FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski studij računarstva

Računalna geometrija i robotski vid

Laboratorijska vježba 5

TRODIMENZIONALNA REKONSTRUKCIJA SCENE IZ DVIJE SLIKE

Ivan Gudelj, DRB

I. Cilj vježbe: Naučiti postupak trodimenzionalne rekonstrukcije scene iz dvije slike.

II. Opis vježbe: Trodimenzionalna rekonstrukcija scene jedan je od temeljnih problema računalnog vida. Određivanje trodimenzionalne strukture scene na temelju jedne slike još je uvijek neriješen problem zbog višeznačnosti projekcije trodimenzionalnih objekata na dvodimenzionalnu ravninu slike. Naime, jedna slika može predstavljati više različitih trodimenzionalnih scena, tj. može imati potpuno istu sliku. Međutim, postoje metode koje iz dvije slike iste scene snimljene iz različitih pogleda mogu rekonstruirati trodimenzionalnu geometriju scene [1]. U [2] je opisano programsko rješenje problema rekonstrukcije trodimenzionalnog prostora iz dvije slike slikane jednom ili dvjema kamere iz dva različita pogleda, odnosno kuta, čiji se prijevod koristi u ovoj vježbi. Potrebno je pomoću web kamere uslikati dvije slike iste scene snimljene iz različitih pogleda. Web kamera je prethodno kalibrirana te je poznata projekcijska matrica. Nakon toga se detektiraju SIFTznačajke [3] na slikama, te se značajke jedne slike inicijalno sparuju s odgovarajućim značajkama druge slike tako da svaki par značajki odgovara jednoj točki promatrane scene. Sparivanje se izvodi na temelju sličnosti deskriptora pridruženih značajkama. Na temelju ovih parova značajki estimira se fundamentalna matrica koja sadrži informaciju o epipolarnoj geometriji kamera. Estimacija fundamentalne matrice se izvršava RANSAC metodom kojom se ujedno i eliminiraju pogrešno sparene značajke. Na temelju fundamentalne matrice i projekcijske matrice kamere određuje se esencijalna matrica, koja sadrži informaciju o međusobnom položaju kamera koja je neophodna za triangulaciju, odnosno izračunavanje pozicija 3D točke u prostoru.

III. Rad na vježbi: Napisati aplikaciju koja omogućuje sljedeće:

a) Učitati i prikazati prvu i drugu sliku iste scene snimljene iz različitih pogleda.

b) Detektirati SIFT značajke na obje slike te provesti inicijalno sparivanje značajki.

c) Označiti i prikazati značajke dobivene na obje slike te pravcima povezati slične značajke dobivene usporedbom.

d) Na temelju parova značajki estimirati fundamentalnu matricu, F, RANSAC algoritmom primjenom funkcije cv2.findFundamentalMat.

e) Na temelju epipolarnog ograničenja (maske) koje proizlazi iz estimirane fundamentalne matrice F, odbaciti parove značajki koji su krivo spareni.

f) Ponovno označiti, prikazati te pravcima povezati ostale parove značajki dobivene na obje slike

g) Pomoću fundamentalne matrice F, te projekcijske matrice kamere, P, odrediti esencijalnu matricu E prema formuli: $E=P^T \cdot F \cdot P$

