**Завдання**

**Завдання**:

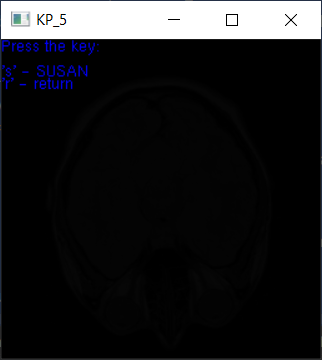
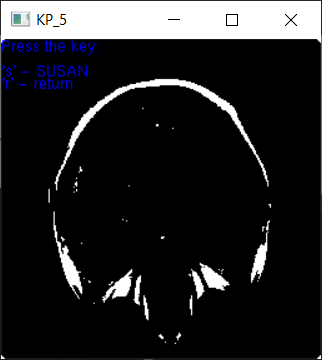
1. Ознайомитися з теоретичними відомостями реалізації алгоритмів сегментації медичних зображень, що засновані на пошуку границь.
2. Розробити програмний додаток для завантаження медичного зображення в форматі DICOM та виконання сегментації медичних зображень, що засновані на пошуку границь.
3. Розміри частини вінка програмного додатка для візуалізації графічних даних (без інтерфейсу користувача) мають відповідати розмірам завантаженого медичного зображення; завантажене медичне зображення має мати масштаб 100% (одному пікселю зображення відповідає один піксель екрана).
4. Створити події від клавіатури, при обробці яких можна окремо виконати заданий у варіанті алгоритм сегментації зображення із відображенням результатів його роботи та відновити оригінальне відображення завантаженого томографічного зрізу.
5. Виконати сегментацію медичного зображення, застосувавши контрастні до поточних кольори для відображення на початвковому зображенні отриманих границь (алгоритм сегментації та спосіб візуалізації отриманих границь виконати відповідно до заданих у варіанті).
6. Скласти і захистити звіт по роботі.

| **Номер варіанту** | **Алгоритм сегментації медичного об’єкта** | **Зв’язність пікселів границі** | **Візуалізація границі** |
| --- | --- | --- | --- |
| 8 | алгоритм найменшого однорідного сегмента, асимільованого ядром | 8 | внутрішня |

**Лістинг программи**

import math  
import operator  
import pickle  
  
from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
import pydicom  
import numpy as np  
from OpenGL.arrays.numpymodule import ARRAY\_TO\_GL\_TYPE\_MAPPING  
  
path\_file = "DICOM\_Image\_16b.dcm"  
  
  
class Image:  
 def \_\_init\_\_(self, path):  
 self.ds = pydicom.read\_file(path)  
 self.image\_pixels = np.array(self.ds.pixel\_array)  
 self.width, self.height = self.ds[0x280010].value, self.ds[0x280011].value  
  
 self.rows = 0; self.cols = 0  
 self.arr\_bits = self.ds[0x280100].value  
 self.is\_Susan = False  
 self.isOverlap = False  
  
  
 def init(self):  
 glClearColor(0, 0, 0, 0.0)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 gluOrtho2D(0, self.width, 0, self.height)  
  
 def display(self):  
 glClear(GL\_COLOR\_BUFFER\_BIT)  
 glColor3f(1, 1, 1)  
  
 draw\_pixel = np.copy(self.image\_pixels)  
 self.rows = len(draw\_pixel)  
 self.cols = len(draw\_pixel[0])  
 # перевіряємо чи потрібно виконувати фільтрацію  
  
 if self.is\_Susan:  
 draw\_pixel = self.SUSAN(draw\_pixel)  
  
 self.draw\_texture(draw\_pixel, GL\_LUMINANCE)  
 self.print\_text(0, self.height - 40, GLUT\_BITMAP\_HELVETICA\_12, "'r' - return")  
 self.print\_text(0, self.height - 30, GLUT\_BITMAP\_HELVETICA\_12, "'s' - SUSAN")  
 self.print\_text(0, self.height - 10, GLUT\_BITMAP\_HELVETICA\_12, "Press the key:")  
  
 glutSwapBuffers()  
  
 def print\_text(self, x, y, font, line):  
 glColor3f(0, 0, 1)  
 glPushAttrib(GL\_DEPTH\_TEST)  
 glRasterPos2d(x, y)  
 for i in line:  
 glutBitmapCharacter(font, ord(i))  
 glPopAttrib()  
  
