**《Python高级应用》**

**大作业实验报告**

项目名称 ： 多维人脸分析系统

姓 名 ： 　　 高心阳

学 号 ： 　 　 084623237

班 级 ： 　 计算机232

成 绩 ：

**人工智能与信息技术学院**

**南京中医药大学**

|  |
| --- |
| **项目要求：** |
| **项目描述**  开发一个基于计算机视觉的多维人脸分析系统，通过摄像头或图片检测人脸，并分析年龄、性别、情绪和种族信息。支持实时监测和图片分析，用户可根据需求自定义检测选项。  **主要功能：**   1. 实时监测人脸并进行分析。 2. 选择本地图片进行人脸检测和分析。 3. 显示检测结果，包括性别、年龄、情绪和种族。 4. 支持用户配置检测选项。 5. 支持图片查看功能，并可调整图片查看尺寸。 |
| **采用的技术** |
| **技术栈**  PyQt5：用于图形用户界面设计。提供窗口布局、按钮、表格、进度条等组件。  OpenCV：处理和显示图像及视频流。绘制检测框。  DeepFace：进行人脸分析，支持年龄、性别、情绪和种族检测。  **库功能简介**  PyQt5：提供丰富的GUI组件用于界面构建。  OpenCV：图像处理库，用于摄像头画面捕获和图像格式转换。  DeepFace：一个高效的人脸分析框架，提供人脸检测和多维分析功能。  QSettings：保存和加载用户配置，例如图片查看尺寸和检测选项。 |
| **关键代码及相应运行结果** |
| |  | | --- | | 界面初始化 | | def init\_ui(self):  self.setWindowTitle("多维人脸分析系统")  self.setFixedSize(1000, 800)  # 主窗口  self.central\_widget = QWidget(self)  self.setCentralWidget(self.central\_widget)  self.main\_layout = QVBoxLayout(self.central\_widget)  # 图像显示区域  self.image\_label = QLabel(self)  self.image\_label.setAlignment(Qt.AlignCenter)  self.image\_label.setMinimumHeight(500)  self.image\_label.mousePressEvent = self.show\_large\_image  self.main\_layout.addWidget(self.image\_label, 3)  # 进度条  self.progress\_bar = QProgressBar(self)  self.progress\_bar.setVisible(False)  self.main\_layout.addWidget(self.progress\_bar)  # 结果  self.result\_table = QTableWidget(self)  self.result\_table.setColumnCount(5)  self.result\_table.setHorizontalHeaderLabels(['序号', '性别', '年龄', '情绪', '种族'])  self.result\_table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)  self.result\_table.setMinimumHeight(200)  self.main\_layout.addWidget(self.result\_table)  # 按钮  self.button\_layout = QHBoxLayout()  # 实时监测  self.start\_button = QPushButton("启动实时检测", self)  self.start\_button.clicked.connect(self.toggle\_realtime)  self.button\_layout.addWidget(self.start\_button)  # 图片检测  self.load\_button = QPushButton("选择图片", self)  self.load\_button.clicked.connect(self.load\_image)  self.button\_layout.addWidget(self.load\_button)  # 检测选项配置  self.config\_button = QPushButton("检测选项", self)  self.config\_button.clicked.connect(self.show\_detection\_options)  self.button\_layout.addWidget(self.config\_button)  self.main\_layout.addLayout(self.button\_layout) | |  |  |  | | --- | | 实时摄像头监测 | | def start\_realtime(self):  try:  if self.cap is None:  self.cap = cv2.VideoCapture(0)  if not self.cap.isOpened():  raise IOError("无法打开摄像头")  self.timer.timeout.connect(self.update\_frame)  self.timer.start(30)  self.realtime\_running = True  self.start\_button.setText("停止实时检测")  except Exception as e:  QMessageBox.warning(self, "错误", f"启动实时监测失败: {str(e)}")  self.stop\_realtime() |  |  | | --- | | 图片加载与分析 | | def load\_image(self):  try:  file\_path, \_ = QFileDialog.getOpenFileName(self, "选择图片", "", "Image Files (\*.jpg \*.jpeg \*.png)")  if not file\_path:  return  # 显示进度条  self.progress\_bar.setVisible(True)  self.progress\_bar.setValue(0)  # 处理图片  def process\_image():  try:  # 读取图片  self.progress\_signal.progress.emit(20)  frame = cv2.imread(file\_path)  frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  # 获取启用的检测选项  actions = [self.option\_mapping[option] for option, enabled in  self.detection\_options.items() if enabled]  if not actions:  self.progress\_signal.error.emit("请至少选择一种检测选项")  return  self.progress\_signal.progress.emit(40)  # 分析图片  analysis = DeepFace.analyze(  img\_path=file\_path,  actions=actions,  enforce\_detection=False  )  self.progress\_signal.progress.emit(80)  # 更新UI  self.process\_face\_analysis(frame, analysis)  self.progress\_signal.progress.emit(100)  self.progress\_signal.finished.emit()  except Exception as e:  self.progress\_signal.error.emit(str(e))  # 启动处理线程  threading.Thread(target=process\_image, daemon=True).start()  except Exception as e:  self.progress\_signal.error.emit(str(e)) | |  |  |  | | --- | | 检测选项配置 | | def show\_detection\_options(self):  dialog = QDialog(self)  layout = QVBoxLayout()  checkboxes = {}  for option, default in self.detection\_options.items():  checkbox = QCheckBox(option)  checkbox.setChecked(default)  checkboxes[option] = checkbox  layout.addWidget(checkbox)  save\_button = QPushButton("保存设置")  save\_button.clicked.connect(lambda: self.save\_options(dialog, checkboxes))  layout.addWidget(save\_button)  dialog.setLayout(layout)  dialog.exec\_() | |  | |