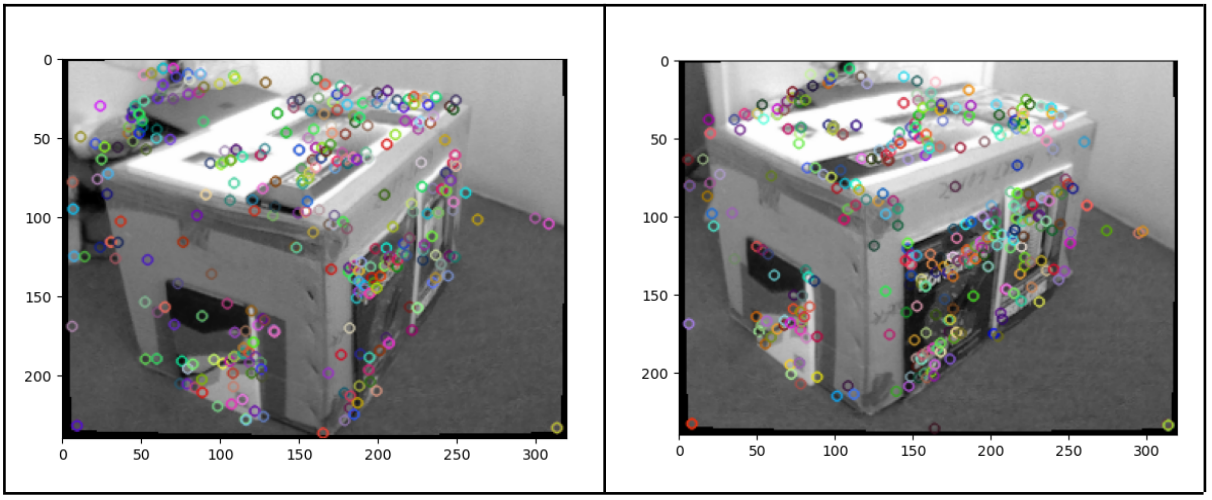
h) Estimirati 3D točku za svaki od ostalih parova značajki primjenom funkcije convert_2d_points_to_3d_points te spremiti 3D koordinate u json datoteku. Funkcija convert_2d_points_to_3d_points služi za estimiranje 3D točke iz skupa sparenih značajki odnosno točaka. convert_2d_points_to3d_points(points_2d_L, points_2d_R, E, P) Argumenti ove funkcije su značajke na prvu sliku points_2d_L, odgovarajuće značajke na drugu sliku points_2d_R, esencijalna matrica E te projekcijska matrica P.

RJEŠENJE:

Prilikom pokretanja aplikacije postavljamo kalibraciju kamere na postojeće parametre kamere. Parametri kamere se nalaze ispod:

```
cameraParamsMatrix = [[346.40640259, 0.0, 180.93818665],
                      [0.0, 341.74661255, 101.59582520],
                      [0.0, 0.0,  1.0]]
```

Nakon što se učitaju dvije slike koje prikazuju istu scenu ali iz različitih kuteva. Vrši se detekcije SIFT značajki pomoću funkcije detectAndCompute. Nakon što su značajke pronađene, prikazuju se dvije slike sa označenim značajkama na njima




```
imageL= cv.imread("imageL.bmp")
imageR= cv.imread("imageR.bmp")
#-------------------------------------------------------
imageL_grayscale = cv.cvtColor(imageL, cv.COLOR_BGR2GRAY)
imageR_grayscale = cv.cvtColor(imageR, cv.COLOR_BGR2GRAY)
#-------------------------------------------------------
sift=cv.xfeatures2d.SIFT_create()
keypointsL, desL = sift.detectAndCompute(imageL_grayscale, None)
keypointsR, desR = sift.detectAndCompute(imageR_grayscale, None)
#-------------------------------------------------------
imageL_sift = cv.drawKeypoints(imageL,keypointsL,None)
plt.figure(1)
plt.imshow(imageL_sift)
imageR_sift = cv.drawKeypoints(imageR,keypointsR,None)
plt.figure(2)
plt.imshow(imageR_sift)
#-------------------------------------------------------
```

Poklapanje značajki se izvodi korištenjem FlannMatchera odnosno metode knnMatch(). Nakon čega se isrctavaju parovi Keypointa te ih filtriramo.

```
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks = 50)

#flann = cv.FlannBasedMatcher(index_params, search_params)
flann =  cv.DescriptorMatcher_create(1)
parovi = flann.knnMatch(desL,desR,k=2)

#bf=cv.BFMatcher()
#parovi= bf.knnMatch(desL,desR,k=2)
#-----------------------------------------------------------
#Filtriranje parova
dobri_parovi = []
newLeftPoints = []
newRightPoints = []
for m,n in parovi:
    if m.distance < 0.75*n.distance:
        dobri_parovi.append([m])
        newLeftPoints.append(keypointsL[m.queryIdx].pt)
        newRightPoints.append(keypointsR[m.trainIdx].pt)
matched_image= cv.drawMatchesKnn(imageL,keypointsL,imageR,keypointsR,dobri_parovi,
                                 None,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.figure(3)
plt.imshow(matched_image)
plt.title("Filtering Matches")
```

Estimacija fundamentalne matrice prikazana je na slici ispod. Za estimaciju se koristi funkcija FindFundamentalMat.

