年龄估计算法技术实现

学生姓名：刘畅

院　　系：计算机科学与技术

班　　级：计算机 0710 班

指导教师：马丙鹏

2011 年 6 月 9 日

摘要

对人脸图像进行年龄估计是图像处理领域的一个重要课题，该研究具有重要的科学意义和应用价值。在现实世界中，电子顾客关系管理系统、安全控制和监控模拟、生物测定学，娱乐等方面都有相关的人类年龄估计的应用，它已经变成了计算机视觉方面的一个热门话题。本文在国内外已有的研究基础上，实现了年龄估计系统，并获得了较好的实验结果。

本文描述了 BIF（Bio-Inspired Feature，BIF）算法的相关理论，并针对算法流程给出了具体的实现及优化，实现了算法从 MATLAB 平台向 C++平台的移植，最终通过和原始的 MATLAB 版本产生的数据结果进行比较，表明代码的正确性和高效性。在完成了算法的设计和优化后，本文采用面向对象的编程方法，利用 VC++ 6.0 设计出人脸年龄估计系统，该系统包括了图像预处理，人脸检测，人脸建模，人脸模型匹配，特征参数提取，支持向量机(SVM)训练，年龄分类等这些功能。

关键词：人脸识别，年龄估计，支持向量机，生物仿生特征

Abstract

Age estimation on facial image is an important task in image processing; the research is of great scientific importance and instrumental value. In reality, electronic customer management system, security control, monitored control system, biometrics and entertainment all have relative applications on human age estimation; it has become a hot topic in computer vision. This paper design and build the human age estimation system based on the research at home and abroad, it shows a good result in our estimation system.

This paper describe the theory about the Bio-Inspired Feature Method and gives the concrete implement and improvement based on the process of the algorithm, accomplishing the transplantation from MATLAB platform to C++ platform. In the end, by comparing the new result with MATLAB result, it shows the correctness and efficiency of the code. After finishing the design and optimization of the algorithm, based on object-oriented programming method, this paper uses VC++ 6.0 to develop facial age estimation system. This system includes image pre-processing, face detect, face model, model matching, feature extraction, SVM training and age recognition.

Keywords: Face Recognition, Age Estimation, Support Vector Machines, Bio-Inspired Feature

目录

1 绪论

1.1 国内外发展概况

作为心灵的窗口，人脸传递了很多和个体特征有关的重要视觉信息。在过去几十年，自从图像处理技术在图像和计算机视觉方面获得广泛的应用之后，面部特征呈现出来的人脸特征，比如个人身份、面部表情、性别、年龄、种族起源以及姿态，都极大的引起了工业界和学术界的注意。其中，有两个基本的问题一直促使这些技术不断发展，它们分别是：

⑴ 人脸图像合成：用自定义的单独的或者混合的面部特征来表示图像，比如身份，表情，性别，年龄，种族，姿态等等。

⑵ 人脸图像分析：用面部特征来解释面部图像，比如表情，性别，年龄，种族，姿态等等。

在这些方面中，由于新的应用的出现，近年来基于面部图像的年龄合成和估计已经变成了一个很热门的话题。人类在早期的发育过程中，已经形成了一种能力，这种能力能够让我们判断介于 20 岁到 60 岁的年龄，并利用分组决策的方法以较高的精度从面部图像中构想出衰老的外表。比如，我们能够很容易想象出爱因斯坦的衰老过程，就像图 1.1 所示的一样。在社交网络中，人脸年龄已经被认为是一个重要的线索。机器能够像人类一样表现良好吗？计算机科技的进步已经给这个问题一个肯定的答案。在年龄合成和估计领域有两个基本的任务，如下：

(1) 年龄重现：使用自然的衰老或者年轻效果在每一个人脸上以美观的重现人脸图像。

(2) 年龄估计：用一个确切的年龄或者年龄段来自动标记每一个人脸图像。

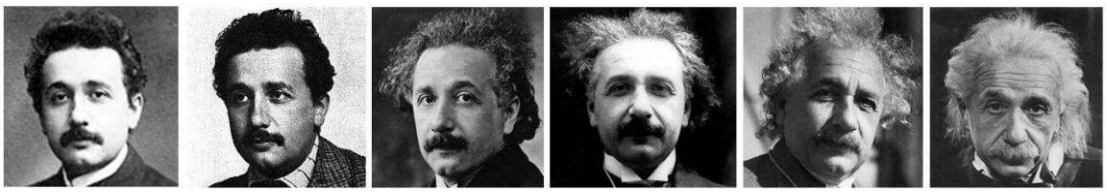

图 1.1 爱因斯坦的面部衰老过程

根据调研，Young 和 Niels[1]是最早提出年龄估计的人。早在 1994 年，他们就提出通过人脸图像进行年龄估计，但是他们的工作较为简单，只是把年龄粗略地分成：小孩、年轻人和老年人三种。2002 年，Hayashi 等人[2]研究了基于 Hough 变换的皱纹纹理和人脸图像肤色分析的年龄和性别识别方法。2003 年，Iga 等人

