****

**І. Мета роботи:**

дослідити принципи та особливості практичного застосування технологій сегментації та кластеризації цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

**ІІ. Завдання:**

Реалізація проекту триває та спрямовано на збільшення функціональності

програмної компоненти Лабораторія провідної ІТ-компанії реалізує масштабний проект розробки універсальної платформи з цифрової обробки зображень для задач Computer Vision. Платформа передбачає розташування back-end компоненти на власному хмарному сервері з наданням повноважень користувачам заздалегідь адаптованого

front-end функціоналу універсальної платформи. Цим формується унікальна для потреб замовника ERP система з технологіями Computer Vision Замовниками ресурсів платформи є: державні та комерційні компанії, що розробляють медичне обладнання з діагностування захворювань за візуальною інформацією; автоматизації аграрного бізнесу в аспекті обліку посівних територій за даними з БПЛА; візуального контролю безпекових заходів на об’єктах критичної інфраструктури: аеропорти, торгівельно-розважальні центри, житлові комплекси тощо.

|  |  |  |
| --- | --- | --- |
| Місяць народження | Джерела даних ДЗЗ | Технічні умови |
| 1-6 | 1. Оперативні:  <https://apps.sentinel-hub.com/eo-browser/?zoom=14&lat=52.04212&lng=29.27444&themeId=WILDFIRES-NORMAL-MODE&visualizationUrl=https%3A%2F%2Fservices.sentinel-hub.com%2Fogc%2Fwms%2Faae18701-6b25-4001-8b2a-b98a1b3806c1&datasetId=S2L2A&fromTime=2022-03-16T00%3A00%3A00.000Z&toTime=2022-03-16T23%3A59%3A59.999Z&layerId=1_FALSE-COLOR>  2. Високоточні:  <https://www.google.com.ua/maps> | Район спостереження –  обрати самостійно.  Об’єкти ідентифікації –  обрати самостійно.  Дата оперативних даних –  обрати самостійно.  Метод і технологія  кластеризації / сегментації  – повинні забезпечувати  можливість розрізнення та  ідентифікацію обраних  об’єктів спостереження. |

І – максимально 7 балів, функціонал скрипта реалізовано у повному обсязі, п.6

технічних вимог (ідентифікація об’єкта) реалізовано шляхом візуального порівняння

контурів.

### ІІІ. Результати виконання лабораторної роботи. **5.3. Результати виконання лабораторної роботи**

**5.3.1. Синтезована математична модель**

Для реалізації перетворень (обертання, переміщення, масштабування) ромб

* **:**

**5.3.2. Результати архітектурного проектування та їх опис**

**5.3.3. Опис структури проекту програми**

**5.3.4. Результати роботи програми відповідно до завдання**

**5.3.5. Програмний код, що забезпечує отримання результату**

*from* graphics *import* \*  
*import* time  
*import* numpy *as* np  
*import* math *as* mt  
  
*#---------------- Формування та відображення статичного ромба ------------------------*xw = 600; yw = 600; st = 100 *# Розміри графічного вікна та параметри перетворень  
  
# Розміри ромба*x1 = xw//2; y1 = yw//2 - st  
x2 = xw//2 + st; y2 = yw//2  
x3 = xw//2; y3 = yw//2 + st  
x4 = xw//2 - st; y4 = yw//2  
  
*def* draw\_initial\_shape(win):  
 obj = Polygon(Point(x1, y1), Point(x2, y2), Point(x3, y3), Point(x4, y4))  
 obj.setFill('blue')  
 obj.draw(win)  
 *return* obj  
  
*#------------------------- Циклічне обертання ромба ------------------------  
  
def* rotate\_shape(win, obj):  
 win.setBackground('white')  
 coords = np.array([[x1, y1, 1], [x2, y2, 1], [x3, y3, 1], [x4, y4, 1]])  
 center\_x, center\_y = xw // 2, yw // 2  
  
 *def* rotate(coords, angle, center):  
 angle = mt.radians(angle)  
 cx, cy = center  
 transformation\_matrix = np.array([  
 [mt.cos(angle), -mt.sin(angle), 0],  
 [mt.sin(angle), mt.cos(angle), 0],  
 [0, 0, 1]  
 ])  
 translated\_coords = coords - np.array([cx, cy, 0])  
 rotated\_coords = translated\_coords.dot(transformation\_matrix.T)  
 final\_coords = rotated\_coords + np.array([cx, cy, 0])  
 *return* final\_coords  
  
 *for* \_ *in* range(72):  
 *if* win.checkMouse():  
 obj.undraw()  
 *return* time.sleep(0.1)  
 obj.undraw()  
 coords = rotate(coords, 5, (center\_x, center\_y))  
 obj = Polygon(Point(coords[0, 0], coords[0, 1]),  
 Point(coords[1, 0], coords[1, 1]),  
 Point(coords[2, 0], coords[2, 1]),  
 Point(coords[3, 0], coords[3, 1]))  
 obj.setFill('green')  
 obj.draw(win)  
 obj.undraw()  
  
*#------------------------- Переміщення ромба ------------------------  
  
def* move\_shape(win):  
 win.setBackground('white')  
 obj = draw\_initial\_shape(win)  
 dx, dy = 5, 5  
  
 *for* \_ *in* range(50):  
 *if* win.checkMouse():  
 obj.undraw()  
 *return* time.sleep(0.1)  
 obj.move(dx, dy)  
 obj.undraw()  
  
*#------------------------- Масштабування ромба ------------------------  
  
def* scale\_shape(win):  
 win.setBackground('white')  
 obj = draw\_initial\_shape(win)  
 coords = np.array([[x1, y1, 1], [x2, y2, 1], [x3, y3, 1], [x4, y4, 1]])  
 center\_x, center\_y = xw // 2, yw // 2  
 scale\_factor = 1.05  
 scale\_limit = 1.5  
 scale\_step = 0.05  
 scale\_cycles = 3  
  
 *for* \_ *in* range(scale\_cycles):  
 increasing = *True  
 for* step *in* range(12):  
 *if* win.checkMouse():  
 obj.undraw()  
 *return* time.sleep(0.2)  
 obj.undraw()  
 factor = scale\_factor *if* increasing *else* 1 / scale\_factor  
 scaling\_matrix = np.array([  
 [factor, 0, center\_x \* (1 - factor)],  
 [0, factor, center\_y \* (1 - factor)],  
 [0, 0, 1]  
 ])  
 coords = coords.dot(scaling\_matrix.T)  
 obj = Polygon(Point(coords[0, 0], coords[0, 1]),  
 Point(coords[1, 0], coords[1, 1]),  
 Point(coords[2, 0], coords[2, 1]),  
 Point(coords[3, 0], coords[3, 1]))  
 obj.setFill('red')  
 obj.draw(win)  
  
 *if* step == 5:  
 increasing = *False* obj.undraw()  
  
win = GraphWin("2-D трансформації", xw, yw)  
obj = draw\_initial\_shape(win)  
win.getMouse()  
obj.undraw()  
rotate\_shape(win, obj)  
win.getMouse()  
move\_shape(win)  
win.getMouse()  
scale\_shape(win)  
win.getMouse()  
win.close()

**5.4. Висновки**

В результаті виконання лабораторної роботи було реалізовано алгоритми 2D-перетворень геометричних об'єктів у графічному вікні. Було досліджено та застосовано математичні моделі для обертання, переміщення та масштабування ромба. Розроблена програма коректно виконує всі необхідні перетворення, анімація працює плавно, зміна стану об'єкта відбувається за допомогою натискання миші. Всі завдання лабораторної роботи виконані у повному обсязі.