**Завдання**:

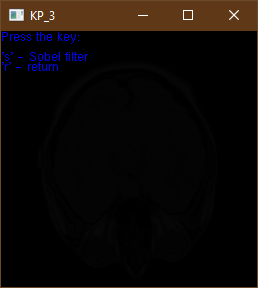
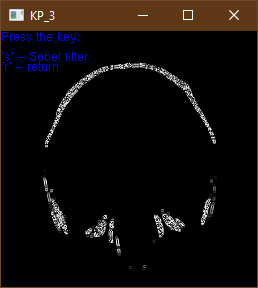
1. Ознайомитися з теоретичними відомостями реалізації фільтрації зображення
2. Розробити повноекранній програмний додаток для завантаження медичного зображення в форматі DICOM та виконання фільтрації.
3. Розміри частини вікна програмного додатка для візуалізації графічних даних (без інтерфейсу користувача) мають відповідати розмірам завантаженого медичного зображення; завантажене медичне зображення має мати маcштаб 100% (одному пікселю зображення відповідає один піксел екрана).
4. Створити події від клавіатури, при обробці яких можна окремо виконати заданий у варіанті вид фільтрації, а також відновити оригінальне зображення.
5. Виконати фільтрацію медичного зображення (вид фільтрації та спосіб обчислення граничних пікселів використати відповідно до заданих у варіанті).
6. Скласти і захистити звіт по роботі

| Номер варіанту | Вид фільтрації | Обчислення граничних пікселів |
| --- | --- | --- |
| 8 | Фільтр Собеля | довизначення за межами циклічним віддзеркаленням. |

**Лістинг программи**

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\_Sober = 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\_Sober:  
 draw\_pixel = self.filtration(draw\_pixel, 2)  
  
 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' - Sobel filter")  
 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 filtration(self, data, border\_size):  
 arr\_filter = np.zeros((self.height, self.width)) # відфільтрований масив  
  
 data\_norm = self.normalization\_func(data, 0.7, 1) # нормалізовані дані  
 data\_pad = self.border\_pixels(data\_norm, border\_size) # нові дані з доданими границями  
  
 for i in range(self.height):  
 for j in range(self.width):  
 # викликаємо фільтр для кожного пікселя  
 arr\_filter[i, j] = self.filter\_Sobel(data\_pad[i:i + 3, j:j + 3])  
  
 return np.array(arr\_filter, 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 = 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 np.array(normalization, np.uint8)  
  
 # довизначаємо границі пікселів за межами зображення  
 def border\_pixels(self, data, size):  
 height, weigh = data.shape  
  
 # розміри збільшеного масиву  
 new\_height, new\_weigh = height + 2 \* size, weigh + 2 \* size  
 # новий массив  
 new\_data = np.zeros((new\_height, new\_weigh))  
  
 # довизначаємо границі пікселів за межами зображення  
 for i in range(new\_height):  
 for j in range(new\_weigh):  
 new\_data[i][j] = data[(i - size) % height][(j - size) % weigh]  
  
 return np.array(new\_data, np.uint8)  
  
 # фільтр Собеля  
 def filter\_Sobel(self, data):  
 x\_mask = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])  
 y\_mask = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])  
  
 return np.sqrt((x\_mask \* data).sum()\*\*2 + (y\_mask \* data).sum()\*\*2)  
  
 def keyboard\_func(self, my\_key, x, y):  
 key = unicode(my\_key, errors='ignore')  
 if key == 's':  
 self.is\_Sober = True  
 if key == 'r':  
 self.is\_Sober = 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\_3')  
  
  
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. *Що таке фільтрація зображень, навіщо потрібно її виконувати і коли?*

Фільтрація – це процес заміни яскравості кожної точки початкового зображення деяким іншим (усередненим) значенням яскравості, що можна вважати спотворенням зображення.

Незалежно від того, яким методом було отримано зображення, іноді на ньому можуть бути пришутні перешкоди (шуми), або не явно видно деякі елементи на зображення. Саме для того аби позбутися перешкод, або виділити певні частини зображення використовується фільтрація.

