รายงาน

การวิเคราะห์ภาพพระพุทธรูปรอบองค์พระปฐมเจดีย์

Analysis of Buddha images around the Phra Pathom Chedi

จัดทำโดย

640710806 ธนบูรณ์ ตันติภิรมย์

640710819 ภัทรพล อุ่นกำเนิด

640710820 ราม ชวาลธวัช

640710830 สุธี ชื่นชม

640710917 ธีรภาพ นาสุข

เสนอ

ผศ. โอภาส วงษ์ทวีทรัพย์

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชาการรวบรวมและกลั่นกรองข้อมูล

สาขาวิทยาการข้อมูล คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

ภาคเรียนที่ 2 ปีการศึกษา 2565

สารบัญ

บทคัดย่อ

การจัดการโครงงานเรื่อง การวิเคราะห์ภาพพระพุทธรูปรอบองค์พระปฐมเจดีย์ มีการนำ library ที่ ได้ศึกษาเรียนรู้จากรายวิชา Getting and cleaning data และ ยังมีการศึกษาหาความรู้เพิ่มเติมเพื่อให้ได้ข้อมูลที่ถูกต้องมากขึ้น และ ในการจัดทำโครงการครั้งนี้ได้มีการใช้ภาษาโปรแกรม โดยใช้ Python programming language

ผลจากการศึกษา ได้ข้อสรุปว่าโครงการครั้งนี้ประมวลผมออกมาได้เป็นอย่างดี และสามารถนำไปศึกษาและใช้ได้จริง

คำสำคัญ

- Python

- PowerPoint

- Microsoft Word

- PIL

- PyThaiNLP

ที่มาและความสำคัญของปัญหา

พระพุทธรูปรอบองค์พระ เรานำมาประมาณ 96 องค์ แต่ละองค์ก็เป็นพระใหม่พระเก่าซึ่งมีปรางหลายปรางหลายยุคสมัย ปัญหาที่พบเจอ ข้อมูลซับซ้อน ช่วงแรกอาจจะไม่เข้าใจถึงองค์ประกอบต่าง ๆ ของภาพพระพุทธรูป ความเชี่ยวชาญในการวิเคราะห์ภาพอาจจะต้องใช้เวลาในการศึกษาและเรียนรู้เพิ่มเติมเกี่ยวกับการวิเคราะห์ภาพ ภาพนั้นไม่ชัดเจนดีพอ บางครั้งเกิดจากการซูมหรือถ่ายภาพไม่ดีพอ ทำให้การวิเคราะห์ลำบากขึ้น

วัตถุประสงค์

1. เพื่อพัฒนาทักษะในด้านการเขียนโปรแกรมในการทำงานเพื่อนำไปต่อยอดได้ในอนาคต
2. เพื่อตรวจสอบว่าพระในองค์พระชื่ออะไรความแม่นยำ

ขอบเขตของงานที่จะทำ

ตรวจสอบความเหมือนกันของ พระรอบองค์พระปฐมเจดีย์

Hardware

Asus TUF F17

CPU: Intel® Core™ i5-11400H Processor

GPU: NVIDIA® GeForce RTX™ 3050

RAM: 16 GB DDR4-3200 SO-DIMM

Storage: SSD 512GB

Software

1. IDLE (Python 3.11 64-bit)
2. Google
3. Visual studio code

ประโยชน์ที่คาดว่าจะได้รับ

1. การอนุรักษ์ศิลป์และวัฒนธรรมไทยเพื่อให้เห็นถึงความสำคัญของการอนุรักษ์ศิลป์และการพิทักษ์สิ่งแวดล้อมธรรมชาติซึ่งจะสามารถอาศัยอยู่ได้ในหมู่ชนต่อไป
2. ได้รับการทำงานเป็นทีม
3. เป็นการเผยแพร่ความรู้ต่อไปการนำความรู้ไปสร้างสรรค์ผลงานศิลปะและนำไปใช้ประโยชน์ต่อสังคมได้ต่อไป