用支持向量机(SVM)开发了一个用于估计年龄的分类器。2004 年，Lanitis 等人[3]提出一种基于面部外观的统计模型。通过比较了不同分类器的性能，他们认为机器几乎可以和人一样估计出人的年龄。同年，Nakano 等人[4]提出了基于图像边缘信息的年龄分类算法。根据肤色模型将皮肤区域从背景中分割出来，然后使用 sobel 算子进行边缘检测，将超过一定阈值的像素点个数作为特征送入神经网络中进行学习。2005 年，Zhou 等人[5]提出用 Boosting 的方法作为回归方法进行年龄的估计，并用实验表明该方法比基于 SVMs 的方法还要好。2006 年，Geng 等人[6]提出衰老模式子空间的方法，通过学习一些代表性的子空间来建模衰老模式，利用子空间投影的方法重构人脸图像的衰老模式，从而得到该图像在衰老模式下的位置，由此来估计年龄。2008 年，Guo 等[7]用子空间学习方法得到衰老流形的低维嵌入，从而提取出人脸与衰老相关的特征，并设计了一个局部可调节回归方法用于学习和估计年龄。

在国内，关于年龄估计方面的研究起步较晚，文献[8]研究了基于 Boosting RBF 神经网络的人脸年龄估计方法，首先利用非负矩阵分解方法提取人脸特征，然后通过 RBF 神经网络确定一个人脸图像及其相符年龄之间的估计函数。通过利用 Boosting 方法构造一个基于神经网络的函数序列，将它们组合成一个加强的估计函数，来提高神经网络的泛化能力和故障诊断的准确性，从而进行年龄估计。文献[9]提出一种基于人工免疫识别系统的年龄估计方法, 先利用 AAM 方法提取用于年龄估计的人脸特征，然后利用人工免疫识别系统方法，进行人脸图的年龄估计。

年龄估计是一个比较复杂的问题。这是因为，人的年龄特征在外表上很难准确地被观察出来，即使是人用眼睛也很难准确地判断出一个人的年龄。人脸的年龄特征通常通过皮肤纹理、皮肤颜色、光亮程度和皱纹纹理等方面表现出来，然而这些因素通常与个人的遗传基因、生活习惯、性别、性格特征和工作环境等方面相关。因此，很难用一个统一的模型去定义人脸图像的年龄，通常需要通过大量样本的学习才能较好地估计出人的年龄层次。

1.2 课题的目的和意义

人脸年龄估计是生物测定学研究的内容之一，是模式识别领域中的一个前沿课题，该课题的研究已有多年的历史。目前人脸年龄估计越来越成为当前模式识别和人工智能领域的一个研究热点。人脸年龄估计系统作为一种重要的人脸识别领域的应用，可以广泛地应用于电子顾客关系管理系统、安全控制和监控模拟、生物测定学，娱乐等方面多种场合。因而，人脸年龄估计问题的研究不仅具有重大的实用价值，而且在模式识别中具有重大的理论意义。

因此，实现该系统对于我们学习和研究年龄估计具有很重要的意义。毕业设计的完成，系统的实现，不仅是理论上对新的研究成果的认识，而且锻炼了自己在大学四年来所学的知识和培养的能力，增加了自学能力，编码能力和沟通交流的能力，为本人以后的工作和学习打下坚实的基础。

1.3 课题要解决的问题

本课题要实现的系统主要包括三个部分，第一部分是人脸检测部分，主要实现的功能是从一张完整的人脸中识别并裁减出相关的脸部图像；第二部分是特征提取部分，主要实现的功能是对上面裁减后的人脸图像利用 BIF 算法处理，获得提取后的特征；第三部分是进行 SVM 分类，具体就是提取后的特征进行相关的处理，然后利用分类器分类，获得最终的结果——年龄。

在对模式识别的基本步骤进行了解后，着重理解了基于生物特征的年龄估计的相关理论部分，接着阅读了 MATLAB 关于生物特征估计的代码，在理解代码之后，绘制相关的流程图，并利用 C++语言进行重写，以完成 BIF 算法部分的编码。此部分要解决的问题是：

(1) 如何进行图像的表示和图像块的存取
(2) 如何利用 MATLAB 提供的 C++接口编写 MEX 程序
(3) 如何测试改写后算法的正确性和可靠性，并结合 VC++的特点优化

