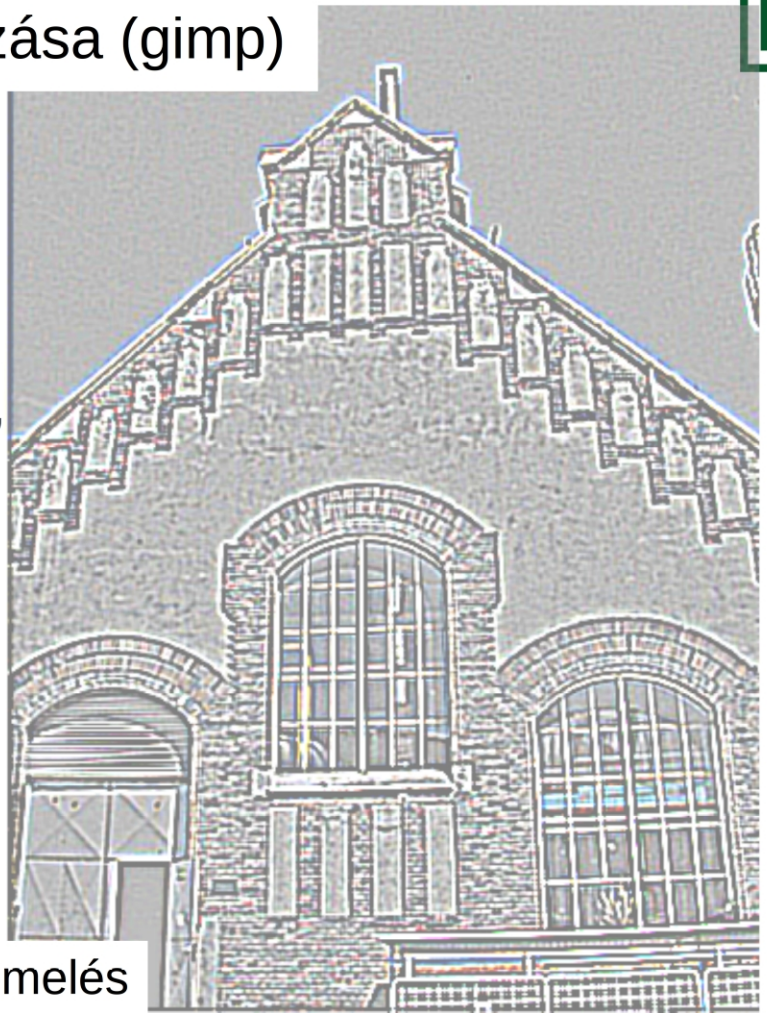


High-pass szűrők – él kiemelés

Élkiemelés, detektálás

- Szűrők (kernel) alkalmazása (gimp)