การแบ่งงานของสมาชิกในกลุ่ม

- 640710806 ธนบูรณ์ ตันติภิรมย์

  หน้าที่ ทำ powerpoint

- 640710819 ภัทรพล อุ่นกำเนิด

  หน้าที่ code

- 640710820 ราม ชวาลธวัช

  หน้าที่ code,word

- 640710830 สุธี ชื่นชม

  หน้าที่ word

- 640710917 ธีรภาพ นาสุข

  หน้าที่ powerpoint

ขั้นตอนการทำงานของโปรแกรม

คำสั่งเรียกใช้ไลบรารี่

```
import tkinter as tk
from tkinter import *
from tkinter.ttk import *
from tkinter import filedialog, ttk
import tkinter.font as font
from PIL import Image,ImageTk
import os
import cv2
import json
from ttkthemes import ThemedTk
```

ฟังก์ชัน `select_image()` ทำงานดังนี้:

1. เปลี่ยนข้อความของปุ่ม `button2` เป็น "Check" และปุ่ม `button1` เป็น "Select A Image"

2. ให้ผู้ใช้เลือกไฟล์ผ่านหน้าต่าง `filedialog.askopenfilename()` และเก็บที่อยู่ไฟล์ที่เลือกไว้ใน `file_path`

3. เปิดและปรับขนาดรูปภาพที่อยู่ใน `file_path` ให้มีขนาด 430x450 พิกเซล

4. สร้าง `PhotoImage` จากรูปภาพที่เปลี่ยนขนาดแล้ว

5. แสดงรูปภาพใน `img1_label` และแสดงที่อยู่ไฟล์ใน `path_label`

6. หากเกิดข้อผิดพลาดในการทำงาน จะแสดงข้อความ "Cannot read image" ในปุ่ม `button1`

```
def select_image():
    global file_path, img1
    button2.config(text = 'Check')
    button1.config(text = "Select A Image")
    try:
        file_path = filedialog.askopenfilename(title='Select A File')
        image = Image.open(file_path)

        img1 = cv2.imread(file_path)
        w, h, c = img1.shape
        if w > h:
            image = image.rotate(-90)
        image = image.resize((430, 450))

        photo = ImageTk.PhotoImage(image)
        img1_label.config(image=photo)
        img1_label.image = photo
        path_label.config(text='Directory : ' + file_path, font = ('Verdana', 9))
    except Exception as err:
        print(err)
        button1.config(text = "Cannot read image")
```

ฟังก์ชัน `Process(selected_algorithm)` ทำงานดังนี้:

1. เปลี่ยนข้อความของปุ่ม `button2` เป็น "Check"

2. ปรับขนาดภาพ `img1` ลงเป็น 40% ของขนาดเดิม

3. ใช้อัลกอริทึมที่เลือกไว้ใน `selected_algorithm` เพื่อค้นหาคุณลักษณะของ `img1` และเก็บ keypoints และ descriptors ในตัวแปรที่เกี่ยวข้อง

4. วนลูปผ่านภาพหลักทั้งหมดที่อยู่ใน `main_file_list` และค้นหาคุณลักษณะของแต่ละภาพ พร้อมทั้งนับจำนวนคู่คุณลักษณะที่ตรงกันและความคล้ายคลึงระหว่างภาพ

