第二章 用户界面设计

用户界面是软件人机交互、操作逻辑的重要部分，可以让软件的操作变得舒适、简单。

2.1 用户界面需求

用户界面的需求如下：

1）页面内容：界面和按钮设置布局合理、简洁，功能便于查找和使用。

2）导航结构：页面具有明确的导航指示，且便于理解，方便用户使用。

3）艺术风格：界面、版面形象简洁明了、布局合理,字号大小适宜、字体选择合理，前后一致，美观大方；动与静搭配恰当,动静效果好；色彩和谐自然,与主题内容相协调。

2.2用户界面详细设计

详细设计是对软件概要设计的细化，是对概要设计中各模块的具体实现过程的设计，该设计描述了各模块的实现代码文件结构、各函数或方法的调用关系以及具体函数或方法的实现过程，其描述采用伪代码、自然语言、流程图、表格等方式。

本文主要采用类Java伪代码、实际代码和自然语言描述的方法进行详细设计的描述。

2.2.1 界面功能

本软件的图形界面较为简单，主要包含一个软件主界面。软件主界面可以实现选择照片、16位识别、32位识别以及结果展示功能，其界面流程图如图1所示：

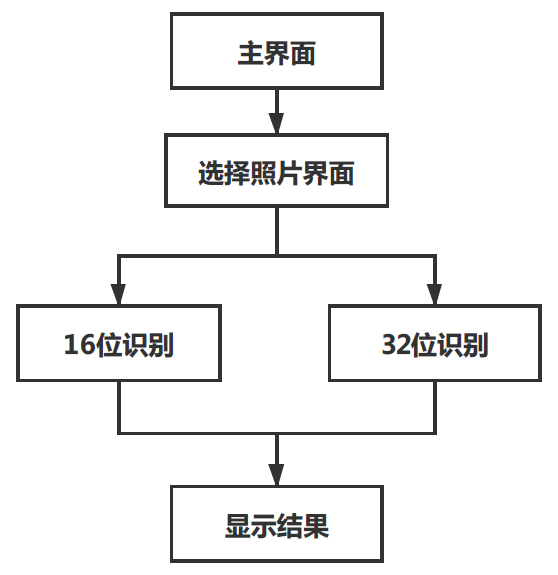


图1 软件主界面流程图

2.2.2 界面组件布局

本软件的图形界面主要由Java图形用户界面工具包Swing编写。在Swing中，每个界面对应一个窗口，每个窗口又由许多部件组成，如，这些部件各自实现了不同的功能，是界面中的重要组成部分。

软件主界面的布局情况如表1所示：

表1 软件主界面布局情况表

|  |  |  |  |
| --- | --- | --- | --- |
| 部件名称 | 部件位置 | 类型 | 功能 |
| topPanel | 页面上方 | JPanel | 展示标题和uestc校徽 |
| uestc | topPanel左方 | JLabel | 展示uestc校徽 |
| title | topPanel中部 | JLabel | 展示标题 |
| centerPanel | 页面中下方 | JPanel | 展示图片和按钮 |
| clPanel | centerPanel左方 | JPanel | 展示按钮和被选择的图片 |
| crPanel | centerPanel右方 | JPanel | 展示九张图片 |
| selectPic | clPanel上方 | JButton | 选择图片 |
| jLabelSelectedPic | clPanel中部 | JLabel | 显示被选择的图片 |
| proRetr32 | clPanel下方 | JButton | 用32位哈希值检索数据库 |
| proRetr16 | clPanel下方 | JButton | 用16位哈希值检索数据库 |
| jLabelNinePic | crPanel | JLabel | 展示9张最相似图片 |