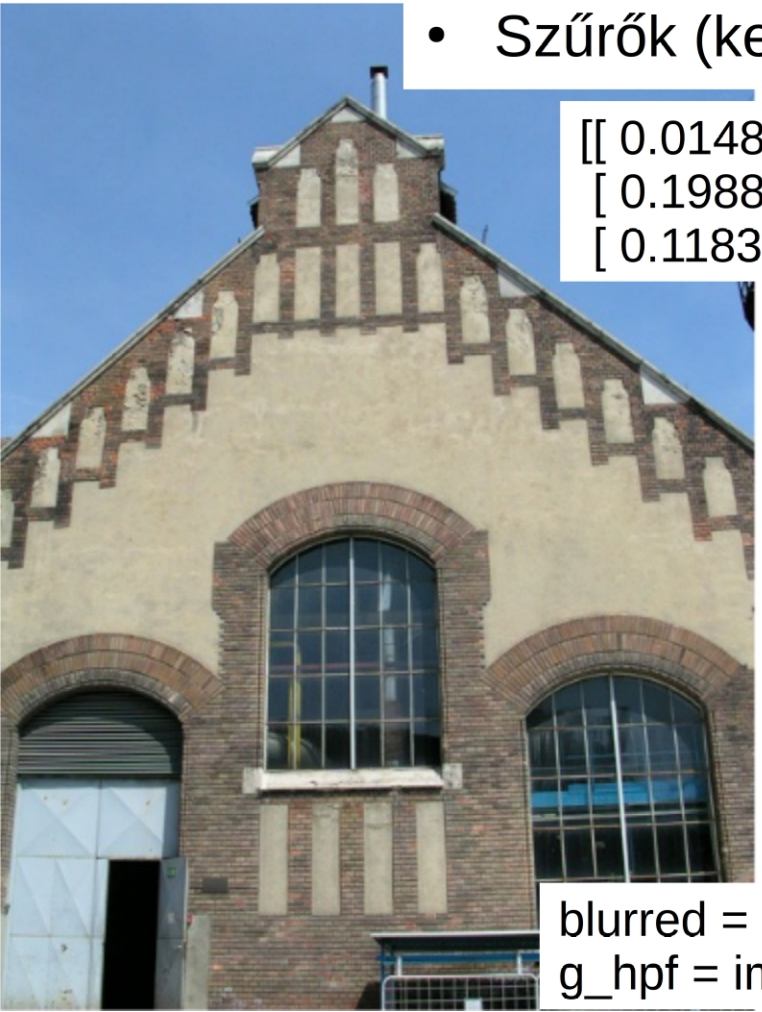
[[-1, -1, -1, -1, -1],
[-1, 1, 2, 1, -1],
[-1, 2, 4, 2, -1],
[-1, 1, 2, 1, -1],
[-1, -1, -1, -1, -1]]