1. *Чим локалька фільтрація зображення відрізняється від глобальної?*

Під час виконання локальної фільтрації розрахунки проводяться не по всіх точках зображення, а порівняно по невеликому околу кожної точки зображення. Такий підхід дозволяє збільшити швідкість проведення фільтрації, а також зручність у роботі з цифровими зображеннями.

1. *Як розрахувати коефіцієнти маски фільтра Гауса? Навести приклад розрахунків.*

Під час використання фільтра Гаусса в ньому всередині буде розмита пляма зі зменшенням яскравості від середини до границь маски.

Маска формується за наступними формулами:

де M та N дозмыр маски, а σ – середньоквадратичне выдхилення розподылу Гаусса.

1. *Диференціальні оператори визначення границь для безперервних функцій та для дискретних зображень.*

До диференціальних операторів визначення границь на зображенні відносяться Робертса, Превіта, Собеля, ілотропний фільтр.

За допомногою цих операторів можгна визначити наближене значення першої частинної похідної зміни інтенсивності в горизонтальному () та вертикальному () напрямках.

Далі визначаємо магнітуду градієнта для пікселя:

Та напрямок градієнта:

Маска фільтра Робертса визначаэться:

Маска фільтра Превіта визначаэться:

Маска фільтра Собеля визначаэться:

Маска ізотропного фільтра визначаэться:

1. *Як провести апроксимацію перших частинних похідних методом скінчених різниць?*

Апроксимація перших частинних похідних методом скінченний різниць виконується за формулами:

де – функція яскравості.

Похідні пропорційні швидкості зміни яскравості у віповідних напрямках.

Проте коли ми обираємо границі напрямок яких є довільним можна використовувати магнітуду градієнта (модуль градієнта яскравості).

Також використовується напрям градієнта.

Оператори визначення границь зображення з використанням перших похідних задаються за допомогою градієнта фільтрів. В разі їх застосування ділянки зображення з однорідними значеннями обнуляються, а ділянки зміни яскравості виділяються.

1. *Як провести апроксимацію других частинних похідних методом скінчених різниць?*

Формули для апроксимації других частинних похідних:

Другі похідні спрямовані на підкреслення перепаду яскравості. Для отримання положення границь на зображенні необхідно після фільтрації взяти всі пікселі модуль значень яких перевищує деякий поріг.

1. *Нелінійна фільтрація медичних зображень, її переваги над лінійною.*

Використовуючи лінійну фільтрацію зображення відбувається зменшення шумів і водночас розмиваються межі між областями. Нелінійна фільтрація направлена на підвищення чіткості границь, що відбуваються проходженням вікна по зображенню та виконанням нелінійних перетворень відліків зображення за певною маскою фільтра.

Поділяються на:

* Медіанна фільтрація
* Фільтри «максимум» та «мінімум»
* Оператори математичної морфології

1. *Пояснити на прикладі, коли маски медіальнної фільтрації усувають неточкові імпульсні перешкоди, а коли ні.*

Медіанна фільтрація буде усувати неточкові імпульсні перешкоди у випадку коли вони менші ніж половина розміру маски, якщо ж необхідно усунути таку перешкоді необхідно збільшити розмір вікна.

1. *Що таке сепарабельність фільтра? Навести приклад таких фільтрів для обробки забражень та визначити, в чому особливості їх програмно-алгоритмічної реалізації?*

Сепарабельність фільтка – це можливість розкласти ядро фільтру на добуток декількох фільтрів.

Приклад:

Особливістю програмно-алгоритмічної реалізації може бути створення такого фільтру за допомогою множення, а не задавання одразу.

1. *Перелічити морфологічні оператори, на основі яких фільтрів і як вони побудовані?*

Базові морфологічні оператори:

* Операція ерозії – це фільтр «Мінімум»;
* Операція дилатації – це фільтр «Максимум»;
* Операція відкриття – послідовне застосування звужуюючого, а потім розшируючого операторів;
* Операція закриття - послідовне застосування розширюючого, а потім звужуючого операторів;
* Визначення границь – дозволяє знаходити і виділити елементи заданої структури;
* Гранулометрія – вирівнювання та класифікація об’єктів зображення за однією з наявних характеристик;
* Скелетизація – однозначний опис об’єкта у вигляді деякого остову.