本课题的第二部分是具体利用实现的 C++代码在基于 VC++的平台上实现年龄估计系统，这部分主要解决的问题是：

(1) 如何设计和实现各个模块的类及函数接口
(2) 如何利用 VC++编写相关的界面响应函数
(3) 如何进一步提高系统进行自动年龄估计的性能

预计要达到的功能目标是由用户指定输入的图像数据或者摄像头捕捉的视频图像，然后系统经过人脸检测、人眼对准、特征提取、分类器分类等流程，最终在右侧的界面处显示截取的人脸部分以及相应的估计的年龄值。同时，双击右边的图像框能够弹出对话框，显示完整的图像信息，并能够利用鼠标的左右键选中人眼中心，手动标注人眼并重新估计。最终，实现该图像的估计任务。

2 年龄估计算法

2.1 年龄估计算法介绍

目前现存的使用面部图样进行年龄估计的系统一般是由两个模块组成：图像表示和年龄估计。对于图像表示，常见的方法有人体测量学模型、动态面部模型（AAM）、年龄模式子空间（AGE）、年龄复现，基于块表情的模型等。一旦给定了一个年龄特征的表示，下面一个步骤就是估计年龄。年龄估计可以被认为是一种多级分类的问题，一个回归的问题，或者是这两个问题的混合体。

本文使用的年龄估计算法模型采用的是生物特征模型（Bio-inspired Feature），在特征表示出来后，我们进行特征的降维处理，然后进行统计学上面的学习，获得分类器，然后根据训练获得的分离器进行年龄分类，获得待估计图像的年龄。整个年龄估计算法的流程如图 2.1 所示。

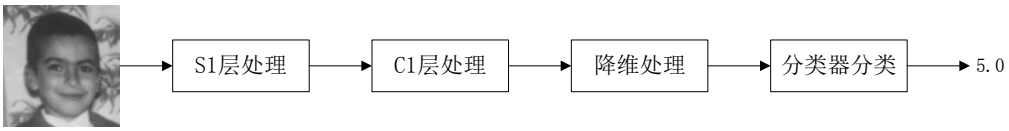

图 2.1　年龄估计算法的流程

2.2 算法流程详述

在上节中，图 2.1 已经大致的描述了年龄估计的算法流程，本节将详细介绍该算法的处理过程。该年龄估计算法大致可分为四步：

(1) 面部灰度图像进行 S1 层上的处理，包括预处理和 Gabor 变化等步骤。

(2) 处理后的特征再经过 C1 层处理，包括相邻求最大值和标准差的处理，结果产生的 C1 特征大概有 7000 个左右。

(3) 将获得的特征进行降维处理，获得较少维度的特征。

(4) 使用学习后获得的分离器或者回归器进行年龄估计，最终输出的结果即为估计的年龄。

下面详细的描述每一步的理论细节：

S1 单元：一个灰度级别的图像输入首先通过简单的 S1 单元被分解。该 S1 单元和 Hubel and Wiesel[10]在最初的可视皮层中发现的经典的简单单元相对应。高波函数是针对 S1 单元，该 S1 单元为皮层简单细胞可视区域提供一个很好的模型。

高波函数如下：

$$G(x,y) = \exp\left(-\frac{(X^2+\gamma^2Y^2)}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}X\right) \quad (2.1)$$

式中，X 和 Y 的表示含义如下：

$$X = x\ \cos\theta + y\ \sin\theta \quad (2.2)$$

$$Y = -x\ \sin\theta + y\ \cos\theta \quad (2.3)$$

式中，X,Y 是高波滤波器的带有θ 角度的旋转，θ 的范围是从 0 到π 。高宽比是固定的，γ =0.3,有效的宽σ ，波长λ 以及滤波器的尺寸 s 都是可以变化的。

方向 θ 以不同的间距统一的从 0 到π 变化，导致结果是不同数量的总的方向。方向的最好的数据是由我们年龄估计问题中的数据决定的。就像 Serre，Wolf，Poggio 的工作[11]一样，S1 分离器被安排用来形成一个尺度的棱锥，在一个系列范围内变化。但是在实验中发现从一个较小的尺寸中开始能够产生更好的结果。原因可能是：

(1) 年龄估计问题中的图像尺寸大小是 64x64，这远小于物体分类识别中的尺度。比如，在参考文献[12]中，所有的输入图像都被降低取样到 140 像素，并且保持了高宽比。

(2) 很小的面部细节对于使细微的年龄差距特征化很重要。另外，在 Serre 的工作[12]中带的数量已经被选定并固定为 8，因此总共有 16 个尺寸，但是，试验中让数据来决定带的最佳数量，从 2，4，6，8 中选择。S1 的可接受的视野类型的数量是由尺寸和方向的结果来决定的。比如，当通过算法选择是 16 个尺度和 12 个方向的时，它可以有 192 种类型，此时角度分别为 0°，15°，30°……165°。