High-pass szűrők – él kiemelés

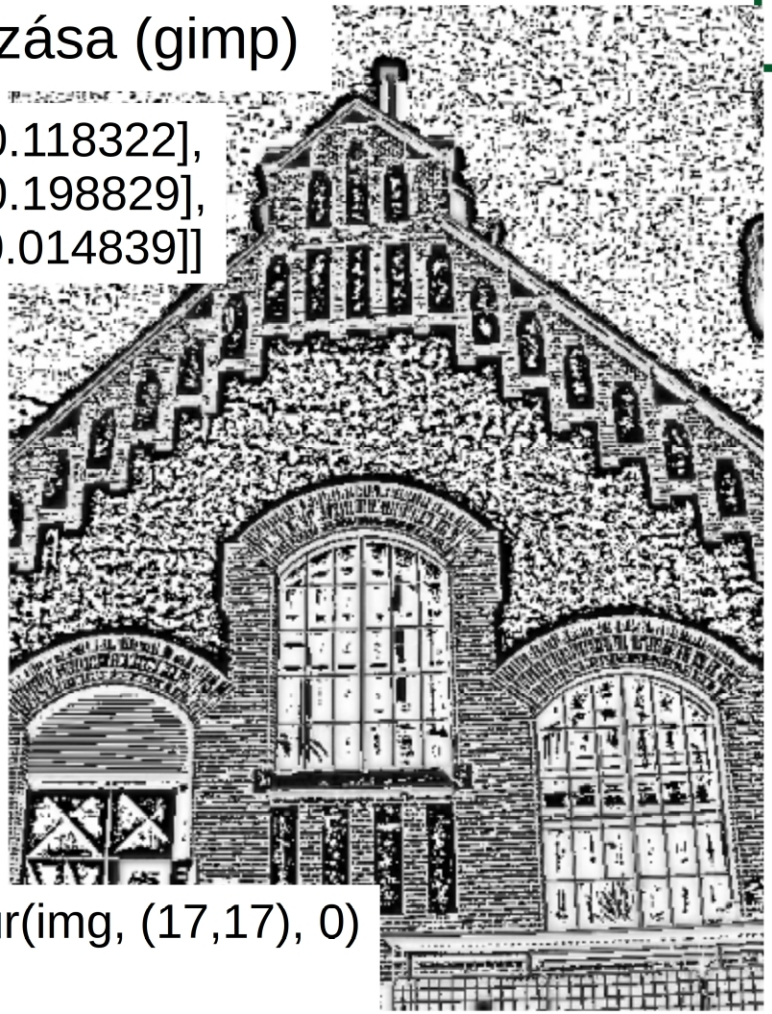
Élkiemelés, detektálás

- Szűrők (kernel) alkalmazása (gimp)



```
[[ 0.014839,  0.049817,  0.118322],
 [ 0.198829,  0.236383,  0.198829],
 [ 0.118322,  0.049817,  0.014839]]
```

3 x 3 Gauss kernel



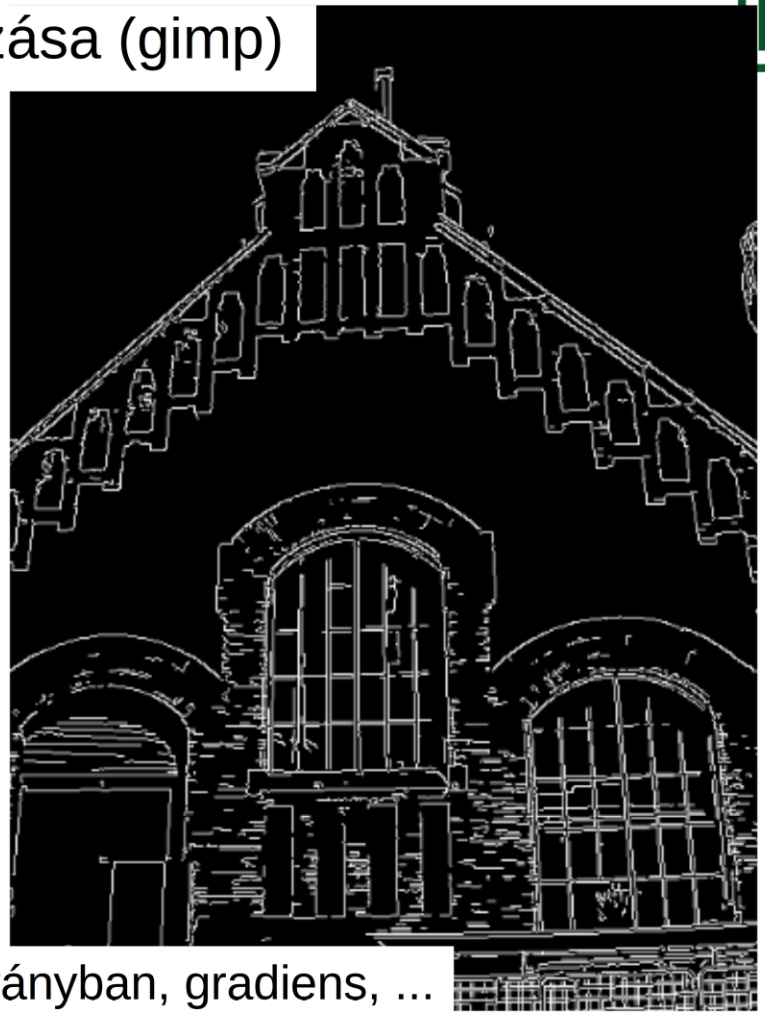
```
blurred = cv2.GaussianBlur(img, (17,17), 0)
g_hpf = img - blurred
```

Élkiemelés, detektálás

- Szűrők (kernel) alkalmazása (gimp)

G_x [[1, 0, -1],
[2, 0, -2],
[1, 0, -1]]

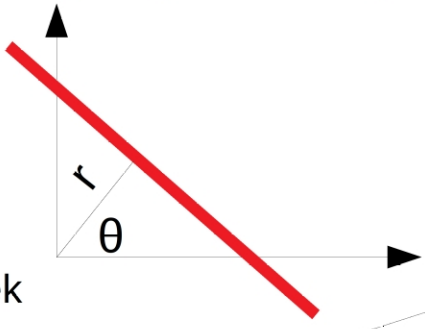
G_y [[1, 2, 1],
[0, 0, 0],
[-1, -2, -1]]



Zaj szűrés, Sobel kernel x és y irányban, gradiens, ...

