**Завдання**

**Завдання**:

1. Ознайомитися з теоретичними відомостями реалізації алгоритмів сегмент ації медичних зображень, щ засновані на формуванні однорідних ділянок.
2. Розробити пограмний додаток для завантаження медичного зображення в форматі DICOM та виконання сегментацію седичних зображень, що засновані на формуванні однорідних ділянок.
3. Розміри частини вікна програмного застосунку 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 та відновити оригінальне відображення завантаженого томографічного зрізу.
5. Виконати сегментацію медичного зображення, застосувавши контрастні до поточних кольори для відображення на початковому зображенні отриманих сегментів (алгоритм сегментації та структуру для збереження отриманих сегментів використати відповідно до заданих у варіанті).
6. Скласти і захистити звіт.

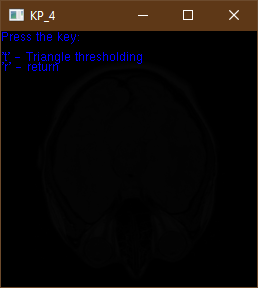
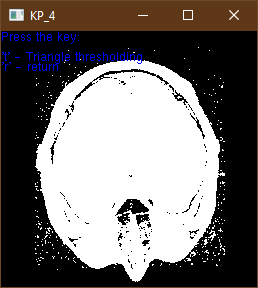
**Варіант виконання завдання**

| **Номер варіанту** | **Алгоритм сегментації седичного об’єкта** | **Структура для збереження результатів сегмент ації** |
| --- | --- | --- |
| 8 | алгоритм трикутника для методу порогової фільтрації | таблиця властивостей |

**Лістинг программи**

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.arr\_bits = self.ds[0x280100].value  
 self.is\_Segmentation = 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)  
 # перевіряємо чи потрібно виконувати фільтрацію  
 if self.is\_Segmentation:  
 draw\_pixel = self.traingle\_thresholding(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, "'t' - Triangle thresholding")  
 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)  
  
 # функція виконує фільтрацію  
 def traingle\_thresholding(self, data):  
 data\_norm = self.normalization\_func(data, 0, 1) # нормалізовані дані  
  
 # будуємо гістограму  
 hist\_value = {key: (data\_norm == key).sum() for key in range(data\_norm.min(), data\_norm.max() + 1)}  
 hist = list(hist\_value.values())  
 b\_min, b\_max = np.argmin(hist), np.argmax(hist)  
  
 # розраховуємо найбільшу відстань між лінієї(від мінімуму до максимуму гістограми) та рівнями яскравості  
 x = [el for el in reversed(range(b\_max, b\_min + 1, 1))]  
 dist = {key: self.distance(b\_min, b\_max, key, hist[b\_min], hist[b\_max], hist[key]) for key in x}  
 tresholding = max(dist.items(), key=operator.itemgetter(1))[0] # порогове значення  
  
 # сегментоване зображення  
 mask = np.copy(data\_norm)  
 for i in range(len(mask)):  
 for j in range(len(mask[i])):  
 if mask[i][j] < tresholding:  
 mask[i][j] = 0  
 elif mask[i][j] >= tresholding:  
 mask[i][j] = 255  
  
 self.save(data\_norm, mask)  
 return mask  
  
 # функція визначає відстань між лінієї(від мінімуму до максимуму гістограми) та рівня яскравості  
 def distance(self, x\_min, x\_max, x\_key, y\_min, y\_max, y\_key):  
 A, B, C = self.equation\_line(x\_min, x\_max, y\_min, y\_max)  
 answer = np.abs(A \* x\_key + B \* y\_key + C) / np.sqrt(A\*\*2 + B\*\*2)  
 return answer  
  
 # функція розраховує парамеьтри для рівняння прямої  
 def equation\_line(self, x1, x2, y1, y2):  
 A = y2 - y1  
 B = -(x2 - x1)  
 C = -x1 \* (y2 - y1) + y1 \* (x2 - x1)  
 return A, B, C  
  
 # функція виконує збереження данних  
 def save(self, data\_norm, mask):  
 # утворюємо таблицю властивостей  
 table = [{'x': x, 'y': y, 'mask': mask[x, y], 'value': data\_norm[x, y]}  
 for x in range(data\_norm.shape[0]) for y in range(data\_norm.shape[1])]  
  
 with open('filename.pickle', 'wb') as handle:  
 pickle.dump(table, handle, protocol=pickle.HIGHEST\_PROTOCOL)  
  
 # 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  
 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 np.array(normalization, np.uint8)  
  
 def keyboard\_func(self, my\_key, x, y):  
 key = unicode(my\_key, errors='ignore')  
 if key == 't':  
 self.is\_Segmentation = True  
 if key == 'r':  
 self.is\_Segmentation = 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\_4')  
  
  
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 формуванні однорідних ділянок?*

Основна ідея полягає у тому, що при формуванні однорідних ділянок, методи відштовхуться від неоднорідності границь, або від однорідності усередині ділянки.