C1 单元：皮层复杂细胞倾向于较大的可接受的视野，C1 单元对应着这些皮层复杂细胞。C1 单元从先前的层，以相同的方向和尺度带在 S1 单元上面合并。每一个尺度都有一对相邻的分离器尺寸。S1 单元的尺度带的索引决定了 S1 相邻单元在 C1 单元上合并的尺寸。在之前的模型中，最大运算符“MAX”被使用作为合并的分离器。Riesenhuber and Poggio[13]探讨并证明了使用非线性的运算符“MAX”而不是线性求和运算符“SUM”的好处，这里使用另外一种非线性的运算符，标准差“STD”的定义如下：

$$std = \sqrt{\frac{1}{N_s \times N_s}\sum_{i=1}^{N_s \times N_s}(F_i - \bar{F})^2} \quad (2.4)$$

公式 2.4 揭示了 S1 单元相邻的N_sxN_s单元中数据的变量性。这里，F_i是同一个尺度带（但是不同的分离器）的相邻两个 S1 单元在索引为 i 时的像素最大值，定义如下：

$$F_i = \max\ (x_i^j, x_i^{j+1}) \quad (2.5)$$

式中，x_i^j和x_i^{j+1}是在位置为 i 时尺度分别为 j 和 j+1 的分离器的值。$\bar{F}$是相邻的$N_s xN_s$单元中分离器值的平均值。在另一个相邻尺度（比如同一个尺度带上）上面合并的操作符“MAX”增加了 2 维转换的容错，比如在一个小数量上面的尺度变化。“MAX”操作符使用相同方向但是不同尺度的分离器合并两个待分离的图像为一个，然后 STD 运算符在合并后的相邻$N_s xN_s$上面操作一次。在之前的模型中，在合并的图像上面进行的操作符是另外一个“MAX”分离。尽管第二次“MAX”能够容许更多的波动和尺寸变化，但是它不能表示数据中的变量性。在年龄估计中，数据的相邻变化性的描述对于细微的年龄差距可能是很重要的，比如脸部的皱纹。

“MAX”合并和“STD”运算符对于每一个方向和尺度带都是独立进行的。比如，考虑第一个带：S = 1。对于每一个方向，它包含两个S_1映射：一个是通过使用大小为 5 x 5 的分离器来获得的，另一个是使用大小为 7 x 7 的分离器获得。这两个映射虽然有相同的大小，但是由于使用不同的分离器，所以不是相同的值。然后运用“STD”运算符作用在最大的映射上，使用一个格的尺寸为$N_s xN_s$ = 6 x 6。对于每一个格单元，从 36 个元素中计算出单个值。C1 响应仅仅是在一个方向上面的一半相邻尺度间隔才被计算出来，比如$N_s/2$。结果是，两个相邻的格单元中仅仅有一半交叠，而这个会极大的减少 C1 层的特征数量，从而使得年龄估计的过程更有效率。

特征降维：当使用预先学习的原型来用于在 S2 单元上面的模板匹配时，C2 上面的最终特征维数是由 S2 上用于比较的原型数量决定的。一个典型的原型数量设置为 N = 1000。这里并不使用 S2 和 C2 单元。C1 特征直接进行合并，从而对每一张图像形成一个特征向量。结果的特征有很高的维度，减少这个维度会使得算法更高效。之前，使用用于物体分类识别或者人类识别的一个相关元素分析产生的一个高权值来选择 C2 特征。这里对于生物特征 C1 使用简单的主成分分析（PCA）方法就能够获得很好的结果。

PCA 是一种线性的转换技术。让 $X = [x_1, x_2 \cdots\cdots x_n]$作为维度 D 的特征向量，这个维度 D 是有 C1 单元在 n 个训练的脸部上训练出来的。X 是训练数据的平均向量。降维的目的是为了找到一个 n x d 的矩阵 P，满足:

$$Y = P^T X \tag{2.6}$$

式中，$Y = [y_1, y_2, y_3, \cdots\cdots y_n,]$是维度 d 的投影新的特征，d << D。

PCA 方法找到了两者之间的联系，使得投影的便利最大化：

$$p = \arg\max_{\|p\|=1} p^T S p \tag{2.7}$$

$$S = \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T \tag{2.8}$$

PCA 技术能够有效的用于生物特征 C1 的降维处理。使用中发现，当使用减少维

数后的特征时年龄估计的错误并没有发生很大的变化。在一些案例中，错误甚至要小于使用所有的 C1 特征。

分类或者回归：当每一个年龄都作为一个分类的标签时，年龄估计可以被认为是一个分类问题。另一方面，年龄估计又可以被认为是一个回归问题，此处每一个年龄被作为一个回归的值。在实验中，遵循论文[14]中的建议，选择使用线性 SVM 用于年龄分类，从而最终获得估计的年龄值。