Vonal felismerés

Hugh algoritmus (bináris élkiemelt képekre)



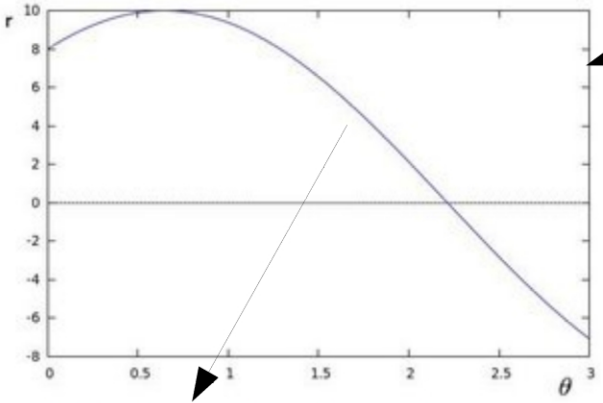
$$y = \frac{-\cos\theta}{\sin\theta}x + \frac{r}{\sin\theta}$$

Egyenes egyenlete y = mx + b

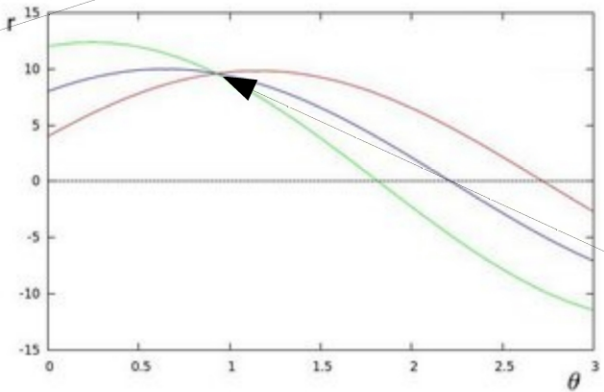
$$r = x_0\cos\theta + y_0\sin\theta$$

x_0, y_0 pontok átmenő egyenesek r, θ függvényében

Egy pont (r, Θ) egy egyenesnek felel meg



Egy (x0, y0) ponton átmenő egyenesek képe a Hugh térben



Görbe ábrák az OpenCV dokumentációból átvéve

Mindel él pontra írjuk fel az összes rajta átmenő egyenest
0 <= Θ < 180°
-d <= r <= d (d a képátmérő)