 def draw\_texture(self, data, internal\_format):  
 gl\_type = ARRAY\_TO\_GL\_TYPE\_MAPPING.get(data.dtype)  
  
 glTexImage2D(GL\_TEXTURE\_2D, 0, internal\_format, self.width, self.height, 0, internal\_format, gl\_type, data)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)  
 glEnable(GL\_TEXTURE\_2D)  
  
 glBegin(GL\_QUADS)  
 glTexCoord2d(0.0, 0.0)  
 glVertex2d(0.0, 0.0)  
 glTexCoord2d(1.0, 0.0)  
 glVertex2d(self.width, 0.0)  
 glTexCoord2d(1.0, 1.0)  
 glVertex2d(self.width, self.height)  
 glTexCoord2d(0.0, 1.0)  
 glVertex2d(0.0, self.height)  
 glEnd()  
  
 glDisable(GL\_TEXTURE\_2D)  
  
 def similarityFunction(self, r0, r, t):  
 if math.fabs(r - r0) > t:  
 return 0  
 else:  
 return 1  
  
 def SUSAN(self, data):  
 data = self.normalization\_func(data, 0, 1)  
 mask\_radius = 3  
 output = np.copy(data)  
  
 for i in range(self.rows):  
 for j in range(self.cols):  
 v\_start = 0 if i - mask\_radius < 0 else i - mask\_radius  
 v\_end = self.rows - 1 if i + mask\_radius + 1 > self.rows else i + mask\_radius  
 h\_start = 0 if j - mask\_radius < 0 else j - mask\_radius  
 h\_end = self.cols - 1 if j + mask\_radius + 1 > self.cols else j + mask\_radius  
  
 similarity = 0  
 for v in range(v\_start, v\_end):  
 for h in range(h\_start, h\_end):  
 if (v == v\_start and h == h\_start) or (v == v\_start and h == h\_end - 1) or (v == v\_end - 1 and h == h\_start) or (v == v\_end - 1 and h == h\_end - 1):  
 similarity += 1  
 else:  
 similarity += self.similarityFunction(output[i][j], output[v][h], 40)  
  
 output[i][j] = similarity  
  
 geo\_threshold = output.max() / 2  
 borders\_map = np.copy(output)  
  
 for i in range(len(output)):  
 for j in range(len(output[0])):  
 if output[i][j] < geo\_threshold:  
 borders\_map[i][j] = geo\_threshold - output[i][j]  
 else:  
 borders\_map[i][j] = 0  
  
 for i in range(len(output)):  
 for j in range(len(output[0])):  
 if borders\_map[i][j] != 0:  
 borders\_map[i][j] = 255  
  
 return np.array(borders\_map, np.uint8)  
  
 # функція виконує нормальзацію  
 def normalization\_func(self, pixels, p\_min, p\_max):  
 pixel\_max = int(float(p\_max \* pixels.max()))  
 pixel\_min = int(float(p\_min \* pixels.max()))  
  
 new\_min = 0  
 new\_max = np.iinfo(np.int8).max#255  
 normalization = []  
  
 for row in pixels:  
 new\_row = []  
 for pixel in row:  
 new\_pixel = ((pixel - pixel\_min) / (pixel\_max - pixel\_min)) \* (new\_max - new\_min)  
 if new\_pixel <= 0:  
 new\_pixel = 0  
 if new\_pixel > new\_max:  
 new\_pixel = new\_max  
 new\_row.append(new\_pixel)  
 normalization.append(new\_row)  
  
 return normalization  
 # return np.array(normalization, np.uint8)  
  
 def keyboard\_func(self, my\_key, x, y):  
 key = unicode(my\_key, errors='ignore')  
 if key == 's':  
 self.is\_Susan = True  
 if key == 'r':  
 self.is\_Susan = False  
 self.display()  
  
  
def init\_window(width, height):  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - width) // 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - height) // 2)  
 glutCreateWindow('KP\_5')  
  
  
def main():  
 glutInit()  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
 file = Image(path\_file)  
 init\_window(file.width, file.height)  
 file.init()  
 glutDisplayFunc(file.display)  
 glutKeyboardFunc(file.keyboard\_func)  
 glutMainLoop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат

**Контрольні запитання**

1. *В чому полягає основна ідея методів сегментації, заснованих на пошуку границь об’єктів на зображенні?*

Такі методи засновані на принципі, що різним об’єктам на зображенні відповідають ділянки, які мають приблизно однакові значення яскравості, а на границях пікселі, яскравість яких істотно знінюється.