3 年龄估计算法实现

3.1 程序综述

程序主要实现特征提取的功能，具体来说就是输入一副特定大小（如 64x64）的灰度图像，根据前面算法的流程，能够通过 S1 层和 C1 层的处理，输出相应的提取后的特征。程序的编写完全脱离了 MATLAB 的环境，不需要采用任何的 MATLAB 提供的 lib 库文件，只需要输入相应的数据，就能够获得最终的特征。数据以 dat 文件存储，接口参数为 4936(64x64)维数的一维数组 ImgData、图像的尺寸（宽和高）组成的一维数组 ImgSize 以及相应的尺度 Scale；输出参数为对应的特征数组 outputGabor 以及该数组的大小 size。整个程序充分的考虑到内存的优化和效率的提升，特别是 MATLAB 和 VC 各自格式的表示和处理方便，实现代码的简洁和高效。

3.2 程序模块介绍

由于实现这部分的功能只是我们整个年龄估计系统的一个模块，所以该部分才用一个类 BIF 来实现，定义相关的类成员变量和接口函数，以方便进行各种数据操作。

3.2.1 类成员变量介绍

```
int pos_buffer[8];      //定义每个位置下标记录相对于上一次记录的偏移量
int TempDim[8];         //定义产生的子矩阵的列数
int pos_num[8];         //定义每个位置下标记录的长度
double * pos_all;       //定义产生的位置下标的一维数组指针
int dim; //指示一张图片有多少个元素，例如 1024*94，这里就是 1024
double * InputImgData;      //拷贝指针，指向图像单元的指针。
int numGabor;               //定义 Gabor 数量
int HeiWid;             //定义宽和高的乘积
int Height;             //定义输入图像的高
int Width;              //定义输入图像的宽
const int SxSy;         //常量，固定参数
int Scale;                  //定义变化的尺度
int m_nOrientation;         //定义变化的方向
int kernelsize[2];      //定义 Gabor 核心的宽和高
```

3.2.2 类成员函数介绍

BIF(double *ImgSize, int m_nScale);// 类构造函数，实现成员和索引初始化

virtual ~BIF(); //类析构函数，释放动态内存空间

void segmentimage(double *ImgData,double *ImgSize,double *Pos,double *output_newImgData,int *output_pos); //图像块索引函数，采集特定图像块索引

void imgblockindex(double *ImgSize,double *blockSize,double *gapSize,double *output_pos_all); //图像索引排列函数，采集特定参数的图像所有索引并排序

void BifTransform(double *ImgData, int ImgDataSize,double *outputGaborFeature,int &size);//BIF 算法实现函数，特征提取获得图像特征

void DataConverse(double *inputGabor,double *outputGabor);//数据格式转换函数，实现 VC 格式数据转换为 MATLAB 格式。

3.2.3 函数实现

（1）构造函数 BIF

输入参数：ImgSize 表示输入图像数据的宽和高这个数组的指针，对应一个两个元素的一维数组[width,height], m_nScale 表示处理图像时候用到的尺寸。这里 ImgSize 可以采用[32,32]或者[64,64],m_nScale 可以采用 16 或者 12。实际的系统实现中为了获得较高的精度，采用了 ImgSize =[64,64]的图像大小和 m_nScale=16 的尺度。

函数说明：构造特定类型（输入可变的图像尺寸和尺度）的模型，比如输入参数为 ImgSize = [32,32], m_nScale=16,则表示创建一个图像大小为 32x32，图像尺度为 16 的类变量，这样对于后面处理多幅图像的时候，只需要调用一次构造函数就能够创建出该变量，并通过调用该变量的成员函数 BifTransform 来传递图像数据参数，实现特征提取。通过该构造函数的使用，可以极大的提高运行效率，节省运行开销。

（2）析构函数~BIF

函数说明：析构相应的动态类成员变量并做收尾处理，对于类成员变量中的部分指针，由于函数处理中会使用到大量的动态数组，所以会通过 new 运算符来对成员变量指针赋值，在使用完毕后，需要清空相应的内存单元，避免内存泄露。这里主要析构索引数组 pos_all 和图像数据 InputImgData。