Több ponthoz tartozó görbe egy pontban metszi egymást => egy egyenesre esnek

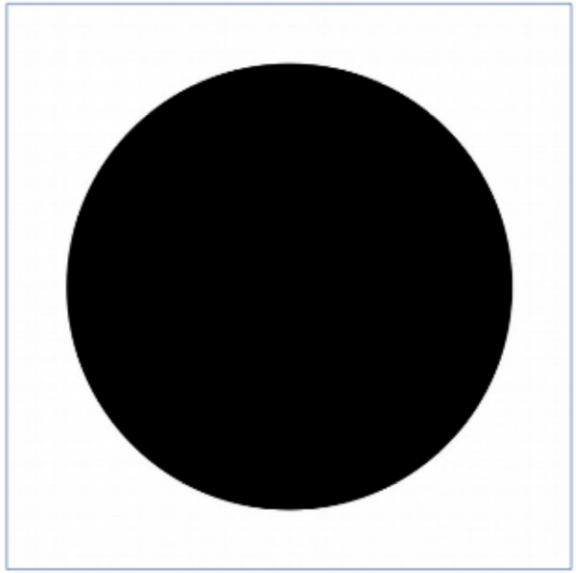
Vonal felismerés

lines.py

```
import cv2
import numpy as np
import os.path
from sys import argv

if len(argv) < 2:
    print("Usage: {} img_file [img_file ...]".format(argv[0]))
    exit()

# process images
for fn in argv[1:]:
    img = cv2.imread(fn)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 120)
    minLineLength = 20
    maxLineGap = 5
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 10, minLineLength, maxLineGap)
    for line in lines:
        for x1, y1, x2, y2 in line:
            print(x1, y1, x2, y2)
            cv2.line(img, (x1,y1), (x2,y2), (0, 255, 0), 8)
    fn1 = os.path.split(fn)
    fn2 = os.path.join(fn1[0], "e_" + fn1[1])
    fn3 = os.path.join(fn1[0], "l_" + fn1[1])
    cv2.imwrite(fn2, edges)
    cv2.imwrite(fn3, img)
```

Canny:
1. zajszűrés (5x5 Gauss)
2. Sobel x és y (gradiens és irány)
3. Lokális maximum keresés
4. Min-max küszöbértékek

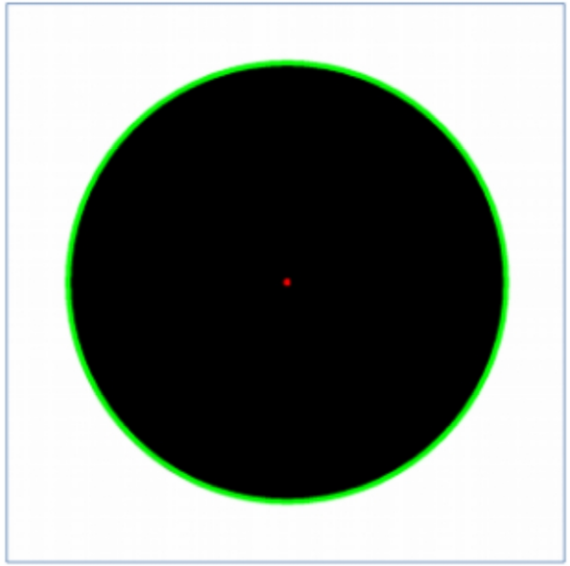
r – 1 pixelenként
Θ – fokonként (rad)
Min. 10 metszés



