FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA
OSIJEK

Diplomski studij – Robotika i umjetna inteligencija

Računalna geometrija i robotski vid

Laboratorijska vježba 2

Podjela ravnina na poligone

Ivan Gudelj, DRB

Osijek, 2021.

Cilj vježbe:

Naučiti kako se pomoću programske biblioteke OpenCV može izvesti Delaunay triangulacija te kako se može izraditi jednostavno korisničko sučelje. Naučiti kako se koristi dvosmjerno povezani popis rubova.

Zadatak:

Izraditi konzolnu aplikaciju koja :

- generira 20 nasumično izabranih točaka unutar kvadrata dimenzije 400 piksela;
- generira Delaunay triangulaciju skupa točaka iz koraka 1;
- prikazuje triangulaciju dobivenu u koraku 2 u prozoru dimenzija 440 x 440;
- omogućuje izbor nekog od trokuta triangulacije klikom lijeve tipke miša;
- izabrani trokut prikazuje crvenom bojom, a njemu susjedne trokute (one koji s njim dijele zajedničku stranicu) prikazuje plavom bojom

Rješenje:

Najprije je bilo potrebno definirati podučje u kojem će se odvijati rad programa te njegova podjela na podpodručja (Subdivisions) što smo uradili kodom u nastavku. Također je bilo potrebno kreirati 20 nasumičnih točaka što smo uradili funkcijom koja je navedena dolje.

```
Unutar Maina:
   windowSize = 440
   picSize = 400
   numberOfPoints = 20
   windowName = "Gudelj - Delaunay Triangulation"
   subDivision = createSubDivPlain(picSize)
   image = np.full((windowSize, windowSize, 3), 255, dtype=np.uint8)
    .
    .
   createRandomPointsInPlain(subDivision, numberOfPoints, picSize)

#Funkcija čiji je zadatak kreiranje podjele slike na subdivizije
def createSubDivPlain(picSize):
   square = (0, 0, picSize, picSize)
   return cv2.Subdiv2D(square)

#Dodavanje točki u unaprijed podjeljenu ravninu
def createRandomPointsInPlain(subDivision, pointNumber, side):
   for i in range(pointNumber):
      point = (random.randint(1, side - 1), random.randint(1, side - 1))
      subDivision.insert(point)
```

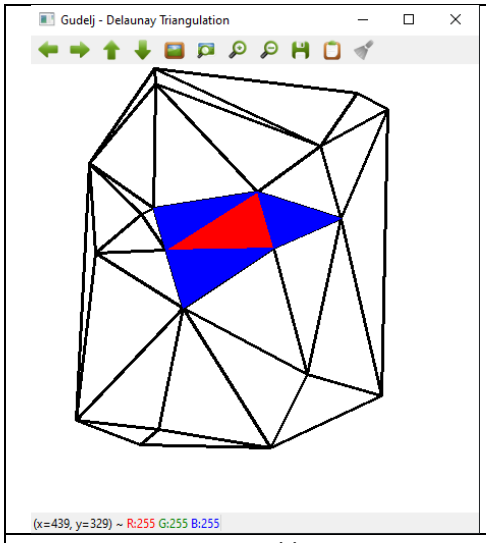
Nadalje je bilo potrebno iscrtati trokutove Delaunay triangulacije – funkcija drawDelaunayTriangulation() te prikazati ih što smo učinili s funckijama showDelaunayTriangulation() – Prikaz popisa trokutova iz subdivizije.

```
#Funkcija koja prima sve točke koji predstavljaju vrhove trokuta te se u njoj iscrtavaju
# trokutovi Delaunay triangulacije

def drawDelaunayTriangle(image, triangleVertices):
    lineColor = (0, 0, 0)
    lineThickness = 2
    intVertices = []
    for i in range(len(triangleVertices)):
        intVertices.append(int(triangleVertices[i]))
    point1 = (intVertices[0], intVertices[1])
    point2 = (intVertices[2], intVertices[3])
    point3 = (intVertices[4], intVertices[5])
    cv2.line(image, point1, point2, lineColor, lineThickness)
    cv2.line(image, point2, point3, lineColor, lineThickness)
    cv2.line(image, point3, point1, lineColor, lineThickness)

#Funkcija kojoj se predaje čista slika, podjela na subdivizije(područja),
# te ime prozora koja za zadatak ima prikazivanje Delaunay triangulacije

def showDelaunayTriangulation(image, subDivision, windowName):
    triangleList = subDivision.getTriangleList()
    for i in range(len(triangleList)):
        drawDelaunayTriangle(image, triangleList[i])
    cv2.imshow(windowName, image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

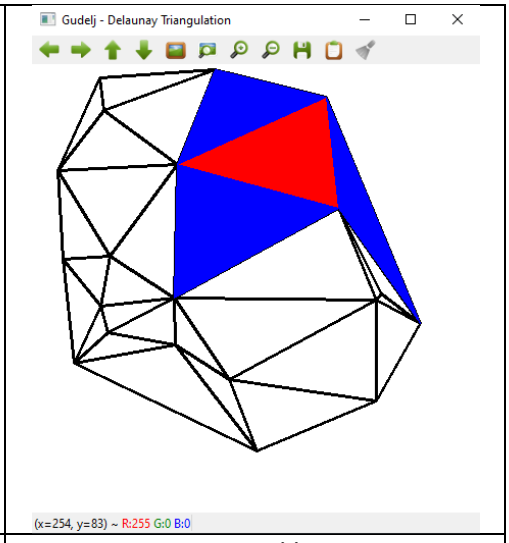
Još jedan od zadataka jest da se omogući takav event pritiskom tipke miša na jedan od trokuta da bi se susjedni trokutovi markirali kao takvi (Kliknut trokut crveni, susjedni trokuti plavi). Rješenje je implementirano u vidu onClickEvent() funkcije koja uz pomoć funkcija getTriangle() i getNextEdge() dobija informacije o susjednim trokutovima. Kod u nastavku objašnjava rješenje istog zadatka.