（3）图像块索引函数 segmentimage

输入参数：ImgData 指示输入一维的图像数组指针，ImgSize 指示图像数据宽和高这个数组的指针，Pos 指针指示一个数组，该数组由四个元素组成，分别是起始点的左边 x,y 以及切割图像时候的子块宽和高 sub_width 和 sub_height，output_newImgData 指示切割图像后的图像元素，output_pos 指示切割图像时的像素的索引值。例如大小为[32,32]的图像首先通过块拷贝函数 memmove 转换为大小为 1024 的一维数组 ImgData，传入该函数；然后从坐标为[5,5]处开始切割，

子块的宽和高均为 6，那么数组 Pos 即为[5,5,6,6]，处理完毕后将取出来的索引放在 output_pos 指向的数组中，索引对应的像素值放在 output_newImgData 指向的数组中。

函数说明：该函数的功能是在原始图像上切取合适大小的子块，并保留这些切取的子块的每一个像素索引值在 output_pos 数组中，然后将切取的这些图像像素保存在新的数组 output_newImgData 中。此函数实现了取图像的块索引，用于获得原始图像的一部分图像。具体实现过程是：

首先获得 x 方向上的终点值，这里虽然给定了 width，但是由于 x 的初始值 Pos[0]以及 sub_width 的存在，所以取得的子块的横坐标的最大值并不一定是 width，也可能是 x+sub_width-1,因此取得 x 方向的终点值为 min(width, x + sub_width - 1)。

其次就是要获得 y 方向上每次递增的量 sub_height，虽然该变量由 Pos 数组直接给出，但是由于需要在给定规格的图像上切割，所以受图像大小的限制，为此同样需要取最小值 min(sub_height, height – y + 1)。

最后就需要从起始坐标(x, y)开始找到每一个像素点对应存储数组中的索引值。找到起始点横坐标为 x，那么对应之前的 x-1 列，每一列的元素都有 height 个，而且在本列（第 x 列）中前 y-1 行也都空缺，因此点(x, y)对应的一维数组下标应该是(x-1)*height+y, 这样直到了起始点的索引，对于其他点的索引就很容易得到相关的值。在循环中，用变量来表示横坐标的变化，相应的求出不同纵坐标对应的索引值，然后依次排序，即可得到我们需要的子块的全部索引和对应的图像数据。

这里对于该段代码的编写需要注意的是，MATLAB 按照列存储，因此相应的代码中我们使用的是列序，而由于最终的系统需要移植到 VC 代码中，因此最终要按照行序来排列，由于前后处理效率差异不大，所以我选择在提取完所有的索引值后，再对索引重排列为行序格式。

（4）图像索引排列函数 imgblockindex

输入参数：ImgSize 指示图像的宽和高[width,height]，blockSize 指示子块的宽和高[sub_width,sub_height],gapSize 指示间距[gap_width,gap_height],output_pos_all 指示用来存放下标的输出数组。具体来说就是，要通过不断的对图像取子块来获得索引值，而子块的起始点相距上一点相差[gap_subwidth,gap_height]，比如起始位置为[1,1]，而子块大小为[6,6]，间距 gapSize 为[2,2]。那么在调用 segmengtimage 后，下一个子块的起始点就是[3,3]，子块大小仍然不变。

函数说明：函数实现对图像取块的索引，并对索引排序。通过调用上面的函数 segmentimage，实现对于一张完整图像，根据相关参数取得块索引并按照先后顺序保存该索引值。

在前面关于 segmentimage 函数的介绍中，该函数能够实现对于给定子块和原始图像参数的情况下求得子块索引的功能。这里函数 imgblockindex 函数则通过对该函数的循环调用，获得原始图像的全部索引，并按照相应的顺序来存放在一个数组中。具体的流程是：

首先，通过参数获得相应的参数，包括图像数据和截取子块参数，然后根据这些信息，通过循环调用 segmentimage 函数来获得下标。在每次循环中，由于子块参数在传入进来之后就保持不变，一直都是 sub_width 和 sub_height，但是起始点却在不断变化，具体来说就是循环体内，横坐标方向上从 1 开始，每次增加 gap_width，直到达到了 width-sub_width+1,同时在纵坐标方向上，从 1 开始，每次增加 gap_height,直到达到了 height-sub_height+1。这样变化的参数就只有横纵坐标，然后依次用该参数传入 segmentimage 函数中动态获得新的数组的索引值，并设置 index 参数，用来保存下来每次的索引，循环结束后即可得到全部的索引，并且获得已经排序好的值。

在代码的相关实现部分需要注意的是，MATLAB 能够灵活的处理动态数组的增减、拷贝、叠加，而在 C++中，要实现动态数组需要在堆上开辟空间，使用完后需要及时的释放空间，避免内存泄露，因此，在实现相关功能的部分，C++代码需要编写相关函数来处理数组操作，同时注意到下标的转换和起始位置的不同，在实现上会有更多的细节需要注意。