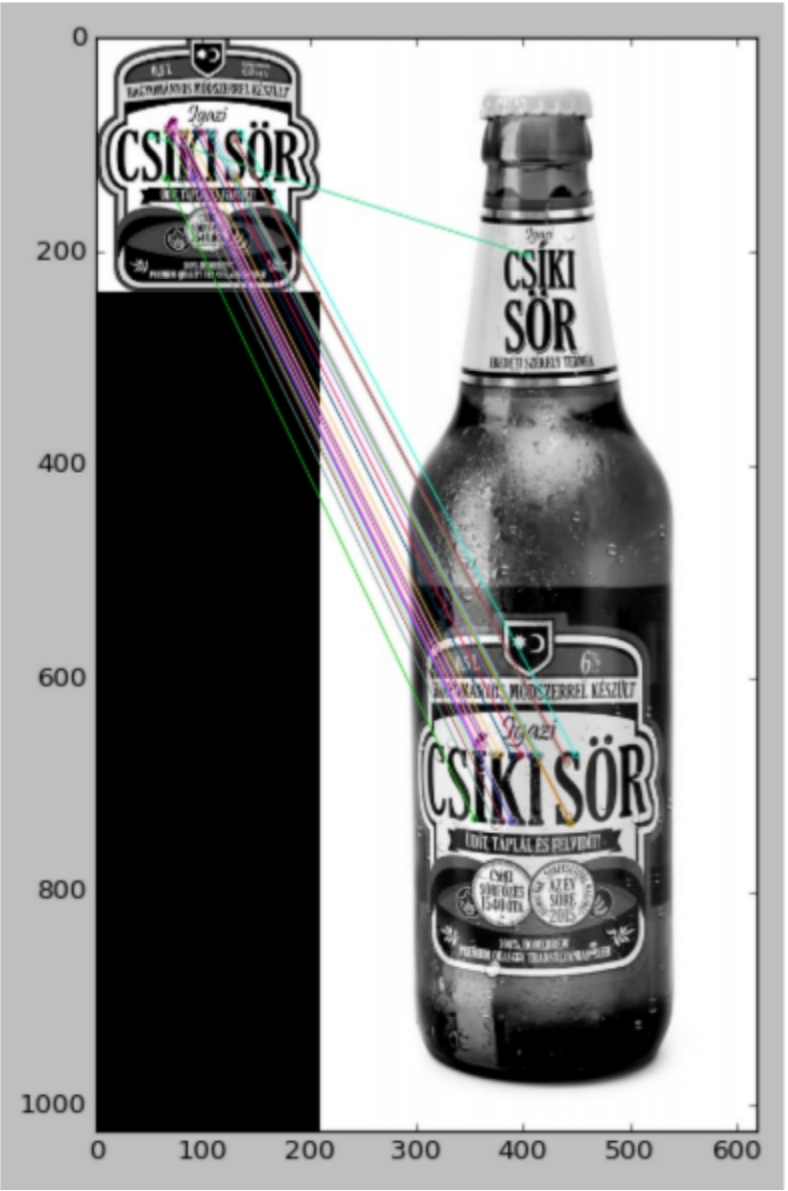
Körök felismerése

circles.py

```
import cv2
import numpy as np
import os.path
from sys import argv

if len(argv) < 2:
    print("Usage: {} img_file [img_file ...]".format(argv[0]))
    exit()
for fn in argv[1:]:    # process images
    src_img = cv2.imread(fn)    # load image
    # convert image to gray scale
    gray_img = cv2.cvtColor(src_img, cv2.COLOR_BGR2GRAY)
    img = cv2.medianBlur(gray_img, 5)     # noise reduction
    #find circles
    circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 1, 400,
                   param1=100, param2=30, minRadius=10, maxRadius=1000)
    print(circles)
    circles = np.uint16(np.around(circles))
    for i in circles[0,:]:
        # draw the outer circle
        cv2.circle(src_img, (i[0], i[1]), i[2], (0, 255, 0), 10)
        # draw the center of the circle
       cv2.circle(src_img, (i[0], i[1]), 2, (0,0,255), 10)
    fn1 = os.path.split(fn)
    fn2 = os.path.join(fn1[0], "c_" + fn1[1])
    cv2.imwrite(fn2, src_img)
```



Alakfelismerés



orb.py

```
import cv2
from matplotlib import pyplot as plt
from sys import argv

if len(argv) < 3:
    print "Usage {} sample image".format(argv[0])
    exit(0)
img1 = cv2.imread(argv[1], 0)
img2 = cv2.imread(argv[2], 0)
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1, des2)
matches = sorted(matches, key = lambda x:x.distance)
img3 = cv2.drawMatches(img1, kp1, img2, kp2, matches[:25],
    img2, flags=2)
plt.imshow(img3)
plt.show()
```

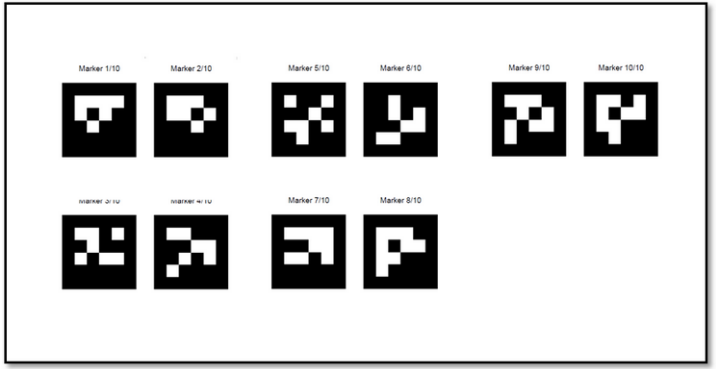
kp1 424 db.
kp2 500 db.