软件主界面的示意图如图2所示：

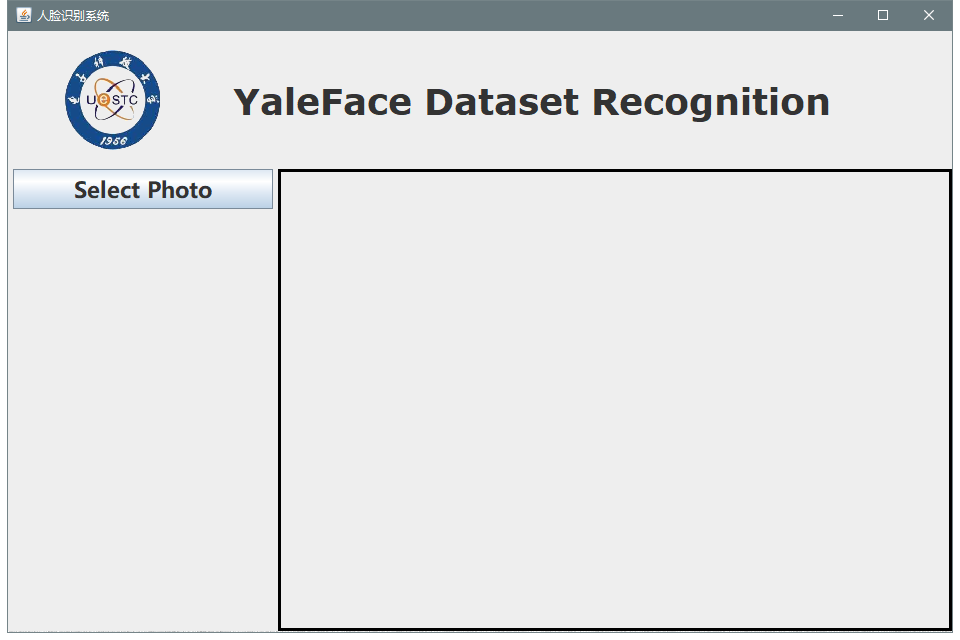


图2 软件主界面示意图

在软件主界面上方通过topPanel展示了校徽和软件标题，点击左侧“Select Photo”按钮后会弹出选择文件的子界面，其界面示意图如图3所示：

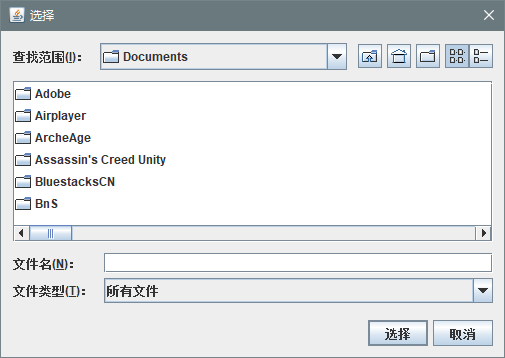


图3 选择文件界面示意图

在该界面，用户可以方便地浏览并选择相应的照片文件，进行后续的人脸检索操作。当用户选定了一张照片后，界面左侧会出现该照片，并在其下方分别出现“Face Recognition(32bit)”和“Face Recognition(16bit)”按钮，其界面示意图如图4所示：