（5）BIF 算法实现函数 BifTransform

输入参数：ImgData 指示保存图像数据的一维数组，ImgDataSize 指示该一维数组的元素个数，outputGaborFeature 指示输出提取后特征的数组，size 表示该特征数组的元素个数。由于前面已经叙述过关于图像数据的维数转换，这里 ImgData 不再介绍。在调用构造函数时会获得相关一维数组的大小，比如[32,32]对应着 1024 维，即为这里的 ImgDataSize。最终会将输出的数组传递给 outputGaborFeature 所指向的数组，用来保存结果，对应的用 size 来表示该数组的大小，便于后面开辟动态内存时设置可变大小的数组。

函数说明：此函数通过调用该类的各个子函数，按照年龄估计算法 BIF 的流程，实现对输入灰度图像进行特征提取的功能。依次通过 Gabor 变换，相邻尺度的求最大值，按照索引函数求得下标，并根据下标获得数据元素，求得方差后保存特征数组。最终，实现了算法中从 S1 层到 C1 层的处理，获得了最终的特征数据。

系统实现的核心就是利用 BIF 算法来提取特征，而前面编写的大量的子函数都只是为了实现该算法中的一个模块。在函数 BifTransform 中，利用论文[15]中的算法描述，如图 2.1 中所示，实现特征抽取的任务，具体的实现如下：

首先对于尺度（Scale）进行折半，以减少运算量。对相邻的尺度，通过设置 Gabor 参数，提取相应的 Gabor 特征， 正如前面算法部分所述，BIF 算法需要处

理的实际上是主要就是两步特征处理，第一步就是在 S1 层上面进行的 Gabor 变化，这里我们正是通过相邻尺度求最大值来实现；第二步就是对获得的特征进行筛选和求标准差的运算处理。这里以 Scale 为 16，输入图像为 32x32 为例说明（对于其他尺度和方向也都成立）。

经过 C1 处理后获得的是一维的 Gabor 特征，维度就是图像 width 和 height 的乘积。但是，并非所有获得的 Gabor 特征都是需要处理的，因为这样运算量非常之大，效率很低，而需要的数据只是在前面 imgblockindex 函数中得到的相关索引对应的数据。由于每一个方向（Orientation）上都要进行尺度变换，因此，在每一个尺度(从 0 到 Scale/2)循环中，提取到的 Gabor 特征都是 4096 维的，对于每一个方向上都进行了处理，但是这 1024 维只需要相关的索引项即可。为此，MATLAB 中的处理方法是将获得的 Gabor 特征变换为 1024x8 的二维数组，然后依据索引项及个数得到 dim_pos，获得新的特征的行数为 dim_pos/sub_heiwid，而每一行元素则都是根据 imgblockindex 函数获得的索引值 index 取出相应的行上面的所有元素，组成新的矩阵，并求出这个矩阵的标准差，即第二步处理中“STD”算子，这样通过循环取出索引，组成新矩阵，求它的标准差，赋值给特征数组的某行，来逐行获得特征数组的数据，最终经过多次循环，获得某个尺度上的特征矩阵，并重置为一个一维数组 GaborFeature 保存，由于每一个尺度上都会获得一个这样的矩阵和对应一维数组，因此最终结果的 Gabor 特征即为所有 GaborFeature 的累积。

然而由于涉及到平台的移植和 C++的特性，所以使用上面这种办法改写的 C++代码效率低下，原因是进行过多的矩阵变化和跨行访问数组，因此纯粹照搬 MATLAB 的数据表示和处理并不适合在 C++平台下高效运行，考虑到一维数组的随机访问性能优越，这里采用记录每次循环中相关矩阵的行列数，并存储在 TempDim，pos_num 等数组中，在访问时根据相关的起始地址动态计算索引，直接访问数据，减少了空间上开辟多维数组的开销和时间上数组转置、访问的时间开销，极大的提高了效率。关于这部分算法的改进，在下面“算法和改进部分”会有进一步详细的描述，但是就整个算法的核心思想而言，MATLAB 和 C++这两种实现方法都是相同的。

（6）数据格式转换函数 DataConverse

输入参数：inputGabor 指示输入的 Gabor 特征数组，outputGabor 表示转置后的 Gabor 特征数组。具体来说就是，经过 BifTransform 函数后获得的特征 GaborFeature 即为这里的输入参数 inputGabor，而最终要存储的是转换格式表示后的新数据，用 outputGabor 指向的动态数组来表示。