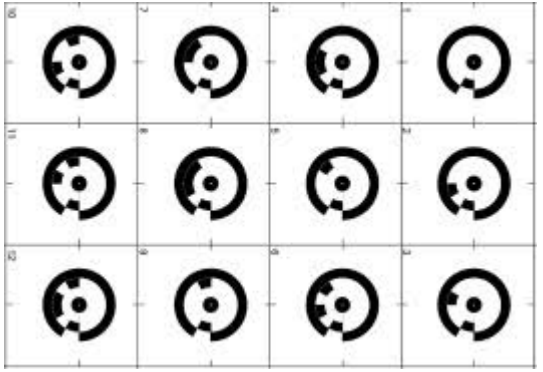
matches 168 db.

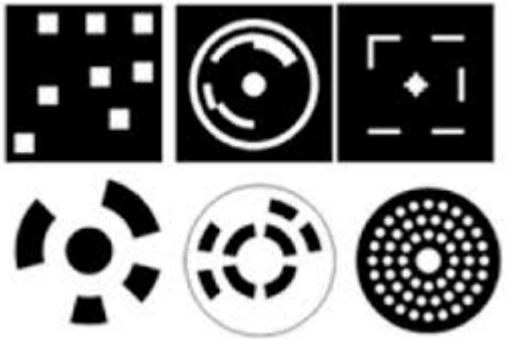
A minta kép más méretben, elfordulva, torzulva jelenhet meg

Kódolt pontok

Kódolt egyedi minták
Alkalmazások
- kiterjesztett valóság
- fotogrammetria
- beltéri navigáció



