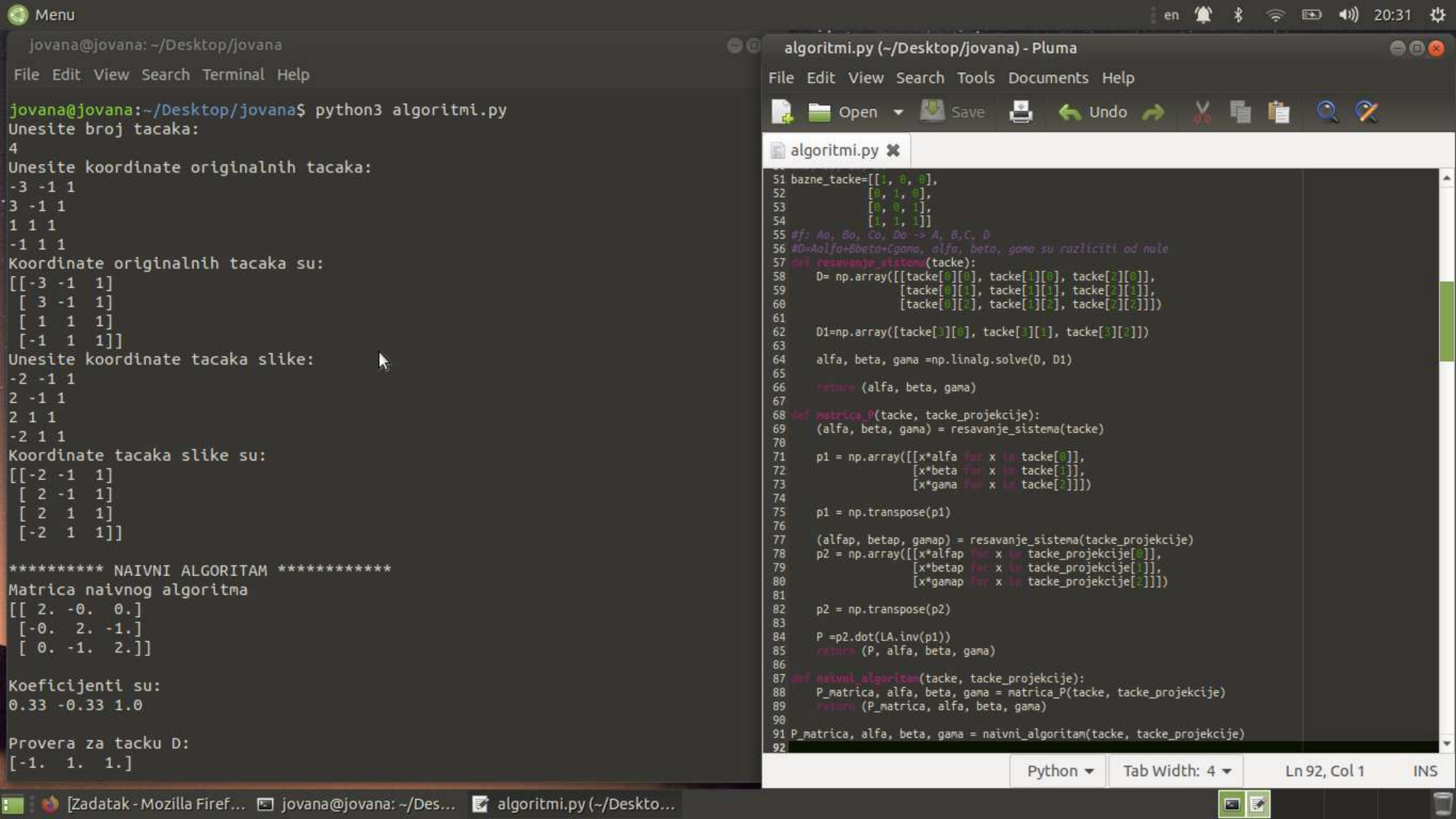
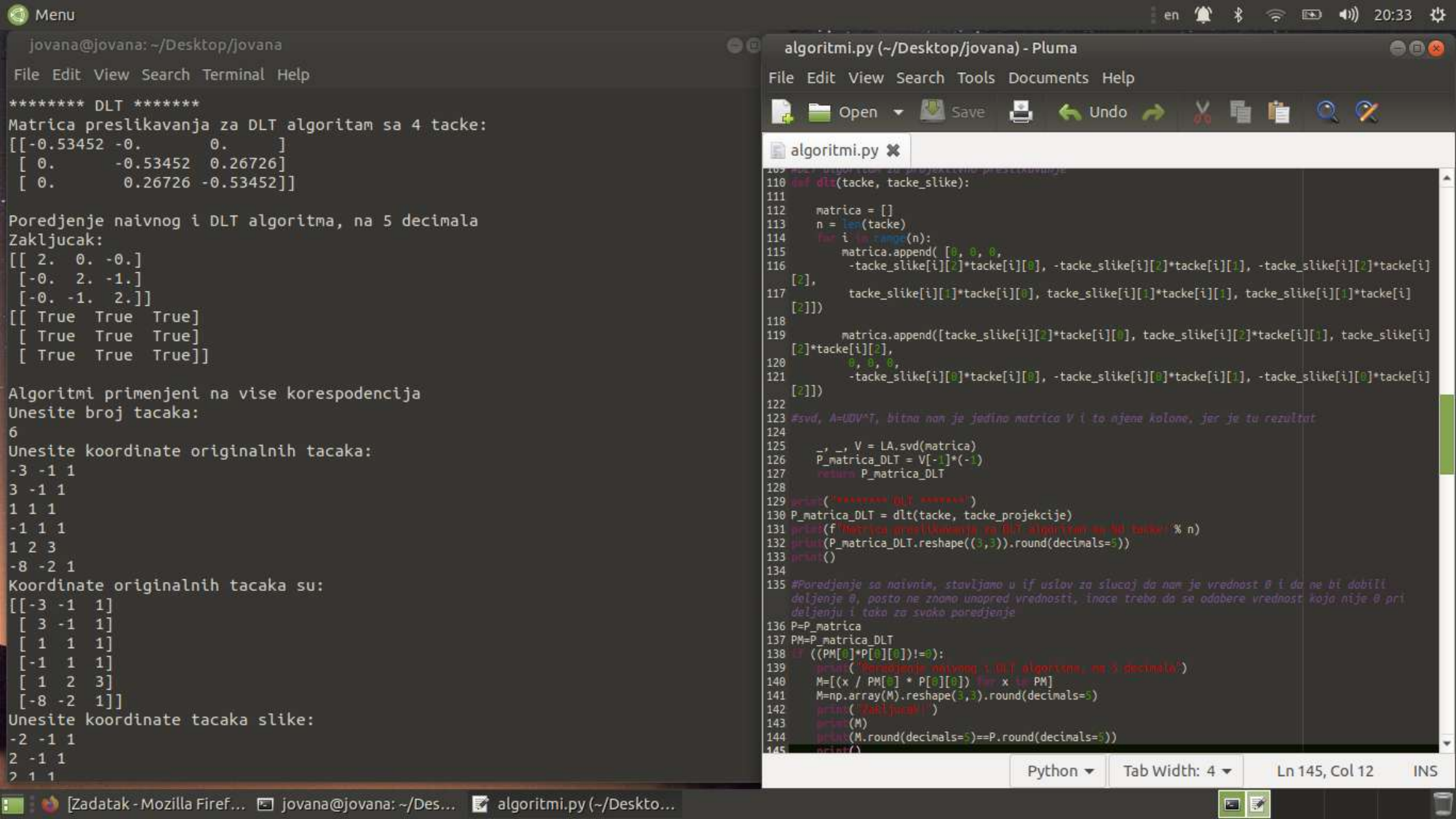