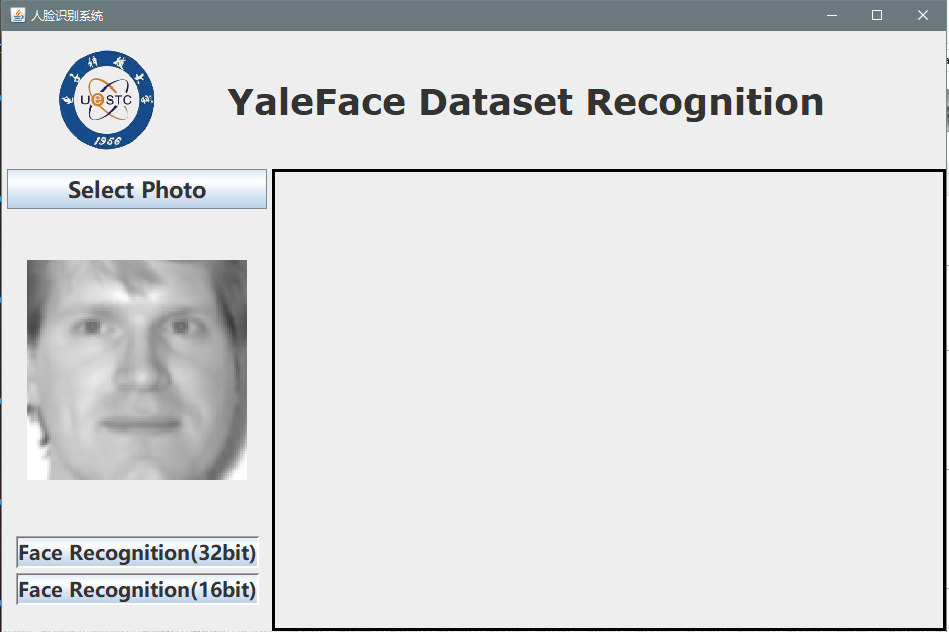


图4 选择照片后的界面示意图

此时，分别点击“Face Recognition(32bit)”和“Face Recognition(16bit)”按钮，即可在界面右侧显示检索到的9张最相似图片。以图4中所选择的的照片为例，32位识别和16位识别后的软件界面示意图如图5和6所示：

