**浙江工商大学计算机科学与技术学院**

**《视觉计算方向专业实践》报告**

**专 业： 计算机科学与技术**

**班 级： 计科2203班**

**学 号： 2212190505**

**姓 名： 王倩**

**2025 年 3月9日**

1. 实验内容

用给定的图像完成SIFT,ORB或其他基于关键点的特征提取，完成图像的匹配，并比较两种算法的时间性能。设计GUI界面，在界面上可以选择待匹配图像，在给定路径下的图像中找出最匹配的图像并显示在界面上，界面上还可以展示匹配时间等实验结果。

二、代码

match.py

import cv2

import numpy as np

import time

import os

def load\_images\_from\_folder(folder):

""" 从文件夹加载所有图像 """

images = {}

for filename in os.listdir(folder):

img\_path = os.path.join(folder, filename)

img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)

if img is not None:

images[filename] = img

return images

def match\_images(query\_img\_path, dataset\_folder, method='SIFT'):

""" 进行图像匹配 """

if method not in ['SIFT', 'ORB']:

raise ValueError("Method must be 'SIFT' or 'ORB'")

query\_img = cv2.imread(query\_img\_path, cv2.IMREAD\_GRAYSCALE)

if query\_img is None:

raise ValueError(f"Cannot load image: {query\_img\_path}")

dataset\_images = load\_images\_from\_folder(dataset\_folder)

if not dataset\_images:

raise ValueError("No images found in dataset folder!")

best\_match = None

best\_score = float('-inf')

best\_match\_img = None

best\_match\_result = None

match\_time = None

# 选择特征检测器

detector = cv2.SIFT\_create() if method == 'SIFT' else cv2.ORB\_create()

# 计算查询图像特征

kp1, des1 = detector.detectAndCompute(query\_img, None)

if des1 is None or len(kp1) == 0:

raise ValueError("No keypoints found in query image")

# 选择匹配器

matcher = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=True) if method == 'SIFT' else cv2.BFMatcher(cv2.NORM\_HAMMING, crossCheck=True)

start\_time = time.time()

for filename, img in dataset\_images.items():

kp2, des2 = detector.detectAndCompute(img, None)

if des2 is None or len(kp2) == 0:

continue # 跳过无关键点的图片

matches = matcher.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

# 避免除 0 错误

valid\_matches = [m for m in matches if m.distance > 0]

score = sum([1 / (m.distance + 1e-6) for m in valid\_matches[:50]]) if valid\_matches else 0 # 取前50个匹配点

if score > best\_score:

best\_score = score

best\_match = filename

best\_match\_img = img

best\_match\_result = cv2.drawMatches(query\_img, kp1, img, kp2, valid\_matches[:50], None, flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

end\_time = time.time()

match\_time = end\_time - start\_time

if best\_match\_result is None:

raise ValueError("No valid matches found in dataset")

return best\_match, best\_match\_result, match\_time

2、Test.py

import sys  
import cv2  
import os  
import traceback  
from PyQt5.QtWidgets import QApplication, QLabel, QPushButton, QVBoxLayout, QHBoxLayout, QWidget, QFileDialog, QComboBox, QMessageBox  
from PyQt5.QtGui import QPixmap, QImage  
from match import match\_images  
  
class ImageMatcherApp(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.setWindowTitle("图像匹配系统")  
 self.setGeometry(100, 100, 900, 600)  
  
 # UI 组件  
 self.select\_btn = QPushButton("选择查询图像")  
 self.select\_btn.clicked.connect(self.load\_query\_image)  
  
 self.method\_selector = QComboBox()  
 self.method\_selector.addItems(["SIFT", "ORB"])  
  
 self.match\_btn = QPushButton("开始匹配")  
 self.match\_btn.clicked.connect(self.run\_match)  
  
 self.query\_label = QLabel("查询图像")  
 self.query\_label.setFixedSize(400, 300)  
  
 self.result\_label = QLabel("最佳匹配图像")  
 self.result\_label.setFixedSize(400, 300)  
  
 self.time\_label = QLabel("匹配时间: --")  
  
 # 布局  
 layout = QVBoxLayout()  
 top\_layout = QHBoxLayout()  
 bottom\_layout = QHBoxLayout()  
  
 top\_layout.addWidget(self.select\_btn)  
 top\_layout.addWidget(self.method\_selector)  
 top\_layout.addWidget(self.match\_btn)  
  
 bottom\_layout.addWidget(self.query\_label)  
 bottom\_layout.addWidget(self.result\_label)  
  
 layout.addLayout(top\_layout)  
 layout.addLayout(bottom\_layout)  
 layout.addWidget(self.time\_label)  
  
 self.setLayout(layout)  
  
 self.query\_img\_path = None  
 self.dataset\_folder = r"D:\Pycharm\Projects\PythonProject\A0C632"  
  
 def load\_query\_image(self):  
 file\_path, \_ = QFileDialog.getOpenFileName(self, "选择查询图像", "", "Images (\*.png \*.jpg \*.jpeg)")  
 if file\_path:  
 self.query\_img\_path = file\_path  
 pixmap = QPixmap(file\_path)  
 self.query\_label.setPixmap(pixmap.scaled(self.query\_label.width(), self.query\_label.height()))  
  
 def run\_match(self):  
 if not self.query\_img\_path or not os.path.exists(self.dataset\_folder):  
 QMessageBox.critical(self, "错误", "请选择查询图像，并确保数据集路径存在！")  
 return  
  
 method = self.method\_selector.currentText()  
 try:  
 best\_match, match\_img, match\_time = match\_images(self.query\_img\_path, self.dataset\_folder, method)  
 self.time\_label.setText(f"最佳匹配: {best\_match}, 耗时: {match\_time:.4f} 秒")  
  
 height, width, channel = match\_img.shape  
 bytes\_per\_line = 3 \* width  
 q\_img = QImage(match\_img.data, width, height, bytes\_per\_line, QImage.Format\_RGB888)  
 self.result\_label.setPixmap(QPixmap.fromImage(q\_img).scaled(self.result\_label.width(), self.result\_label.height()))  
  
 except Exception as e:  
 traceback.print\_exc() # 打印详细错误信息  
 QMessageBox.critical(self, "匹配失败", str(e))  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = ImageMatcherApp()  
 window.show()  
 sys.exit(app.exec\_())

三、运行结果



