**系统设计文档**

**——**关于“基于情感分析的养老院智能陪护系统”之人脸识别及情感分析

1.介绍

## 1.1目的

本文档为小组关于“基于情感分析的养老院智能陪护系统”之人脸识别及情感分析部分的系统设计说明书。该文档介绍了整个系统的整体网络结构和功能结构的概貌，从总体架构上给出整个系统的轮廓，对功能需求、性能需求和其它非功能性需求进行了详细的描述。。供概要设计人员和软件开发人员参考。

在此文档的指导下，小组成员开发过程保证软件开发的质量、需求的完整和可追溯性。为明确软件需求、安排项目规划与进度、组织软件开发与测试，主要是从用户角度（需求或市场人员根据用户要求编写）描述软件需要实现的功能，各个功能模块之间的关系与重要性。

该文档的使用者为：软件开发人员、软件测试人员、系统设计人员以及各类相关开发者。

## 1.2范围

本文档主要用于建构“基于情感分析的养老院智能陪护系统”之人脸识别及情感分析部分的逻辑模型，阐述说明一个问题“本系统的各种属性以及本系统是什么？”。通过详述软件的功能和为客户提供的服务，为开发人员划定简略的工作范围，通过较为清晰的结构使开发人员在合适的范围内进行开发。

由前面的需求分析，得出了软件的基本需求，和基本的数据流图，要实现整个项目需要对用户的需求进行设计，概要设计主要是利用比较抽象的语言对整个需求进行概括，确定软件的体系结构，主要描述“设计什么”和“如何设计”。根据需求规格说明书所列出的功能和性能需求，要明确说明本项目的设计思路、整体结构和主要有哪些模块组成，同时还要描述清楚各模块的结构及模块间信息传递的接口。为设计人员提供详细设计依据；为研发管理人员提供管理依据；为系统测试人员提供测试依据；为文档人员提供书写各种说明书的依据。

**1.3 本文工作和系统开发**

本文首先对系统的业务与用户的需求进行了分析，然后对系统的功能进行详细的设计，并在分析与设计的同时用 UML建模语言对其建模，采用 PowerDesigner 工具绘制描述各种模型的图形，目的使开发者与用户更有效的实现功能与理解需求，并使程序人员能顺利的实现系统，在系统的实现阶段重点讲述设计到编码的过渡，以体现 UML建模的模型对系统开发带来的好处。

本文的编写结合了面向对象的软件开发方法与 UML建模语言的优势与特色，同时与记账管理系统紧密结合。既体现了面向对象的软件开发思想 ，又包容了 UML建模语言所特有的语言建模优势。在编写中我们大量的使用 UML语言所提供的各种建模图形来描述记账管理系统，例如描述用户需求的用例图，类图，描述系统功能的顺序图，状态图。

## 1.4项目背景

1. 软件的名称：“基于情感分析的养老院智能陪护系统”之人脸识别及情感分析部分
2. 开发工具与语言：pycharm、Python。

3、用户：系统管理员。

# 总体设计

## 2.1功能分析

（1）采集人脸数据

目的：将老人、工作人员和义工的人脸图像收集起来，为训练人脸识别模型准备数据。

备注：人脸采集，需要用到人脸检测算法检测画面中是否有脸、有几张脸、和得到脸部坐标，所以该程序放到计算机视觉部分。

（2）功能：训练人脸识别模型

目的：用于检测陌生人、有老人微笑时判断是哪位老人、检测义工、和义工交互时是哪位老人

备注：陌生人检测、情感分析、义工交互检测都用到了人脸识别

（3）功能：训练情感分析模型

目的：用于检测哪位老人笑了

备注：情感分析模型和人脸识别模型同时使用，因为在检测表情的同时，还要识别出是哪位老人。

（4）功能：老人情感分析

目的：检测老人是否高兴

备注：情感分析分为2种情况，1）如果老人笑了说明老人高兴，2）如果老人没笑说明老人正常。一旦老人笑了，会将截图保存到硬盘，并实时插入一条数据到数据库。一旦数据库有了新数据，Web端的实时报表会立即呈现。