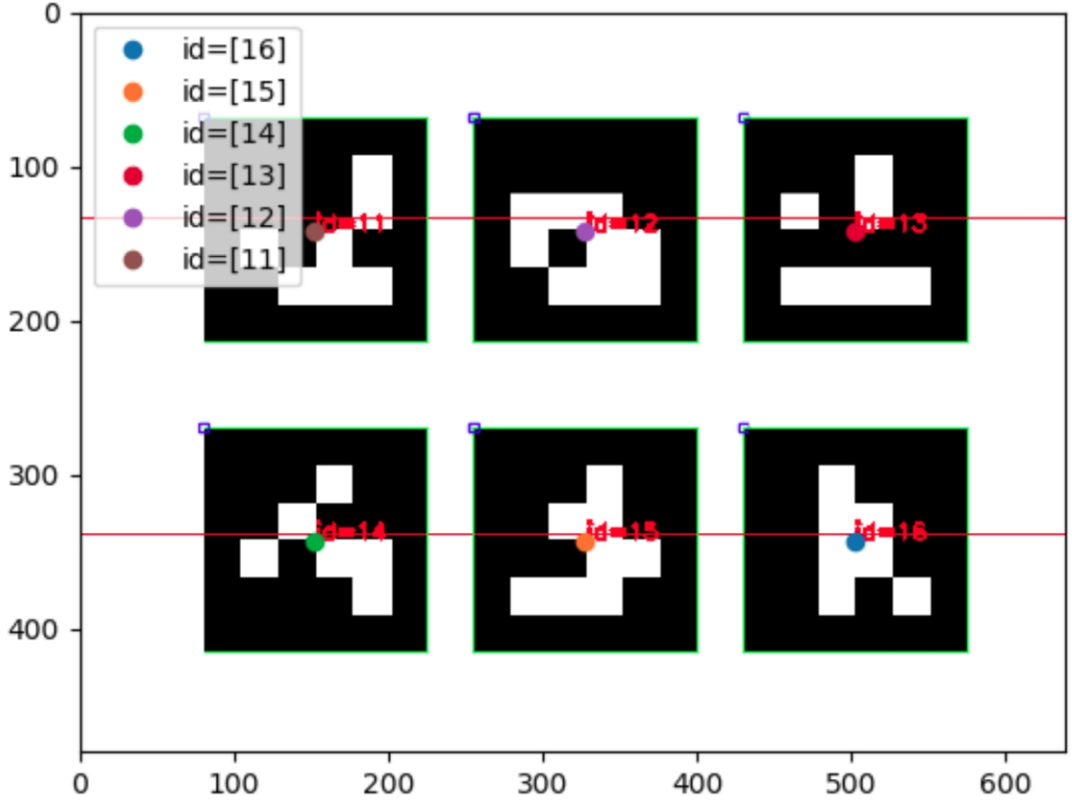


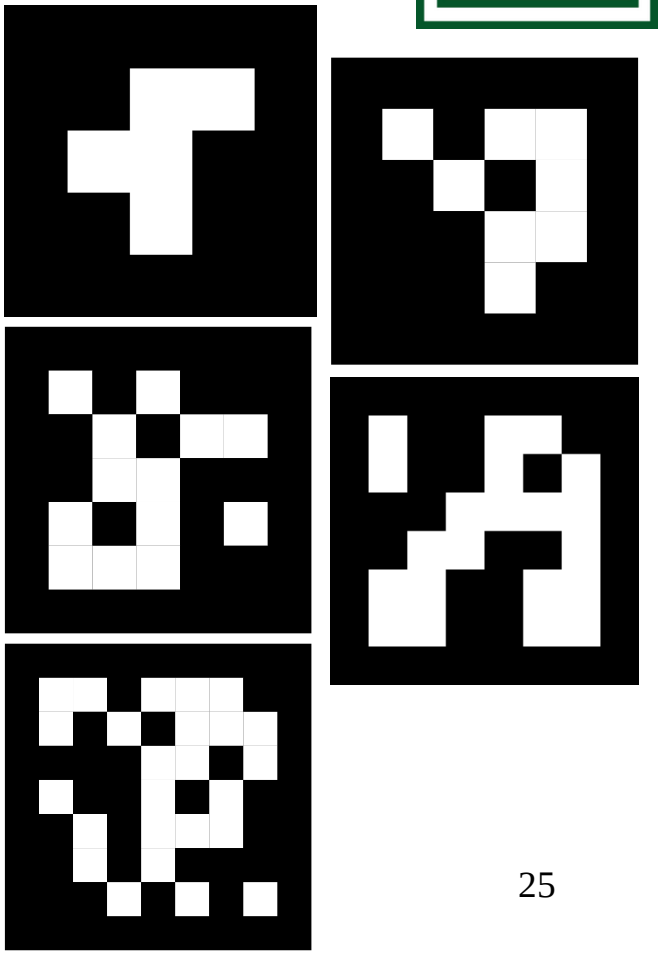


A jeleket rögzítjük a tárgyon
- AR – a kódolt pont helyén valami más jelenik meg a képernyőn
- FG – illesztőpont automatikus felismerése a képen
- BN – a kódolt pontok koordinátái ismertek az alkalmazásban

ArUco kódok

3 x 3, 4 x 4, 5 x 5 , … négyzetrács + fekete keret





ArUco kódok

3 x 3, 4 x 4, 5 x 5 , … négyzetrács + fekete keret

Feldolgozás menete:

ArUco kódok

3 x 3, 4 x 4, 5 x 5 , … négyzetrács + fekete keret

Feldolgozás menete:
Fekete-fehér átalakítás (küszöbérték?)

ArUco kódok

3 x 3, 4 x 4, 5 x 5 , … négyzetrács + fekete keret

Feldolgozás menete:
Fekete-fehér átalakítás (küszöbérték?)
Négyszögletes körvonalak kikeresése

ArUco kódok

3 x 3, 4 x 4, 5 x 5 , … négyzetrács +
fekete keret

Feldolgozás menete:
Fekete-fehér átalakítás (küszöbérték?)
Négyszögletes körvonalak kikeresése
Perspektív torzulás kiküszöbölése
0/1 értékek meghatározása