```
def onClickEvent(event, x, y, flags, userData):
    if event == cv2.EVENT_LBUTTONDOWN:
        subDivision = userData[0]
        image = np.copy(userData[2])
        windowName = userData[1]
        firstEdge = subDivision.locate((x, y))[1]
        mainTriangle = getTriangle(firstEdge, subDivision)
        triangle1 = getTriangle(subDivision.rotateEdge(firstEdge, 2), subDivision)
        secondEdge = getNextEdge(firstEdge, subDivision)
        triangle2 = getTriangle(subDivision.rotateEdge(secondEdge, 2), subDivision)
        thirdEdge = getNextEdge(secondEdge, subDivision)
        triangle3 = getTriangle(subDivision.rotateEdge(thirdEdge, 2), subDivision)
        cv2.fillConvexPoly(image, mainTriangle, (0, 0, 255))
        cv2.fillConvexPoly(image, triangle1, (255, 0, 0))
        cv2.fillConvexPoly(image, triangle2, (255, 0, 0))
        cv2.fillConvexPoly(image, triangle3, (255, 0, 0))
        cv2.imshow(windowName, image)
```

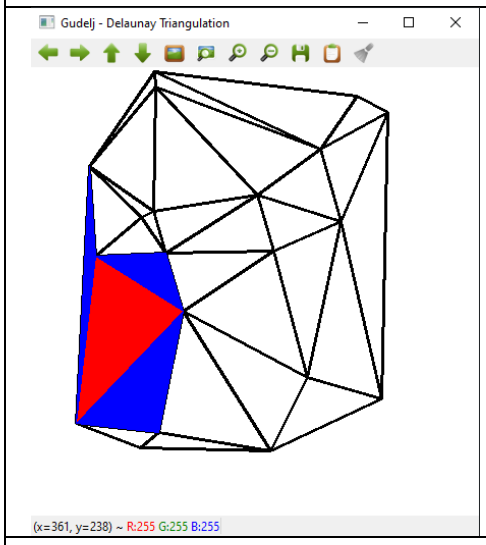
Prikaz konkretnog rješenja:



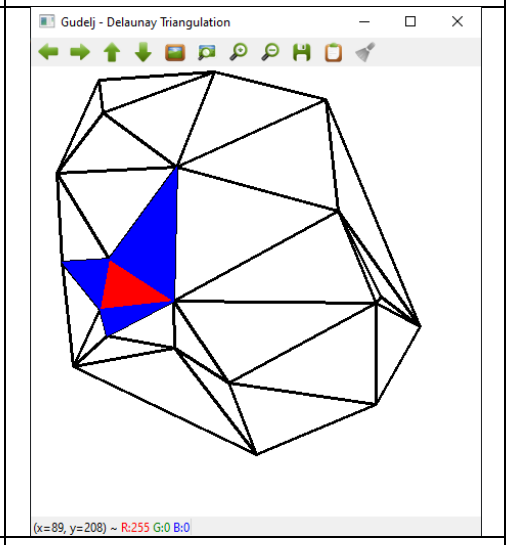
RUN1 - Slika1



RUN2 - Slika1



RUN1 - Slika2



RUN2 - Slika2