（5）功能：检测并追踪陌生人

目的：发现有陌生人就报警，并追踪。

备注：发现陌生人会将截图保存到硬盘，并实时插入一条数据到数据库。一旦数据库有了新数据，Web端的实时报表会立即呈现。

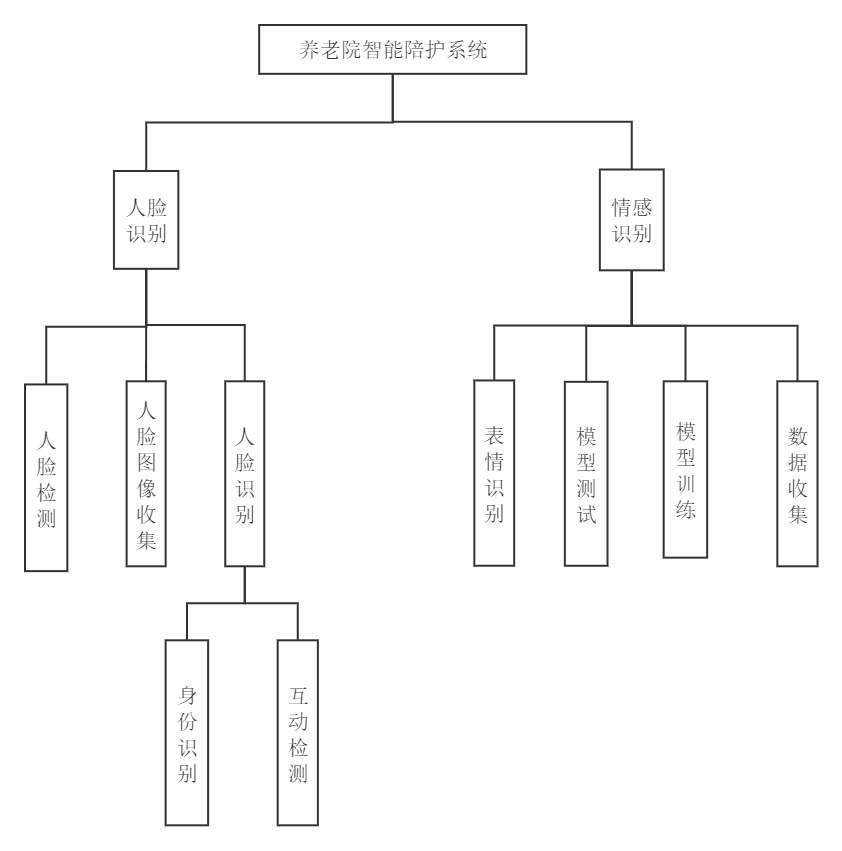
（6）功能：检测义工老人互动

目的：判断义工在和哪位老人互动

备注：判断条件为：义工和老人的距离在1米之内；一旦检测到有交互，会将截图保存到硬盘，并实时插入一条数据到数据库。一旦数据库有了新数据，Web端的实时报表会立即呈现。