5. คำนวณความคล้ายคลึงระหว่างภาพและเก็บข้อมูลเกี่ยวกับภาพที่ตรงกันมากที่สุด

6. แสดงผลลัพธ์การค้นหาและภาพผลลัพธ์ใน `img1_label`

7. บันทึกภาพผลลัพธ์ในไฟล์ `result_path`

```
def Process(selected_algorithm):
    global file_path, result_path, img1, main_image_name, image_similarity, image_good_matches, best_match
    button2.config(text = 'Check')
    try:
        # resize image
        scale_percent = 40 # percent of original size
        width = int(img1.shape[1] * scale_percent / 100)
        height = int(img1.shape[0] * scale_percent / 100)
        dim = (width, height)
        img1 = cv2.resize(img1, dim, interpolation = cv2.INTER_AREA)

        # Find Image's key and descriptor
        algos = {'ORB                              *' : cv2.ORB_create(),
                 'KAZE' : cv2.KAZE_create(),
                 'SIFT' : cv2.SIFT_create(),
                 'AKAZE' : cv2.AKAZE_create()}

        algorithm = algos[selected_algorithm]
        print(f'Algorithm : {selected_algorithm}')
        kp1, des1 = algorithm.detectAndCompute(img1, None)
```

```
        # Data Storages
        main_image_name = []
        image_matches = []
        image_good_matches = []
        kps2 = []
        desor2 = []
        good_match_list = []
        image_similarity = []
        ## Read Main Image
        print('\n')
        print(' --------------- Processing --------------- ')
        for main in main_file_list:
            print(f'Reading {main}')
            img2 = cv2.imread(path + f'\{main}', 0)
            kp2, des2 = algorithm.detectAndCompute(img2, None)
            matcher = cv2.BFMatcher()
            matches = matcher.knnMatch(des1, des2, k=2)
            threshold = 0.75
            good_matches = []
            for m, n in matches:
                if m.distance < threshold*n.distance:
                    good_matches.append([m])
            print(f'Good Matches : {len(good_matches)}')

            # Store all Output into Data storages
            try:
                similarity = (len(good_matches) / len(matches)) * 100
                similarity = round(similarity, 2)
                image_similarity.append(similarity)
                print(f'Match Percentages : {similarity:.2f}%')
            except:
                image_similarity.append('0%')
            image_matches.append(len(matches))
            kps2.append(kp2)
            desor2.append(des2)
            main_image_name.append(main)
            image_good_matches.append(len(good_matches))
            good_match_list.append(good_matches)

        best_match = image_good_matches.index(max(image_good_matches))
        print('\n')
        print(' --------------- Result --------------- ')
        print(f'Best Match Image : {main_image_name[best_match]} \nMatch Percentage
              : {image_similarity[best_match]}%\nGood Matches : {image_good_matches[best_match]}')

        # Read the best match to show the result

        show_result()
        best_match_image = cv2.imread(path + f'\{main_image_name[best_match]}', 0)
        matching_result = cv2.drawMatchesKnn(img1, kp1, best_match_image, kps2[best_match]
                                             , good_match_list[best_match], None, flags = 2)

        result_path = 'Matching_Result.jpg'
        cv2.imwrite(result_path, matching_result)
        image = Image.open(result_path)
        image = image.resize((700, 450))
        photo = ImageTk.PhotoImage(image)
        img1_label.config(image=photo)
        img1_label.image = photo

    except Exception as err:
        print(err)
        button2.config(text = 'Image not found')
```

ฟังก์ชัน `show_result()` ทำงานดังนี้:

1. อ่านไฟล์ข้อความและโหลดข้อมูลเป็น JSON

2. ค้นหาและเก็บชื่อภาพและชื่อภาษาไทยที่สอดคล้องกัน

3. แสดงผลลัพธ์การค้นหาภาพที่ตรงกันมากที่สุดใน `result`

4. กำหนดข้อความใน `label5` เป็น "Matcing Result"

5. สร้างเอฟเฟกต์อนิเมชั่นให้กับกล่องข้อความ `textbox1`, `textbox2`, และ `textbox3`
เพื่อทำให้กล่องข้อความเคลื่อนที่แนวตั้งจนกว่าขนาดของกล่องข้อความจะถึง 160

6. แสดงเอฟเฟกต์อนิเมชั่นให้กับข้อความ `label1`, `label2`, และ `label3` เพื่อทำให้ข้อความปรากฏทีละตัวอักษรทุก 500 มิลลิวินาที