1. *Як вивести та спростити для програмної реалізації формули DoG- та LoG- операторів?*

Виведемо:

Спочатку зображення сгладжується згорткою з Гаусівським ядром певної ширини

де,

Разом із іншим , можемо отримати друге сгладжування зображення:

Ми можемо показати, що різниця двох сгладжених гаусівських зображень, називається різницею гіусів (*DoG-*), яку ми використаємо для визначення ребер зображення:

*DoG* як оператор або ядро згортання визначається як:

Спростити вираз можна шляхом апроксимації непреривного виразу зверху.

The Laplacian of Gaussian це багатовимірне узагальнення вейвлета Рікера. Для його визначення необхідно взяти двовимідний Laplacian of the Gaussian:

1. *Що необхідно виконати, щоб зробити знайдені краї тонкими?*

Необхідно виконати пошук локальних максимумів, та встановити тільки їх як границі зображенян.

1. *Що визначає функціонал енергії моделі активного контура?*

Енергія контуру залежить від форми та розміру контуру та від його положення на зображені, вона записується як сума функціоналів «внутрішньої» та «зовнішньої» енергії контуру. Визначається наступним рівнянням:

1. *Зовнішні і внутрішні сили моделі активного контура, їх властивості.*

Зовнішня характеризує невідповідність контура зображення, а внутрішня реалізу вимоги до гладкості контуру.

Внутрішня:

Зовнішня:

Внутрішня енергія визначає форму кривої та залежить від першої та дрігої похідної радіус-вектора точки на кривій за параметром *s*. Зовнішня змущує активний контур рухатися в напрямку шуканої границі об’єкта на зображенні та визначається даними зображення і набуває занчення границь.

1. *Що задають коефіцієнти α, β та γ у визначенні енергії внутрішніх і зовніщніх сил моделі контурів?*

Коефіцієнт *α* описує кіт скорочення, чим він більше тим більше скорочення в напрямку дії сили. Коефіціент *β* регулює швидкість зміни кривої в напрямку, перпендикулярному її границі, якщо він має високе значення, крива інтенсивно чинить опір згинанню, а низький дозволяє кривій згинатися. Коефіцієнт *γ* це зважений коефіцієнт діїі зовнішньої сили.

1. *Навести та пояснити рівняння Е’лера-Лагранжа для моделі активних контурів.*

Рівняння Ейлера-Лагранжа:

Його можна представити як рівняння балансу сил:

де, та – внутрішня та зовнішні сили моделі відповідно.

1. *Навести та пояснити формулу розрахунку значення функції подібності яскравсті пікселів маски до яскравості ядра під час використання найменшого однорідного сегменту, асимільованого ядром.*

Робота алгоритму заключається у побудові навколо кожного пікселя маски, де центральний піксель – це ядро, пікселі в межах маски утворюють зону *USAN* – однорідний сегмент асимільований ядром. Для кожної зони розраховуються зрначення функції подібності яскравості пікселів маски до яскравості ядра:

де – координати ядра маски на зображенні, – координати пікселів маски, та – яскравості пікселів з координатами та , – заданий поріг для значення різниці яскравості.

1. *Навіщо і як проводять визначення карти границь в процесі сегментації з використанням найменшого однорідного сегмента, асимільвуваного ядром?*

Для виявлення границь розраховують площину зони *USAN*:

Та визначають карту границь для кожного пікселя:

де – «геометричний поріг» виявлення границі (частіше за все рівний половині максимально можливого значення площі зони *USAN*).

Алгоритм находит границу на основании того насколько похожи яркості пикселей, и находит границу там где площать похожего участка достигает локального минимума, для повишения устойчивости работи алгоритма карту границ присваивают к определенному пикселю.

1. *Навести декілька способів зберігання результатів сегментації методами, що заснованрі на пошуку границь об’єктів на зображенні.*

При такій сегментації зручно використовувати лінійні списки. Подавши криву, яка утволює границю, у вигляді відрізків, які формують полігом апроскимації, то можна не зберігати інформацию про кожній піксель і тим самим зекономити пам’ять та спростити алгоритм обробки границь.