**2.2 功能图**

根据上述功能，给出功能图

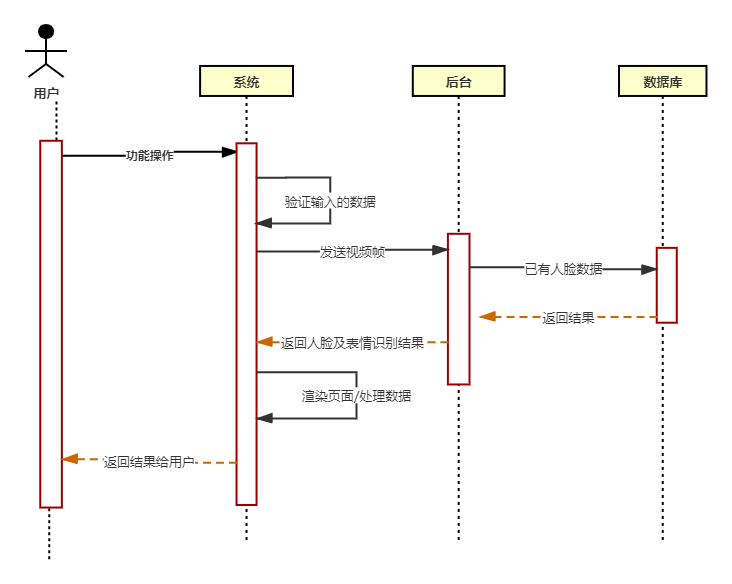


**2.3 模块设计**

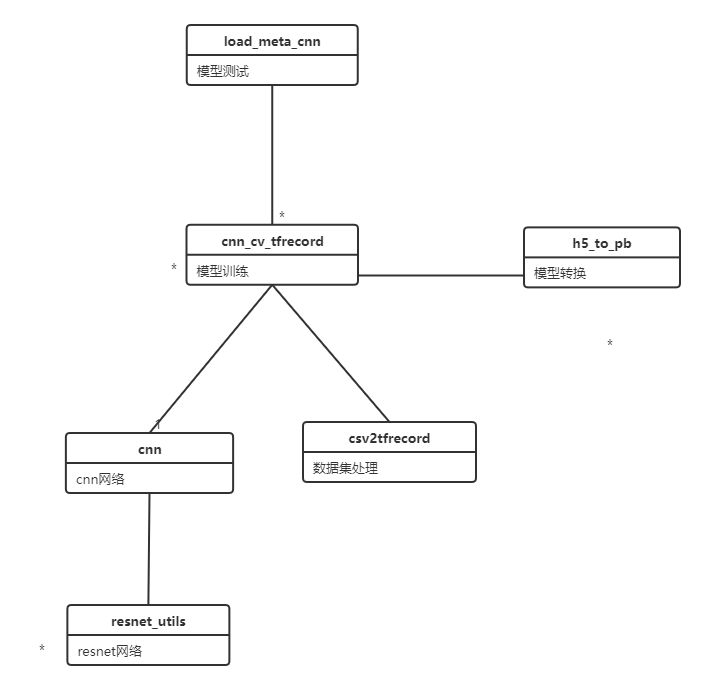
根据功能图，将系统分为两个模块实现

1. 人脸识别模块
2. 表情识别模块

**2.4 总体时序图**

****

**2.5 类图**

****

**2.6 结果展示**

****

1. **表情识别模块**
   1. **表情识别原理**

图像获取：通过人脸识别摄像头等图像捕捉工具获取静态人脸图像或动态图像序列。

图像预处理：图像的大小和灰度的归一化，头部姿态的矫正，图像分割等。这样做的目的是为了改善图像质量，消除噪声，统一图像灰度值及尺寸，为后序特征提取和分类识别打好基础

特征提取：将点阵转化成更高级别图像表述，如形状、运动、颜色、纹理、空间结构等， 在尽可能保证稳定性和识别率的前提下，对众多的图像数据进行降维处理。可以通过提取几何特征、统计特征、频率域特征和运动特征等方法进行特征提取。

1）采用几何特征进行特征提取，主要是对人脸表情的显著特征，例如眼睛、眉毛、嘴巴等的位置变化进行定位、测量，并确定其大小、距离、形状及相互比例等特征，从而进行表情识别。这个方法可与减少输入数据量，但会丢失一些重要的识别和分类信息，使精确性不高 。

2）基于整体统计特征的方法强调尽可能保留原始人脸表情图像中的信息，并允许分类器发现表情图像中相关特征，通过对整个人脸表情图像进行变换获取特征进行识别。可以通过PCA和ICA。这种提取方法会受到外来因素的干扰，例如光照、角度等，导致识别率下降

3）基于频率域特征提取，这是将图像从空间域转换到频率域提取其特征，主要通过小波变换，能够通过定义不同的核频率、带宽和方向对图像进行多分辨率分析，可以有效提取不同方向的细节程度的图像特征，不易直接用在匹配和识别，要ANN 或SVM 分类器结合使用，提高表情识别的准确率。