1. *Перелічити основні обмеження методі порогової фільтрації. Які з них можна вирішити і як?*

Основним обмеження методів порогової фільтрації виступає саме визначення порогу для фільтрації. Визначення порогу можна виконати за допомогою дискримінантног окритерію.

1. *В чому полягає різниця проведення глобальної та локальної порогової фільтрації?*

При проведенні глобальної порогової фільтрації порогове значення визначається для всіх пікселів зображення, а під час локальної – для кожного піксеьля окремо (отримується на основі значень яскравості його сусідів у деякому фіксованому околі).

1. *Внутрішньокласова дисперсія, її визначення.*

Внутрішньокласова дисперсія – це зважена сума дисперсій класів; функціонал якості розподілу об’єктів, для відбору найбільш якісної класифікації.

1. *Які існують алгоримтми покращення результатів порогової фільтрації алгоритмом Ніблека? Пояснити їх суть.*

Алгоритм Ніблека погано працює для зображень з неоднорідним фоном, можлива поява хибних об’єктів. Для вирішення даної проблеми були розроблені модифіковані версії алгоритму (Саволи, Вольфа, Фенга, алгоритм NICK), які в основі своїй використовують додаткові параменри для усунення проблеми.

1. *Використовуючи формулу розрахунку ентропії Шенона, вивести формули для визначення ентропій чорного та білого для порогової фільтрації алгоритмом максимальної ентропії Капура.*

Визначимо, що g = 0,...,G діапазон значень усів порогів бінаризації, а p(g) – вірогідність того, що випадкова змінна дорівнює значеню g. Для зображення розмыром N оцінка вірогідності буде визначатиня:

, де – кількість пікселів, інтенсивність яких рівна g.

Тоді функція розподілу вірогідності :

Основна задача для різних метолів сегментації це визначення передньо та заднього плана, тому визначимо через , оцінку вірогідності для заднього плану, а через , для переднього, де – порогове значення бінаризації, то сумарні функції будуть визначатися, як:

На основі цих функції ми можемо вівести формули математичного очікування та дисперсії:

Отже, єнтропію можрна визначити як:

1. *Що може бути використано в якості критерію зупинки роботи алгоритму сегментації з використанням кластеризації методом k-середніх? Відповідь пояснити для кожного із критеріїв.*

Якщо не відбуваються зміни клатерів та центрів, або вони мінімальні, то в якості критерія зупинки може бути відсутність переходів пікселів з одного кластера в інші після виконання дії певної ітерації алгоритму.

Також критерієм може бути виконання умови оптимальності, зазвичає це мінімізація середньокрадратичної похибки розбиття пікселів на класи.

Або досягнення максимальної заданої кільксоті ітерацій.

1. *Які існують алгоритми сегментації з використанням кластеризації, окрім методом k-середніх? Навести декілька алгоритмів та їх порівняльну характеристику.*

Алгоритми розділення (Divisive clustering):

* На початку все зображення – це один кластер;
* Знаходимо новий окремий кластер який має найбільшу міжкластерну відстань;
* Розбиваємо початковий кластер на два.

Іерархічні методи (Hierarchical clustering):

* На початку кожен піксель – окремий кластер;
* Знаходимо пару кластерів, які мають найменшу міжкластерну відстань;
* Робимо з цих кластерів один.

Процедура середнього зсуву (Mean shift clustering):

* Ініціалізація вікна відбувається на окремій точці (об’єкті);
* Виконується послідовний зсув;
* Кластери відповідають вікнам, які знаходяться біля одного піку.

1. *Визначити, яка з властивостей сегментів, що відповідають однорідним ділянкам, визначає зв’язність, яка діз’юнктність, яка здійсненність, яка повнота, а яка здатність до поділу (сегментабельність).*

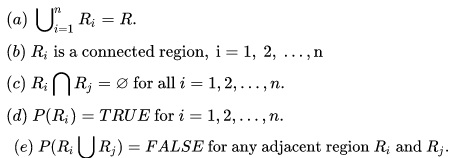
Зв’язність – усі пікселі, що на лежать певній ділянці, мають бути пов’язані між собою;

Діз’юнктність – після проведення сегментації різні ділянки не мають мати спільних пікселів;

Здійсненність – усі пікселі певної ділянки мають мати як мінімум одну загальну властивість, будь-яка ділянка є однорідною за деяким критерієм;

Повнота – кожен піксель зображення має відповідати певній ділянці;

Сегментабельність – різні ділянки мають різні властивості, тобто гарантується, що будь-яке обєднання двох існуючих ділянок не призведе до утворення нової ділянки, адже це не буде відповідати критерію однорідності.



1. означає, що сегментація має бути **повна**;
2. вимагає, щоб кожна точка сегменту була **зв’язана**, за певною визначеною ознакою;
3. сегменти не повинні перетинатись (**сегментованість**);
4. пікселі одного сегменту повинні відповідати певній властивості (**здійсненність**);
5. сегменти відрізняються відносно заданого предикату (**дизюнктивність**).
6. *Навести декілька способів зберігання результатів сегментації методами, що засновані на формуванні однорідних ділянок.*
7. Маркована мапа
8. Талиця властивостей
9. Квадратні дерева