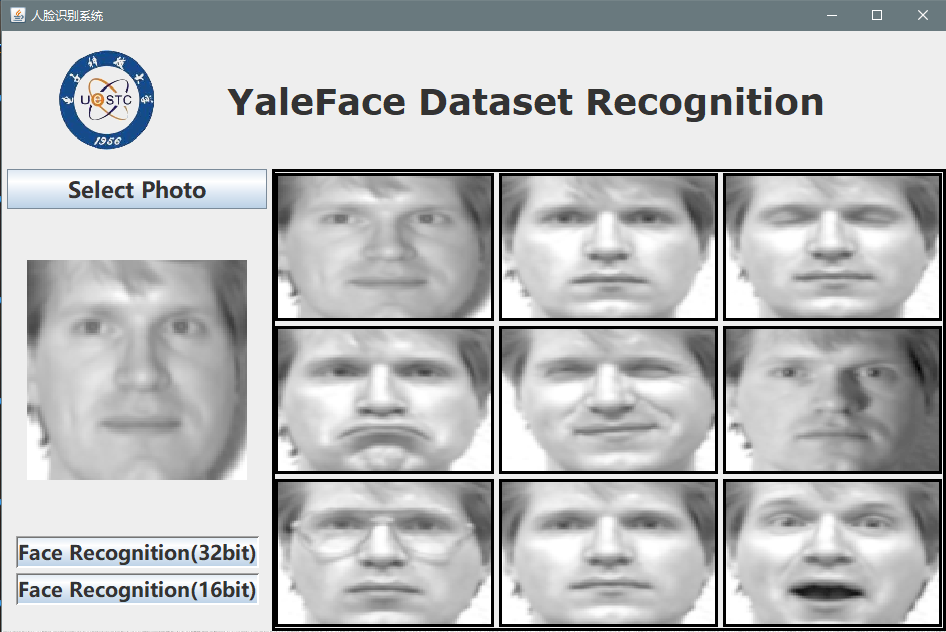


图5 32位识别后的界面示意图

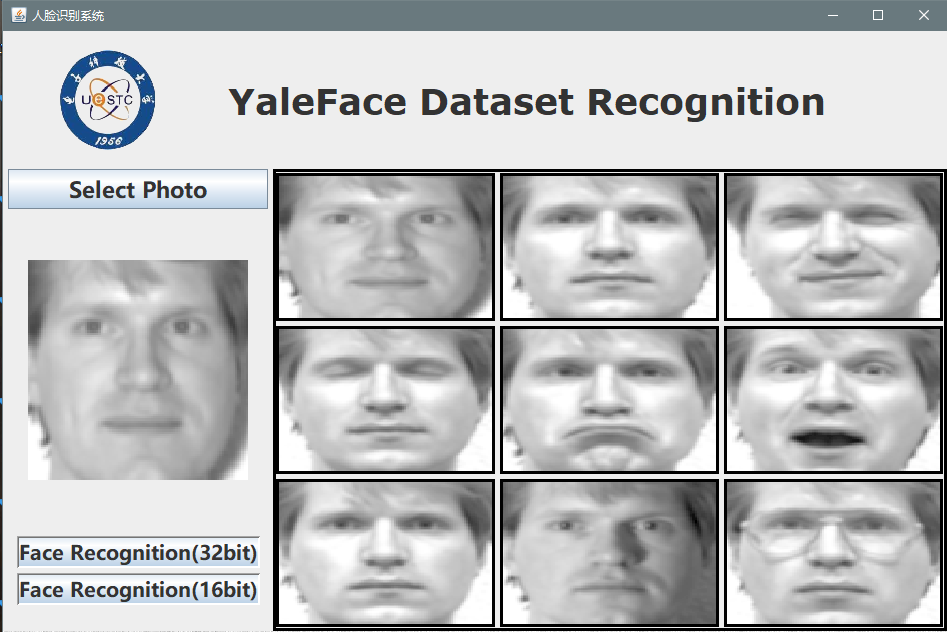


图6 16位识别后的界面示意图

2.2.3 界面设计代码实现

（1）关键函数名

void createTopPanel()

void createCenterPanel()

void createCLPanel()

void createCRPanel()

void showNinePics()

class selectAction implements ActionListener{

public void actionPerformed(ActionEvent e){}

}

class retrieve32Action implements ActionListener{

public void actionPerformed(ActionEvent e) {}

}

class retrieve16Action implements ActionListener{

public void actionPerformed(ActionEvent e) {}

}

其中，createTopPanel()用于创建topPanel，展示uestc校徽和标题“YaleFace Dataset Recognition”， createCenterPanel()用于创建centerPanel，createCLPanel()用于创建clPanel，createCRPanel()用于创建crPanel，showNinePics()用于在crPanel上展示九张图片，selectAction implements ActionListener是selectPic按钮的关联动作：弹出文件选择器，选择要识别的图片。

proRetr32按钮的关联动作为调用retrieveNearestPics32.py文件，得到9张图片的路径和文件名，并将它们展示到界面上。proRetr16按钮的关联动作为调用retrieveNearestPics16.py文件，得到9张图片的路径和文件名，并将它们展示到界面上。

（2）调用数据库的关键代码

本软件主要使用Python对数据库进行调用，以下为调用的数据库部分的关键代码。

1）以下为调用retrieveNearestPics32.py的代码，该python文件根据传入的被选择的图片，查找32位哈希值数据库，输出9张哈希值与被选择的图片最相似的图片。

String[] args=new String[] {"python",dir.getAbsolutePath()+"\\retrieveNearestPics32.py",selectedFilePath,host,user,password};

ninePics=new ArrayList<>();

try {

Process proc=Runtime.getRuntime().exec(args);

BufferedReader in = new BufferedReader(new InputStreamReader(proc.getInputStream()));

String line = null;

while((line=in.readLine())!=null) {

ninePics.add(line);

}

in.close();

proc.waitFor();

}catch (IOException err) {

err.printStackTrace();

} catch (InterruptedException err) {

err.printStackTrace();

}

……

}

}

2）以下为调用retrieveNearestPics16.py的代码，该python文件根据传入的被选择的图片，查找16哈希值数据库，输出9张哈希值与被选择的图片最相似的图片。

String[] args=new String[] {"python",dir.getAbsoluteFile()+"\\retrieveNearestPics16.py",selectedFilePath,host,user,password};

ninePics=new ArrayList();

try {

Process proc=Runtime.getRuntime().exec(args);

BufferedReader in = new BufferedReader(new InputStreamReader(proc.getInputStream()));

String line = null;

while((line=in.readLine())!=null) {

ninePics.add(line);

}

in.close();

proc.waitFor();

}catch(IOException err) {

err.printStackTrace();

}catch(InterruptedException err) {

err.printStackTrace();

}

……

}

}

第三章 数据库设计

数据库是计算机中存储数据的地方，本软件需要对文件的哈希码进行存储，故需要进行数据库的设计。

3.1 数据库需求

数据库的需求如下：

1）安全性：要求数据库具有一定安全性，未经授权的用户无法访问数据。

2）稳定性：要求数据库中的数据保持一定的稳定性，不易损坏或丢失。

3）方便性：要求数据库简便易用。

MySQL是目前比较流行的关系型数据库管理系统之一，由于其开源、简便等特性，本软件使用MySQL作为数据库管理系统。

3.2数据库详细设计

详细设计是对软件概要设计的细化，是对概要设计中各模块的具体实现过程的设计，该设计描述了各模块的实现代码文件结构、各函数或方法的调用关系以及具体函数或方法的实现过程，其描述采用伪代码、自然语言、流程图、表格等方式。