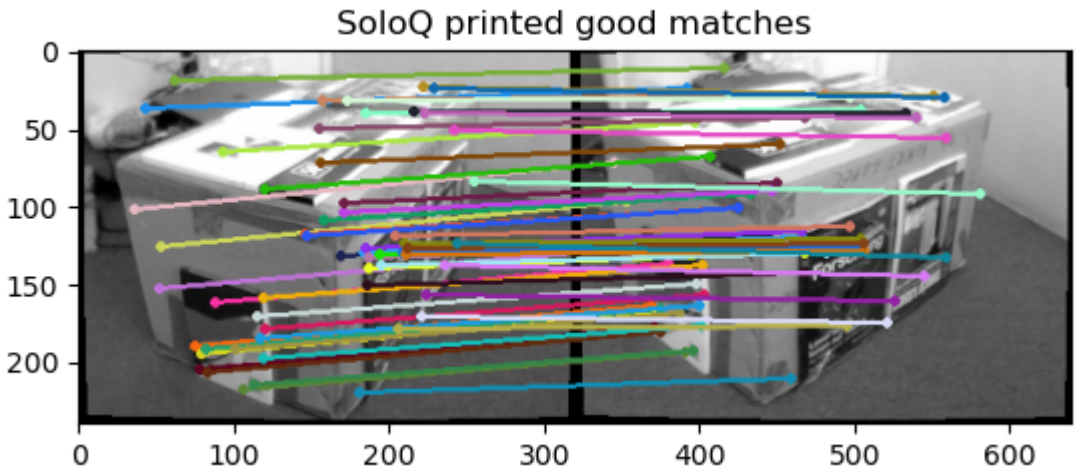
```
#FundamentalMatrixTime
ptsLAsArray = np.asarray(newLeftPoints)
ptsRAsArray = np.asarray(newRightPoints)

leftFilteredKeypoints = []
rightFilteredKeypoints= []
#-----------------------------------------------------------
FundamentalMatrix, mask = cv.findFundamentalMat(ptsLAsArray, ptsRAsArray, cv.FM_RANSAC)
```

Ponovno označavanje i prikazivanje parova značajki (ručno) prikazano je ispod kao i određivanje esencijalne matrice E pomoću fundamentalne matrice F i projekcijske matrice P:

```
im_h = cv.hconcat([imageL, imageR])


for i in range (0,mask.shape[0]):
    random_color = (random.randint(0, 255),random.randint(0, 255),random.randint(0, 255))
    if (mask[i] == 1):
        leftFilteredKeypoints.append(cv.KeyPoint(ptsLAsArray[i,0],ptsLAsArray[i,1],1))
        im_h = cv.circle(im_h,(int(ptsLAsArray[i,0]),int(ptsLAsArray[i,1])) , 2, random_color, 2)
        rightFilteredKeypoints.append(cv.KeyPoint(ptsRAsArray[i,0],ptsRAsArray[i,1],1))
        im_h = cv.circle(im_h,(int(ptsRAsArray[i,0])+320,int(ptsRAsArray[i,1])) , 2, random_color, 2)
        cv.line(im_h, (int(ptsLAsArray[i,0]),int(ptsLAsArray[i,1])), (int(ptsRAsArray[i,0])+320,int(ptsRAsArray[i,1])), random_color,2)
plt.figure(5)
plt.imshow(im_h)
plt.title("SoloQ printed good matches")
cameraParamsTransposedwithFundamentalMatrix= np.matmul(FundamentalMatrix,np.transpose(cameraParamsMatrix))
EssentialMatrix= np.matmul(cameraParamsTransposedwithFundamentalMatrix,cameraParamsMatrix)
#-----------------------------------------------------------
```



Konverzija 2D točaka u 3D točke se izvodi putem unaprijed predane metode convert_2d_points_to_3d_points.convert_2d_points_to_3d_points(). Također ispis 3D točaka za plotanje se vrši koristeći se kodom na slici ispod:

```
EssentialMatrix = np.matrix(EssentialMatrix)
cameraParamsMatrix = np.matrix(cameraParamsMatrix)

new3DPoints = convert_2d_points_to_3d_points.convert_2d_points_to_3d_points(leftFilteredKeypoints,
                                                                             rightFilteredKeypoints, EssentialMatrix, cameraParamsMatrix)

outputFile = open('points_3d.json',"w")
if(outputFile is None):
        print("\nProblem with writing to the file..")
        quit()
json.dump(new3DPoints.tolist(),outputFile)
print("\n3D Tocke su zapisane u JSON datoteku.")
```