7. กำหนดค่าผลลัพธ์และข้อมูลที่เกี่ยวข้องในตัวแปร `entryText1`, `entryText2`, และ `entryText3`

```
def show_result():
    global size1, size2, size3, word1, word2, word3, w1, w2, w3, count1, count2, count3, scriptpath

        # Reading text file
    with open(scriptpath + r'\buddha_names.txt', 'r', encoding='utf-8') as names:
        data = json.load(names)

    picture_name = []
    thai_name = []

    for i in data['buddha']:
        picture_name.append(i['picture_name'])
        thai_name.append(i['thai_name'])

    result = thai_name[picture_name.index(main_image_name[best_match])]

    label5.config(text = 'Matcing Result')

    #Animation Box
    if size1 < 160:
        size1 += 20
        textbox1.place(x = 750, y = 210, width = size1, height = 30)
        window.after(1, show_result)
    elif size2 < 160:
        size2 += 20
        textbox2.place(x = 750, y = 310, width = size2, height = 30)
        window.after(1, show_result)
    elif size3 < 160:
        size3 += 20
        textbox3.place(x = 750, y = 410, width = size3, height = 30)
        window.after(1, show_result)
```

```
    #Animation Text
    if  size1 == 160 and len(word1) != len(w1):
        w1.append(word1[count1])
        label1.config(text = w1)
        count1 += 1
        window.after(500, show_result)

    elif  size2 == 160 and len(word2) != len(w2):
        w2.append(word2[count2])
        label2.config(text = w2)
        count2 += 1
        window.after(500, show_result)

    elif  size3 == 160 and len(word3) != len(w3):
        w3.append(word3[count3])
        label3.config(text = w3)
        count3 += 1
        window.after(500, show_result)

    entryText1.set(result)
    entryText2.set(image_similarity[best_match])
    entryText3.set(image_good_matches[best_match])
```

โค้ดดังกล่าวเป็นส่วนเริ่มต้นของโปรแกรม ซึ่งมีหน้าที่ดังนี้:

1. กำหนดตัวแปร `scriptpath` ให้เป็นที่อยู่ปัจจุบันของสคริปต์

2. กำหนดตัวแปร `picpath` เป็นเส้นทางสำหรับเข้าถึงโฟลเดอร์ภาพ (ImgGreen) โดยใช้ `r'\ImgGreen'` เพื่อระบุว่าเป็นเส้นทางแบบ raw string

3. รวมที่อยู่ปัจจุบันและเส้นทางของโฟลเดอร์ภาพเข้าด้วยกันเพื่อสร้างตัวแปร `path`

4. สร้างรายการ `main_file_list` เพื่อเก็บชื่อไฟล์ภาพหลักที่อยู่ในโฟลเดอร์ `path` โดยตรวจสอบนามสกุลไฟล์ว่าเป็น `.jpg`, `.jpeg`, หรือ `.png` เท่านั้น

```
#Starting program
#Uploading green_screen images (main_image)
scriptpath = os.getcwd() + r'\data'
picpath = r'\ImgGreen'
path = scriptpath + picpath
main_file_list = []
for main_img in os.listdir(path):
    if main_img.endswith('.jpg') or main_img.endswith('.jpeg') or main_img.endswith('.png'):
        main_file_list.append(main_img)
```

- กำหนดค่าเริ่มต้นให้กับตัวแปร `size1`, `size2`, `size3`, `word1`, `word2`, `word3`, `w1`, `w2`, `w3`, `count1`, `count2`, และ `count3` ในรูปแบบที่เหมาะสำหรับการใช้งานในภายหลัง

```
size1 = 0
size2 = 0
size3 = 0
word1 = list('Thai_name')
word2 = list('Similarity')
word3 = list('Similar_points')
w1 = []
w2 = []
w3 = []
count1 = 0
count2 = 0
count3 = 0
```

โค้ดดังกล่าวเป็นส่วนของ GUI ที่สร้างหน้าต่างและองค์ประกอบต่างๆ โดยประกอบไปด้วย:

- การสร้างหน้าต่าง GUI และกำหนดคุณสมบัติต่างๆ เช่น ธีม, ไอคอน, ชื่อหน้าต่าง, ขนาด, และความสามารถในการปรับขนาด

```
# GUI
window = ThemedTk(theme = "adapta")
window.iconbitmap(scriptpath + r'\icon.ico')
window.title('pip install')
window.geometry('1000x600')
window.resizable(False, False)
```

- การสร้างปุ่ม "Select A Image" เพื่อเลือกไฟล์ภาพ

```
#Seleting button
button1 = ttk.Button(window, text="Select A Image", command=select_image)
button1.place(width=200, height=60, x=50,y=40)
```

- การสร้าง combobox เพื่อให้ผู้ใช้เลือกอัลกอริทึม

```
#Combo box
combox = ttk.Combobox(window, values = ['ORB         *', 'KAZE', 'SIFT', 'AKAZE'])
combox.current(0)
combox.place(width= 125, height = 25, x = 825, y = 10)
```

- การสร้างปุ่ม "Check" เพื่อเริ่มกระบวนการ

```
#Process button
button2 = ttk.Button(window, text="Check", command=lambda : Process(combox.get()))
button2.place(width = 200, height = 60, x = 750, y = 40)
```

- สร้างป้ายภาพ img1_label เพื่อแสดงภาพ

```
#Image label
img1_label = ttk.Label(window, text = '')
img1_label.place(x=30,y=115)
```

- สร้างป้ายแสดงเส้นทาง path_label

```
#Path label
path_label = ttk.Label(window, text = '')
path_label.place(x = 30, y = 570)
```

- สร้างป้าย label1 เพื่อแสดงข้อความ "Thai_name"

```
#'Thai_name' label
label1 = tk.Label(window, text = '', font = ('Verdana', 10))
label1.place(x = 755, y = 190)
```

- สร้างกล่องข้อความ textbox1 เพื่อให้ผู้ใช้ป้อนข้อมูลในรูปแบบข้อความ

```
#Thai_name box
entryText1 = tk.StringVar()
textbox1 = ttk.Entry(window, textvariable = entryText1, font = ('Verdana', 9))
textbox1.place(x = 750, y = 210, width = size1, height = 35)
```

- สร้างป้าย label2 เพื่อแสดงข้อความ "Similarity"

```
#'Similarity' label
label2 = tk.Label(window, text = '', font = ('Verdana', 10))
label2.place(x = 755, y = 290)
```

- สร้างกล่องข้อความ textbox2 เพื่อแสดงข้อมูลที่ผู้ใช้ป้อนเกี่ยวกับ "Similarity"

```
#Similarity box
entryText2 = tk.StringVar()
textbox2 = ttk.Entry(window, textvariable=entryText2, font = ('Verdana', 9))
textbox2.place(x = 750, y = 310, width = size2, height = 35)
```

- สร้างป้าย label3 เพื่อแสดงข้อความ "Match Points"

```
#'Match Points' label
label3 = tk.Label(window, text = '', font = ('Verdana', 10))
label3.place(x = 755, y = 390)
```

- สร้างกล่องข้อความ textbox3 เพื่อแสดงข้อมูลที่ผู้ใช้ป้อนเกี่ยวกับ "Match Points"

```
#Match Points box
entryText3 = tk.StringVar()
textbox3 = ttk.Entry(window, textvariable=entryText3, font = ('Verdana', 9))
textbox3.place(x = 750, y = 410, width = size3, height = 35)
```

- สร้างป้าย label4 เพื่อแสดงข้อความ "Algorithm :"

```
#Combo label
label4 = tk.Label(window, text = 'Algorithm : ', font = ('Verdana', 9))
label4.place(x = 750, y = 12)
```