4）基于运动特征的提取，这是提取动态图像序列的运动特征，是今后研究的重点，主要利用到光流法。光流是指亮度模式引起的表观运动，是景物中可见点的三维速度矢量在成像平面上的投影，表示景物表面上的点在图像中位置的瞬时变化，光流场还携带有关运动和结构的丰富信息。其模型是处理运动图像的有效方法，将运动图像函数作为基本函数，根据图像强度守恒原理建立光流约束方程，通过求解约束方程，计算运动参数。这个方法反映了表情变化的实质，受光照不均性影响较小，但有比较大的计算量。

* 1. **模型介绍**

CNN 有2大特点：

（1）能够有效的将大数据量的图片降维成小数据量

（2）能够有效的保留图片特征，符合图片处理的原则

卷积神经网络 – CNN 解决的第一个问题就是「将复杂问题简化」，把大量参数降维成少量参数，再做处理。更重要的是：我们在大部分场景下，降维并不会影响结果。比如1000像素的图片缩小成200像素，并不影响肉眼认出来图片中是一只猫还是一只狗，机器也是如此。

保留图像特征

假如有圆形是1，没有圆形是0，那么圆形的位置不同就会产生完全不同的数据表达。但是从视觉的角度来看，图像的内容（本质）并没有发生变化，只是位置发生了变化。

所以当我们移动图像中的物体，用传统的方式的得出来的参数会差异很大！这是不符合图像处理的要求的。而 CNN 解决了这个问题，他用类似视觉的方式保留了图像的特征，当图像做翻转，旋转或者变换位置时，它也能有效的识别出来是类似的图像。

卷积神经网络-CNN 的基本原理

典型的 CNN 由3个部分构成：

（1）卷积层

（2）池化层

（3）全连接层

如果简单来描述的话：

卷积层负责提取图像中的局部特征；池化层用来大幅降低参数量级(降维)；全连接层类似传统神经网络的部分，用来输出想要的结果。

卷积——提取特征

卷积层的通过卷积核的过滤提取出图片中局部的特征，跟上面提到的人类视觉的特征提取类似。

池化层（下采样）——数据降维，避免过拟合

池化层相比卷积层可以更有效的降低数据维度，这么做不但可以大大减少运算量，还可以有效的避免过拟合。

全连接层——输出结果

这个部分就是最后一步了，经过卷积层和池化层处理过的数据输入到全连接层，得到最终想要的结果。经过卷积层和池化层降维过的数据，全连接层才能”跑得动”，不然数据量太大，计算成本高，效率低下。

CNN 有哪些实际应用？

（1）图像分类、检索

（2）目标定位检测

（3）目标分割

（4）人脸识别

（5）骨骼识别

* 1. **表情识别**

def cnn\_layer(self):

with tf.variable\_scope("conv1"):

conv1\_weight = tf.get\_variable('conv1\_weight', [self.conv\_size[0], self.conv\_size[0],

self.channel, self.conv\_feature[0]],

dtype=tf.float32, initializer=self.initializer)

conv1\_bias = tf.get\_variable('conv1\_bias', [self.conv\_feature[0]], dtype=tf.float32,

initializer=tf.constant\_initializer(0.0))

conv1 = tf.nn.conv2d(self.x\_input, conv1\_weight, [1, 1, 1, 1], padding='SAME')

conv1\_add\_bias = tf.nn.bias\_add(conv1, conv1\_bias)

conv1\_relu = tf.nn.relu(conv1\_add\_bias)

norm1 = tf.nn.lrn(conv1\_relu, depth\_radius=5, bias=2.0, alpha=1e-3, beta=0.75, name='norm1')

with tf.variable\_scope("conv2"):

conv2\_weight = tf.get\_variable('conv2\_weight', [self.conv\_size[1], self.conv\_size[1],

self.conv\_feature[0], self.conv\_feature[1]], dtype=tf.float32, initializer=self.initializer)