Menu
en
20:31
jovana@jovana: ~/Desktop/jovana
File Edit View Search Terminal Help
jovana@jovana:~/Desktop/jovana$ python3 algoritmi.py
Unesite broj tacaka:
4
Unesite koordinate originalnih tacaka:
-3 -1 1
3 -1 1
1 1 1
-1 1 1
Koordinate originalnih tacaka su:
[[-3 -1 1]
[3 -1 1]
[1 1 1]
[-1 1 1]]
Unesite koordinate tacaka slike:
-2 -1 1
2 -1 1
2 1 1
-2 1 1
Koordinate tacaka slike su:
[[-2 -1 1]
[2 -1 1]
[2 1 1]
[-2 1 1]]
********** NAIVNI ALGORITAM ***********
Matrica naivnog algoritma
[[2. -0. 0.]
[-0. 2. -1.]
[0. -1. 2.]]
Koeficijenti su:
0.33 -0.33 1.0
Provera za tacku D:
[-1. 1. 1.]
algoritmi.py (~/Desktop/jovana) - Pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
algoritmi.py
bazne_tacke=[[1, 0, 0],
[0, 1, 0],
[0, 0, 1],
[1, 1, 1]]
#f: Ao, Bo, Co, Do -> A, B,C, D
#D=Aalfa+Bbeta+Cgama, alfa, beta, gama su razliciti od nule
def resavanje_sistema(tacke):
D= np.array([[tacke[0][0], tacke[1][0], tacke[2][0]],
[tacke[0][1], tacke[1][1], tacke[2][1]],
[tacke[0][2], tacke[1][2], tacke[2][2]]])
D1=np.array([tacke[3][0], tacke[3][1], tacke[3][2]])
alfa, beta, gama =np.linalg.solve(D, D1)
return (alfa, beta, gama)
def matrica_P(tacke, tacke_projekcije):
(alfa, beta, gama) = resavanje_sistema(tacke)
p1 = np.array([[x*alfa for x in tacke[0]],
[x*beta for x in tacke[1]],
[x*gama for x in tacke[2]]])
p1 = np.transpose(p1)
(alfap, betap, gamap) = resavanje_sistema(tacke_projekcije)
p2 = np.array([[x*alfap for x in tacke_projekcije[0]],
[x*betap for x in tacke_projekcije[1]],
[x*gamap for x in tacke_projekcije[2]]])
p2 = np.transpose(p2)
P =p2.dot(LA.inv(p1))
return (P, alfa, beta, gama)
def naivni_algoritam(tacke, tacke_projekcije):
P_matrica, alfa, beta, gama = matrica_P(tacke, tacke_projekcije)
return (P_matrica, alfa, beta, gama)
P_matrica, alfa, beta, gama = naivni_algoritam(tacke, tacke_projekcije)
Python
Tab Width: 4
Ln 92, Col 1
INS
[Zadatak - Mozilla Firef...
jovana@jovana: ~/Des...
algoritmi.py (~/Deskto...