- สร้างป้าย label5 เพื่อแสดงข้อความ "Matching Result" โดยเน้นคำว่า "Matching Result" ด้วยเส้นใต้

```
#'Matching Result' label
label5 = tk.Label(window, text = '', font = ('Verdana', 15, 'underline'))
label5.place(x = 755, y = 150)
```

- รันหน้าต่าง GUI

```
window.mainloop()
```

- ลบไฟล์ผลลัพธ์

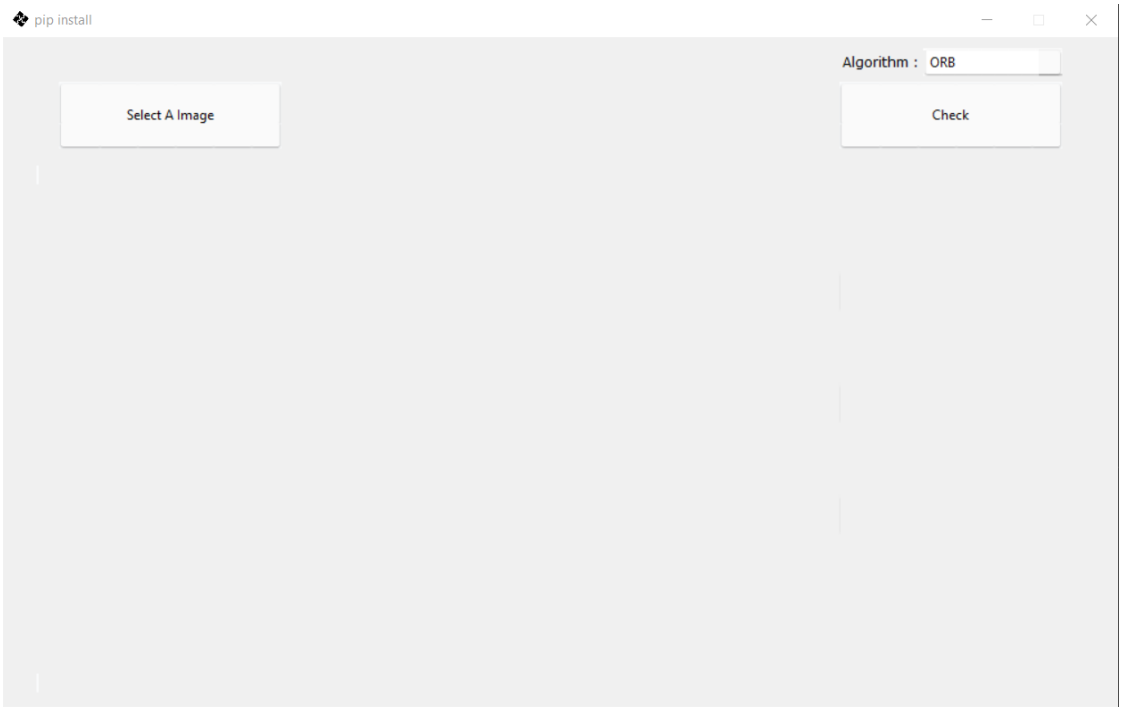
```
try:
    os.remove(result_path)
```

- จัดการข้อผิดพลาดในกรณีที่ไม่สามารถลบไฟล์ได้

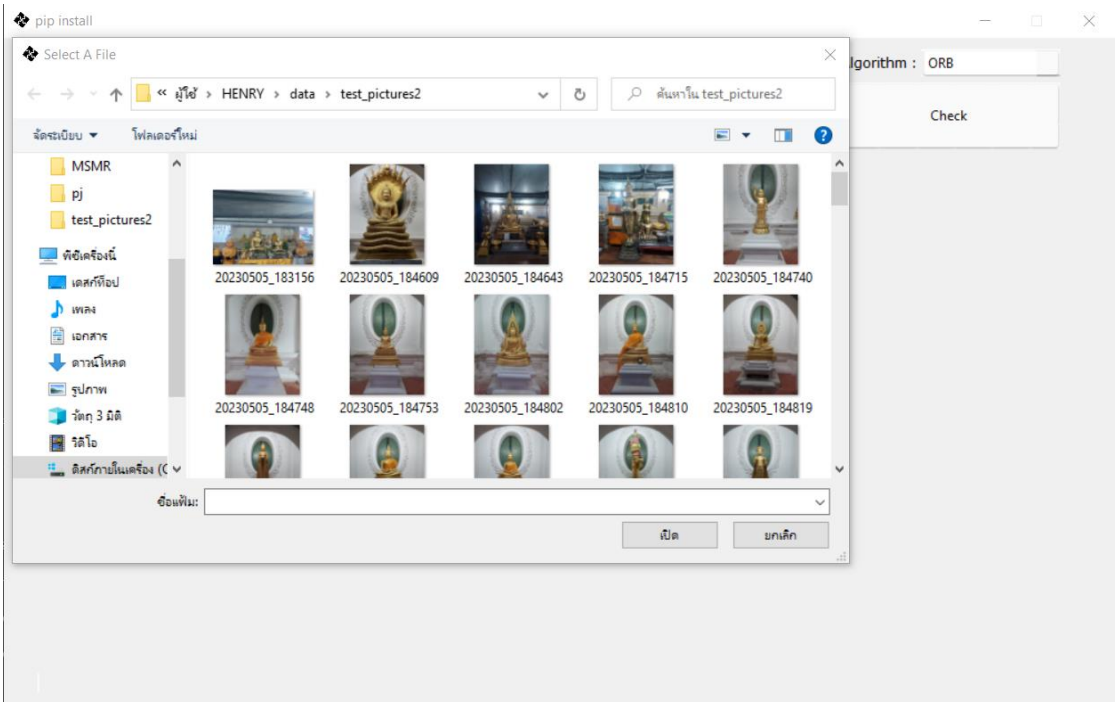
```
except:
    pass
```

ผลการดำเนินงาน

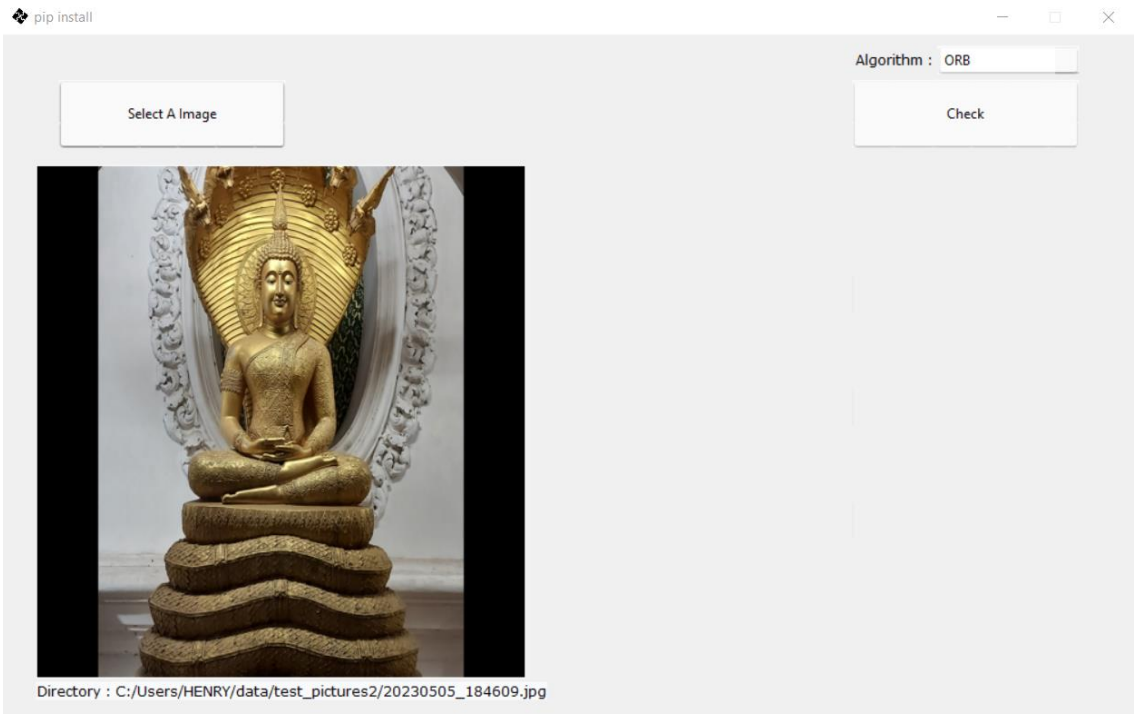
1.หน้าต่าง GUI



