

Diplomski studij računarstva

Računalna geometrija i robotski vid

Laboratorijska vježba 3

Houghova transformacija

Ivan Gudelj, DRB

Osijek, 2022.

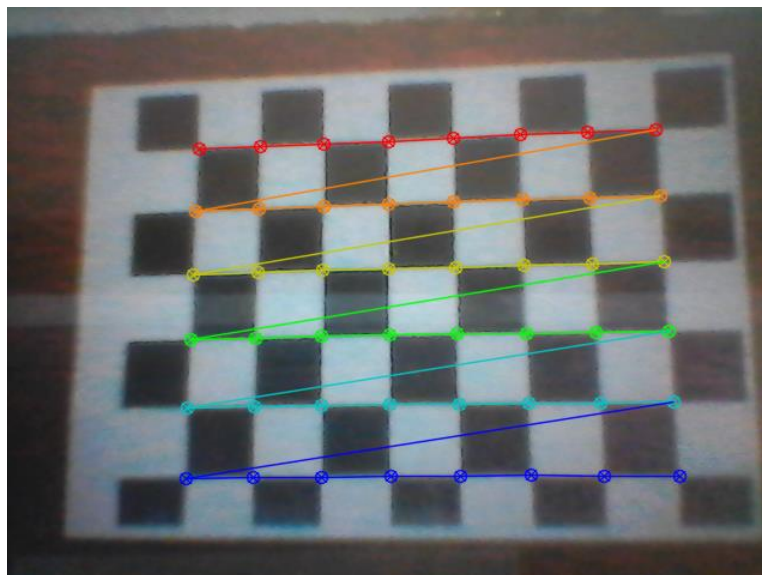
- I. Cilj vježbe: Naučiti kako kalibrirati kameru. Odrediti parametre pravca u 3D prostoru primjenom Houghove transformacije.
- II. Opis vježbe: Potrebno je pomoću, prethodno kalibrirane, web kamere uslikati objekt kvadratnog oblika koji je postavljen na milimetarskom papiru na stolu. Primjenom Houghove transformacije (HT) treba odrediti parametre ρ i θ najdominantnijeg pravca, koji odgovara jednom od rubova objekta na slici. Pod najdominantnijim pravcem podrazumijeva se pravac kojem pripada najveći broj 'glasova' u akumulacijskoj ravnini. Implementacija HT u biblioteci OpenCV vraća popis detektiranih pravaca koji su razvrstani prema broju 'glasova' počevši od najdominantnijeg. Primjenom odgovarajuće transformacije, odrediti ρ i θ tog pravca u koordinatnom sustavu milimetarskog papira. Provjeriti koliko je odstupanje dobivenog pravca od stvarnog (odgovarajućeg) ruba objekta.
- III. Rad na vježbi: a) Skinuti skriptu calibration.py. b) Pokrenuti python skriptu te uz pomoć kalibracijskog panela kalibrirati kameru. Kalibracijski panel možete napraviti pomoću slike ches s board.pdf. c) Napisati funkciju tako da korisnik pomoću web kamere uslika objekt koji se nalazi na milimetarskom papiru na stolu. Omogućiti u programu da se mišem može označiti (klikom) četiri ugla milimetarskog papira na slici. Odrediti rotacijsku matricu i translacijski vektor uz pomoć prethodne učitane intrinzične matrice te koeficijenta distorzije (camera_params.json). Primjenom izraza u prilogu treba odrediti koliko se pravac, dobiven na slici pomoću Houghove transformacije, podudara s odgovarajućim rubom objekta na stolu.

RJEŠENJE:

Korisniku je omogućeno da odabire dali želi ponovno kalibrirati kameru ili želi koristiti postojeće parametre kalibrirane kamere te da odabere jednu od slika koje se nalaze u mapi.

```
newParameters = input("Ukoliko zelite rekalibrirati kameru unesite 1, ukoliko zelite vec postojece psotavke unesite 2:")
if newParameters == "1":
    exec(open('calibration.py').read())
elif newParameters == "2":
    selectedImage = input("Unesite redni broj slike (1-3): ")
    paperImage = cv2.imread("linijaMmpapir"+str(selectedImage)+".jpg") #Stalna je slika budući se kamera crasha sa pokušajem uzivog prepoznavanja
    jsonFile = open("camera_params.json")
    jsonData = json.load(jsonFile)
    jsonFile.close()
    cameraMatrix = jsonData['camera_matrix']
    distCoeffs = jsonData['dist_coeffs']
    cameraMatrix = np.array(cameraMatrix)
    distCoeffs = np.array(distCoeffs)
```