Menu
en
20:33
jovana@jovana: ~/Desktop/jovana
File Edit View Search Terminal Help
******* DLT *******
Matrica preslikavanja za DLT algoritam sa 4 tacke:
[[-0.53452 -0.   0.   ]
 [0.   -0.53452 0.26726]
 [0.   0.26726 -0.53452]]
Poredjenje naivnog i DLT algoritma, na 5 decimala
Zakljucak:
[[2. 0. -0.]
 [-0. 2. -1.]
 [-0. -1. 2.]]
[[True True True]
 [True True True]
 [True True True]]
Algoritmi primenjeni na vise korespodencija
Unesite broj tacaka:
6
Unesite koordinate originalnih tacaka:
-3 -1 1
3 -1 1
1 1 1
-1 1 1
1 2 3
-8 -2 1
Koordinate originalnih tacaka su:
[[-3 -1 1]
 [3 -1 1]
 [1 1 1]
 [-1 1 1]
 [1 2 3]
 [-8 -2 1]]
Unesite koordinate tacaka slike:
-2 -1 1
2 -1 1
algoritmi.py (~/Desktop/jovana) - Pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
algoritmi.py
def dlt(tacke, tacke_slike):
matrica = []
n = len(tacke)
for i in range(n):
matrica.append([0, 0, 0,
-tacke_slike[i][2]*tacke[i][0], -tacke_slike[i][2]*tacke[i][1], -tacke_slike[i][2]*tacke[i][2],
tacke_slike[i][1]*tacke[i][0], tacke_slike[i][1]*tacke[i][1], tacke_slike[i][1]*tacke[i][2]])
matrica.append([tacke_slike[i][2]*tacke[i][0], tacke_slike[i][2]*tacke[i][1], tacke_slike[i][2]*tacke[i][2],
0, 0, 0,
-tacke_slike[i][0]*tacke[i][0], -tacke_slike[i][0]*tacke[i][1], -tacke_slike[i][0]*tacke[i][2]])
#svd, A=UDV^T, bitna nam je jedino matrica V i to njene kolone, jer je tu rezultat
_, _, V = LA.svd(matrica)
P_matrica_DLT = V[-1]*(-1)
return P_matrica_DLT
print("******* DLT *******")
P_matrica_DLT = dlt(tacke, tacke_projekcije)
print(f"Matrica preslikavanja za DLT algoritam sa %d tacke:" % n)
print(P_matrica_DLT.reshape((3,3)).round(decimals=5))
print()
#Poredjenje sa naivnim, stavljamo u if uslov za slucaj da nam je vrednost 0 i da ne bi dobili deljenje 0, posto ne znamo unapred vrednosti, inace treba da se odabere vrednost koja nije 0 pri deljenju i tako za svako poredjenje
P=P_matrica
PM=P_matrica_DLT
if ((PM[0]*P[0][0])!=0):
print("Poredjenje naivnog i DLT algoritma, na 5 decimala")
M=[(x / PM[0] * P[0][0]) for x in PM]
M=np.array(M).reshape(3,3).round(decimals=5)
print("Zakljucak:")
print(M)
print(M.round(decimals=5)==P.round(decimals=5))
print()
Python
Tab Width: 4
Ln 145, Col 12
INS
[Zadatak - Mozilla Firef...
jovana@jovana: ~/Des...
algoritmi.py (~/Deskto...