conv2\_bias = tf.get\_variable('conv2\_bias', [self.conv\_feature[1]], dtype=tf.float32,

initializer=tf.constant\_initializer(0.0))

conv2 = tf.nn.conv2d(norm1, conv2\_weight, [1, 1, 1, 1], padding='SAME')

conv2\_add\_bias = tf.nn.bias\_add(conv2, conv2\_bias)

conv2\_relu = tf.nn.relu(conv2\_add\_bias)

pool2 = tf.nn.max\_pool(conv2\_relu, ksize=[1, self.maxpool\_size[1], self.maxpool\_size[1], 1],

strides=[1, self.maxpool\_stride[1], self.maxpool\_stride[1], 1],

padding='SAME', name='pool\_layer2')

norm2 = tf.nn.lrn(pool2, depth\_radius=5, bias=2.0, alpha=1e-3, beta=0.75, name='norm2')

with tf.variable\_scope("conv3"):

conv3\_weight = tf.get\_variable('conv3\_weight', [self.conv\_size[2], self.conv\_size[2],

self.conv\_feature[1], self.conv\_feature[2]], dtype=tf.float32, initializer=self.initializer)

conv3\_bias = tf.get\_variable('conv3\_bias', [self.conv\_feature[2]], dtype=tf.float32,

initializer=tf.constant\_initializer(0.0))

conv3 = tf.nn.conv2d(norm2, conv3\_weight, [1, 1, 1, 1], padding='SAME')

conv3\_add\_bias = tf.nn.bias\_add(conv3, conv3\_bias)

conv3\_relu = tf.nn.relu(conv3\_add\_bias)

pool3 = tf.nn.max\_pool(conv3\_relu, ksize=[1, self.maxpool\_size[2], self.maxpool\_size[2], 1],

strides=[1, self.maxpool\_stride[2], self.maxpool\_stride[2], 1],

padding='SAME', name='pool\_layer3')

norm3 = tf.nn.lrn(pool3, depth\_radius=5, bias=2.0, alpha=1e-3, beta=0.75, name='norm3')

with tf.variable\_scope("conv4"):

conv4\_weight = tf.get\_variable('conv4\_weight', [self.conv\_size[3], self.conv\_size[3],

self.conv\_feature[2], self.conv\_feature[3]],dtype=tf.float32, initializer=self.initializer)

conv4\_bias = tf.get\_variable('conv4\_bias', [self.conv\_feature[3]], dtype=tf.float32,

initializer=tf.constant\_initializer(0.0))

conv4 = tf.nn.conv2d(norm3, conv4\_weight, [1, 1, 1, 1], padding='SAME')

conv4\_add\_bias = tf.nn.bias\_add(conv4, conv4\_bias)

conv4\_relu = tf.nn.relu(conv4\_add\_bias)

pool4 = tf.nn.max\_pool(conv4\_relu, ksize=[1, self.maxpool\_size[3], self.maxpool\_size[3], 1],

strides=[1, self.maxpool\_stride[3], self.maxpool\_stride[3], 1],

padding='SAME', name='pool\_layer4')

norm4 = tf.nn.lrn(pool4, depth\_radius=5, bias=2.0, alpha=1e-3, beta=0.75, name='norm4')

return norm4

# 4.人脸识别模块

## 4.1 人脸识别原理

### **人脸识别关键问题：**

* 找出所有人脸
* 在不同光线和方向下识别出同一人脸
* 找出每张脸与他人的区别
* 将人脸与已知人脸比较以确定其身份

### **人脸识别步骤：**

#### **（1）. 人脸检测**

要想识别人脸，首先需要在图像或者视频帧中找到所有人脸的位置，并将人脸部分的图像切割出来。可以使用****方向梯度直方图****(HOG)来检测人脸位置。先将图片灰度化，因为色彩对于找到人脸位置并无明显作用，接着计算图像中各像素的梯度。通过将图像变换为HOG形式，我们就可以提取图像的特征，从而获取人脸位置。