本文主要采用类Java伪代码、实际代码和自然语言描述的方法进行详细设计的描述。

3.2.1 数据库逻辑结构模型设计

根据本系统的需求分析，本软件数据库的逻辑设计如下：

（1）数据库表

* test\_dataset16

该表用于存储测试集数据的文件名和该文件对应的16位哈希码。

* test\_dataset32

该表用于存储测试集数据的文件名和该文件对应的32位哈希码。

* train\_dataset16

该表用于存储训练集数据的文件名和该文件对应的16位哈希码。

* train\_dataset32

该表用于存储训练集数据的文件名和该文件对应的32位哈希码。

（2）数据字典

1)test\_dataset16的数据字典如表2所示：

表2 test\_dataset16表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段代码 | 字段类型 | 关键字 | 可否为空 | 外键参照 |
| 文件名 | FILE\_NAME | CHAR(20) | Y | N | N |
| 哈希值 | HASH\_CODE | CHAR(16) | N | N | N |

2)test\_dataset32的数据字典如表3所示：

表3 test\_dataset32表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段代码 | 字段类型 | 关键字 | 可否为空 | 外键参照 |
| 文件名 | FILE\_NAME | CHAR(20) | Y | N | N |
| 哈希值 | HASH\_CODE | CHAR(32) | N | N | N |

3)train\_dataset16的数据字典如表4所示：

表4 train\_dataset16表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段代码 | 字段类型 | 关键字 | 可否为空 | 外键参照 |
| 文件名 | FILE\_NAME | CHAR(20) | Y | N | N |
| 哈希值 | HASH\_CODE | CHAR(16) | N | N | N |

4)train\_dataset32的数据字典如表5所示：

表5 train\_dataset32表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段代码 | 字段类型 | 关键字 | 可否为空 | 外键参照 |
| 文件名 | FILE\_NAME | CHAR(20) | Y | N | N |
| 哈希值 | HASH\_CODE | CHAR(32) | N | N | N |

3.2.2 数据库与程序对接

（1）主要程序名

本程序使用Python语言进行SQL操作，数据库为MySQL，使用Python库pymysql，分为三个.py文件，分别为：

1）loadHashingCodes.py 创建数据库yalefaces\_database，将训练集和测试集的32/16位哈希值信息写入数据库，分别形成test\_dataset16、test\_dataset32、train\_dataset16、train\_dataset32四个表。

2）retrieveNearestPics16.py，根据传入的被选择的图片，查找16哈希值数据库，输出9张哈希值与被选择的图片最相似的图片

3）retrieveNearestPics32.p，可以根据传入的被选择的图片，查找32位哈希值数据库，输出9张哈希值与被选择的图片最相似的图片

（2）关键代码

1）连接数据库：

db=pymysql.connect(host,user,password,)

cursor = db.cursor()

2）创建数据库：

cursor.execute("CREATE DATABASE IF NOT EXISTS yalefaces\_database ")

3）创建表（以train\_dataset32为例）：

cursor.execute("DROP TABLE IF EXISTS TRAIN\_DATASET32")

create\_sql="""CREATE TABLE TRAIN\_DATASET32(

HASH\_CODE CHAR(32) NOT NULL,

FILE\_NAME CHAR(20) NOT NULL)"""

cursor.execute(create\_sql)

4）插入数据（以train\_dataset32为例）：

list=[]

train\_data32[train\_data32==-1]=0

for i in range(train\_data32.shape[0]):

a=train\_data32[i,:].astype(int).astype(str)

s=''.join(a)

list.append((s,"s"+str(i+1)+".bmp"))

insert\_sql="INSERT INTO TRAIN\_DATASET32(HASH\_CODE,FILE\_NAME) VALUES (%s,%s)"

try:

cursor.executemany(insert\_sql,list)

db.commit()

except:

db.rollback()

5）查表（以train\_dataset32为例）：

select\_sql="SELECT \* FROM TRAIN\_DATASET32"

cursor.execute(select\_sql)