jovana@jovana: ~/Desktop/jovana

File Edit View Search Terminal Help

```
 [-0.      -0.26726  0.53452]]

Poredjenje naivnog i DLT algoritma, na 5 decimala
Zakljucak:
[[ 2.  0. -0.]
 [ 0.  2. -1.]
 [-0. -1.  2.]]
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]

******* DLT NORMALIZOVANI ******
T matrica
[[0.4496856  0.         1.27777778]
 [0.         0.4496856  0.22222222]
 [0.         0.         1.        ]]

T' matrica
[[0.61609012 0.         0.58333333]
 [0.         0.61609012 0.16666667]
 [0.         0.         1.        ]]

Matrica dobijena DLT normalizovanim algoritmom za 6 tacaka:
[[ 0.3603  -0.       0.     ]
 [-0.       0.3603  -0.18015]
 [ 0.      -0.18015  0.3603 ]]

Poredjenje naivnog i normalizovanog DLT, na 5 decimala
Zakljucak:
[[ 2. -0.  0.]
 [-0.  2. -1.]
 [ 0. -1.  2.]]
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
jovana@jovana:~/Desktop/jovana$
```

algoritmi.py (~/Desktop/jovana) - Pluma

File Edit View Search Tools Documents Help

Open Save Undo

algoritmi.py

```
209 def normalizacija(ps):
210     #1.centar mase (teziste)
211     centar_mase_x=sum([a[0] for a in ps]) / len(ps)
212     centar_mase_y=sum([a[1] for a in ps]) / len(ps)
213 
214     #2.translacija u koordinatni pocetak
215     tacke_p=[[a[0]-centar_mase_x, a[1]-centar_mase_y] for a in ps]
216     lambda1=srednje_rastojanje(tacke_p)
217 
218     #matrca translacije i skaliranja T=SG, S skaliranje, G translacija
219     T=np.array([[math.sqrt(2)/lambda1, 0, centar_mase_x * (-1)],
220                 [0, math.sqrt(2)/lambda1, centar_mase_y*(-1)],
221                 [0, 0, 1]])
222 
223     return T
224 
225 #racunanje koeficijenta lambda1 koji predstavlja srednje rastojanje tacaka
226 def srednje_rastojanje(tacke_p):
227     lambda1= sum([math.sqrt(tacka_p[0]*tacka_p[0] + tacka_p[1]*tacka_p[1]) for tacka_p in
    tacke_p]) / len(tacke_p)
228     return lambda1
229 
230 #algoritam dlt normalizacija  (P=T'^-1P^-T)
231 def dlt_norm(tacke, tacke_slike):
232   #matrice T i Tp su matrice normalizacije originalnih tacaka i njihovih slika
233     T=np.array(normalizacija(tacke)).reshape((3,3))
234     Tp=np.array(normalizacija(tacke_slike)).reshape((3,3))
235 
236     #Normalizacija tacaka i njihovih slika
237     tacke=np.transpose(tacke)
238     tacke_slike=np.transpose(tacke_slike)
239 
240     tackeN=np.transpose(T.dot(tacke))
241     tacke_slikeN=np.transpose(Tp.dot(tacke_slike))
242 
243     #Primena dlt algoritma na normalizovane tacke i slike
244     dlt=dlt(tackeN, tacke_slikeN)
245     dlt=np.array(dlt_matrica).reshape((3,3))
246 
247     #4. P=T'^-1P^-T
248     rezultat=  (LA.inv(Tp)).dot(dlt).dot(T)
249     return (rezultat, T, Tp)
```

Python Tab Width: 4 Ln 248, Col 44 INS

[Zadatak - Mozilla Firef... jovana@jovana: ~/Des... algoritmi.py (~/Deskto...