Ukoliko korisnik odabere ponovnu kalibraciju kamere i njenih parametara, tada se pokreće datoteka calibration.py koja uz odgovarajuće komande sprema podatke kao što su cameraMatrix i distanceCoefficients i rezultira sljedećem:



Ukoliko pak korisnik odluči da su postojeći parametri zadovoljavajući, učitava se jedna od postojećih slika te se od korisnika traži da odabere 4 vrha milimetarskog papira (uz pomoć funkcije `cv2.setMouseCallback()`).

```
def selectEdge(event, x, y, flags, userData):

    global edgesSelected
    global edgesCoordinates
    global croppedCoordinates
    global minValues
    global maxValues

    if event == cv2.EVENT_LBUTTONDOWN:
        edgesCoordinates.append((x, y))
        if len(edgesCoordinates) >= 4:
            maxValues = np.array(edgesCoordinates[0])
            minValues = np.array(edgesCoordinates[0])
            for i in edgesCoordinates:
                if i[0] > maxValues[0]:
                    maxValues[0] = i[0]
                if i[1] > maxValues[1]:
                    maxValues[1] = i[1]
                if i[0] < minValues[0]:
                    minValues[0] = i[0]
                if i[1] < minValues[1]:
                    minValues[1] = i[1]
            edgesCoordinates = []
            cv2.destroyAllWindows()

def cutImage(ImageForCutting):
    return ImageForCutting[minValues[1]:maxValues[1],minValues[0]:maxValues[0]]
```

U sljedećim dijelovima koda vidimo prepoznavanje parametara pravca prepoznatog na slici koristeći `cv2.HoughLines()` funkcije:

```
def getRTheta(lines):
    for r,theta in lines[0]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*r
        y0 = b*r
        # x1 stores the rounded off value of (rcos(theta)-1000sin(theta))
        x1 = int(x0 + 1000*(-b))
        # y1 stores the rounded off value of (rsin(theta)+1000cos(theta))
        y1 = int(y0 + 1000*(a))
        # x2 stores the rounded off value of (rcos(theta)+1000sin(theta))
        x2 = int(x0 - 1000*(-b))
        # y2 stores the rounded off value of (rsin(theta)-1000cos(theta))
        y2 = int(y0 - 1000*(a))
        cv2.line(croppedImg,(x1,y1), (x2,y2), (0,0,255),2)
    return r,theta
```

Određivanje Ro i theta

```
undistImgCopy = undistortedImg.copy()
croppedImg = cutImage(undistImgCopy)
grayImage = cv2.cvtColor(croppedImg,cv2.COLOR_BGR2GRAY)
edges= cv2.Canny(grayImage, 100, 140,apertureSize=3)
lines = cv2.HoughLines(edges,1,np.pi/180, 70)
plt.figure(3)
plt.imshow(croppedImg)
Ro, theta = getRTheta(lines)
plt.figure()
plt.imshow(croppedImg)
plt.xlabel("Kut izmedu pravca i x osi je: " + str(theta*57.2957795))
cv2.imwrite('linesDetected.jpg', croppedImg)
```