#### **（2）. 人脸对齐**

一张图片中的人脸可能是倾斜的，或者仅仅是侧脸。为了方便给人脸编码，需要将人脸对齐成同一种标准的形状。人脸对齐的第一步就是人脸是特征点估计。Dlib有专门的函数和模型，能够实现人脸68个特征点的定位。找到特征点后，就可以通过图像的几何变换（仿射、旋转、缩放），使各个特征点对齐（将眼睛、嘴等部位移到相同位置）。

#### **（3）.人脸编码**

#### 训练一个神经网络，将输入的脸部图像生成为128维的预测值。训练的大致过程为：将同一人的两张不同照片和另一人的照片一起喂入神经网络，不断迭代训练，使同一人的两张照片编码后的预测值接近，不同人的照片预测值拉远。也就是减小类内距离，增大类间距离。具体算法参考facenet[3]。

#### **（4）. 识别身份**

预先将所有人的连放入人脸库中，全部用上述的神经网络编码为128维并保存。识别时，将人脸预测为128维的向量后，与人脸库中的数据进行比对。比对方法有很多种，可以直接找出阈值范围内欧氏距离最小的人脸，或者训练一个末端的SVM或者knn分类器，直接生成人的代号（身份）。

**4.2 使用模型**

face\_recognition是一款免费、开源、实时、离线的Python人脸识别库，使用世界上最简单的人脸识别库，在Python或命令行中识别和操作人脸。使用dlib最先进的人脸识别技术构建而成，并具有深度学习功能。

Face\_recognition库基于dlib，故使用该库需要先安装dlib。

Windows系统安装dlib可直接在网站https://pypi.org/simple/dlib/ 下载对应版本的whl文件，用pip安装，简单高效。

**4.3人脸识别**

import face\_recognitionimport cv2

input\_movie = cv2.VideoCapture("test1.mp4")

length = int(input\_movie.get(cv2.CAP\_PROP\_FRAME\_COUNT))

# Load some sample pictures and learn how to recognize them.

lmm\_image = face\_recognition.load\_image\_file("jiangwen.png")

lmm\_face\_encoding = face\_recognition.face\_encodings(lmm\_image)[0]

al\_image = face\_recognition.load\_image\_file("huzhiyu.png")

al\_face\_encoding = face\_recognition.face\_encodings(al\_image)[0]

known\_faces = [

lmm\_face\_encoding,

al\_face\_encoding]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

frame\_number = 0

while True:

# Grab a single frame of video

ret, frame = input\_movie.read()

frame\_number += 1

# Quit when the input video file ends

if not ret:

break

# Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

rgb\_frame = frame[:, :, ::-1]

# Find all the faces and face encodings in the current frame of video

face\_locations = face\_recognition.face\_locations(rgb\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)

face\_names = []

for face\_encoding in face\_encodings:

# See if the face is a match for the known face(s)

match = face\_recognition.compare\_faces(known\_faces, face\_encoding, tolerance=0.50)

# If you had more than 2 faces, you could make this logic a lot prettier

# but I kept it simple for the demo

name = None

if match[0]:

name = "Lin-Manuel Miranda"

elif match[1]:

name = "Alex Lacamoire"

face\_names.append(name)

# Label the results

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

if not name:

continue

# Draw a box around the face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 25), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 0.5, (255, 255, 255), 1)

# Write the resulting image to the output video file

print("Writing frame {} / {}".format(frame\_number, length))

output\_movie.write(frame)

# All done!

input\_movie.release()

cv2.destroyAllWindows()

**5、结语**

本文从面向对象的思想出发，讨论了UML建模的语法结构和UML的建模思路与过程， 并基于此对系统进行了分析，设计，以及最后的实现与测试。在统一过程开发中，把一个较大的项目的功能分配到一个个用例上去实现，让开发过程变得比较轻松，更适合多个人共同开发; 同时，通过开发阶段的限制和迭代的划分，使开发过程的透明度更高，目标更明确，整个项目开发的风险也降低了。当然，由于时间关系，系统还有很多不完善的地方，对于系统高要求标准做的还不够，这也是我们日后努力的方向。