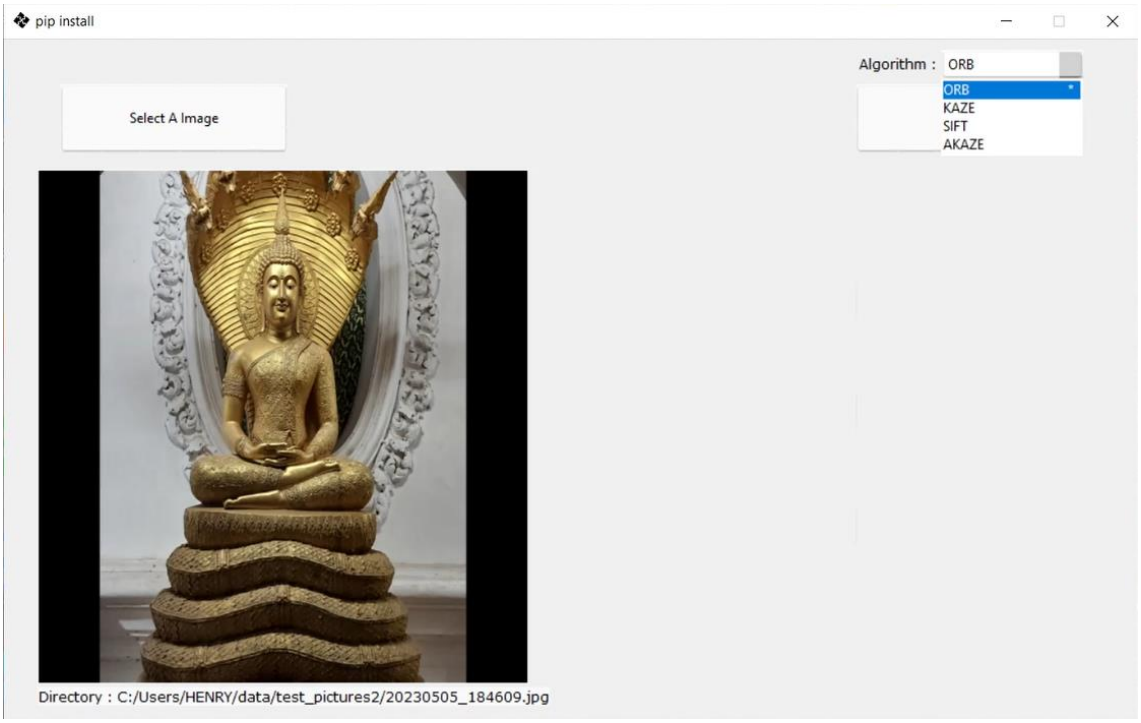
2.กดปุ่ม Select A Image แล้วเลือกรูปภาพที่ต้องการ



3.แสดงภาพที่เลือกที่หน้า GUI



4.เลือก Algorithm ที่จะใช้ในการวิเคราะห์ภาพ



5.กดปุ่ม Check โปรแกรมจะทำการเปรียบเทียบภาพในฐานข้อมูลแล้วแสดงชื่อพระ และค่า Similarity กับ Similar_points