Iscrtavanje Prepoznatog pravca i ispis kuta između pravca i x osi

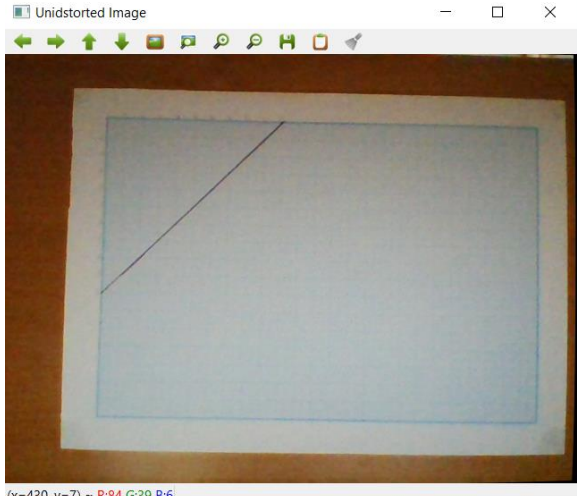
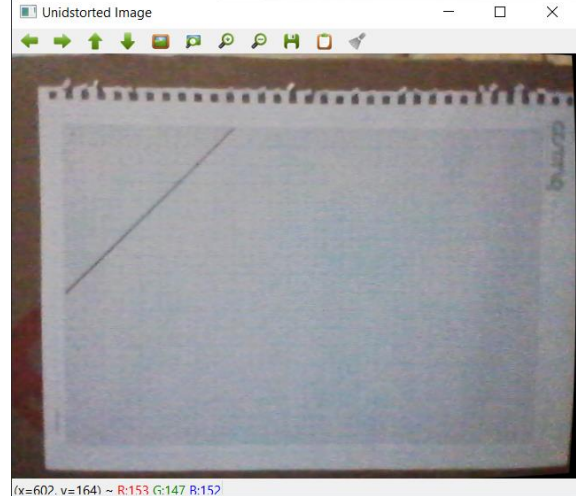
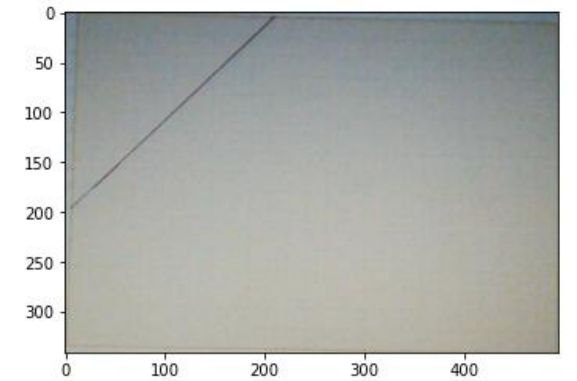
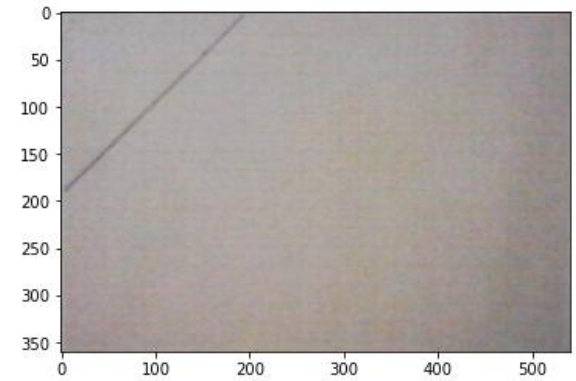
Računanje translacijskih i rotacijskih vektora se radi pomoću funkcije solvePnP. Dok se pomoću funkcije Rodrigues dobiva rotacijska matrica iz rotacijskog vektora te je određivanje parametara pravca napravljeno preko formula koje su dane u predlošku laboratorijske vježbe.

```
ret,rvec,tvec = cv2.solvePnP(objPoints,imgPoints,cameraMatrix,distCoeffs)
rotationMat, JB = cv2.Rodrigues(rvec)
A=cameraMatrix @ rotationMat
B=cameraMatrix @ tvec

lx = A[0, 0] * np.cos(theta) + A[1, 0] * np.sin(theta) - Ro * A[2, 0]
ly = A[0, 1] * np.cos(theta) + A[1, 1] * np.sin(theta) - Ro * A[2, 1]
lr = B[2] * Ro - B[0] * np.cos(theta) - B[1] * np.sin(theta)
realTheta = np.arctan2(ly, lx)
realRho = lr / np.sqrt(lx**2 + ly**2)

print("Prava theta: " + str(realTheta))
print("Pravi rho: " + str(realRho))
```

Rezultati

<p>Slika 1:</p> 	<p>Slika 2:</p> 
<p>Označavanje rubova milim. papira</p> 	<p>Označavanje rubova milim. papira</p> 
<p>Isječena slika nakon označavanja</p>	