函数说明：由于经过 BIF 变换后，我们提取的特征时按照 VC 格式来排列的，也就是以行序为主序的排列方式，图像数据是按照行来逐行排列的，而在 MATLAB

中，数组是按列序为主序进行排列的，也就是说按照列逐列排列的，所以需要将生成的 inputGabor 按照 MATLAB 的格式进行重新排列，获得 MATLAB 下面的表示，从而能够将该数据和原始的 MATLAB 生成数据进行比较，获得算法性能和精确度的判断。但是由于特征数据的获取是在不同的循环体中获得的，也就是说该矩阵的维度是不确定的，那么在转换行序和列序的时候并不能够用一个统一的常量标准来转换，因此，需要记录下每次形成特征数组时的行数，以及对应的偏移量。

为了实现该转换功能，定义了两个数组 result_num 和 result_buffer，分别记录下生成一维特征数据的数组维数和相对于前一个记录的偏移量。然后利用循环计算类成员变量 TempDim 和 sizeNum 的乘积获得每一次特征的维数，放在 result_num 中，同时累积偏移量，放置在数组 result_buffer 中。经过上面这个处理，就能够成功记录相关的转换参数，为后面格式转换部分提供入口参数。

经过上面的前期工作，为了最终获得 MATLAB 形式的特征数据，还需要循环获得待处理的特征数组，利用行序和列序转换的代码，实现数据的重排，具体如下：

```
memcpy(Result,inputGabor+result_buffer[cross_scale],TempDim[cross_scale]  * sizeNum * sizeof(double));
for ( i = 0; i < sizeNum; i++)
{
    for(int j = 0; j < TempDim[cross_scale]; j++)
    {
        Result_reverse[i * TempDim[cross_scale] + j] = Result[j * sizeNum + i];
    }
}
memcpy(outputGabor+result_buffer[cross_scale],Result_reverse,TempDim[cross_scale] * sizeNum * sizeof(double));
```

每次循环中，都利用函数 memcpy 函数拷贝对应部分的特征数组，然后获得的特征数组 result 重排为列序为主序的 result_reverse 数组中，并将该数组保存到最后的输出数组 outputGabor 中，经过多次循环和下标的计算，最终实现了格式的转换。

3.3 算法优化和改进

由于人脸检测系统的核心在于特征的提取，该算法提取特征的效率和准确性，直接影响到整个系统的性能，因此，对该系统的性能要求很高。为了减少时间和

空间上的开销，对于原始的算法实现进行了多次改进，最终达到了良好的优化效果。

3.3.1 改进一：数组处理优化

数组一直是算法中经常处理的对象，因此能否对数组进行合理的操作和访问，会直接影响到算法的效率。由于在原始的处理采用的是类似于 MATLAB 的处理方式，首先对一维数组重构为二维数组，然后针对该二维数组取出特定行数上的所有列，形成一个新的二维数组，最后对该二维数组求标准差，具体就是对每一列求标准差，获得一行标准差，累积循环 dim_pos/sub_heiwid 次后重构为一维数组即得到单次循环的特征。

优化办法是：利用行列关系来获得正确的索引值，而不是使用 reshape 函数来实现矩阵的重排。

```
for(int j = 0; j < sizeNum; j++)       //一共有这么多列，每一列产生一个结果
{
        int *index = new int [sub_heiwid];   //每一列结果要先取出索引
        double *getNum = new double [sub_heiwid];   //取出对应索引的数
        for(int k = 0; k < sub_heiwid; k++) //每列结果即为由这些数的标准差
        {
            index[k] = baseindex[k] + j * dim;//根据规律直接计算出下标
            getNum[k] = GaborFeatureTmp2[index[k] - 1];
        }
        double result = stard_deviation(getNum,sub_heiwid);
        Result[cross_block * sizeNum + j] = result;//计算标准差并保存
        delete []getNum;
        delete []index;
}
```

在上面的这段代码中，通过 baseindex 数组以及 index 数组来分别存放起始地址和实际地址，从而直接根据索引值来访问一维数组元素，不进行矩阵的维数变换和二维动态数组的开辟，减少了内存空间和访问时间，精简了代码，获得了很好的性能。

为了充分的比较优化后和优化前的运行时间，由于单次运行时间较短，因此我采用循环执行多次的方式，获得累积时间，这样一则可以减少单次运行可能造成的偶然性，二则可以扩大微小误差，观察到更加准确的性能提升。这里，我采用执行该算法程序 10 次的方法，利用 MATLAB 的命令 tic，toc 来获得累积时间，然后执行 5 次观察每次的运行时间，对比结果如表 3.1 所示。

表 3.1　优化后和优化前版本执行 10 次的时间对比

序号	优化前的版本(秒)	优化一的版本(秒)	优化二的版本(秒)	提升百分比
1	34.3936	31.9785	8.0197	76.68%
2	32.5937	30.9250	7.9782	75.52%
3	32.3973	30.8179	7.9668	75.41%
4	32.8208	31.0046	8.0126	75.59%
5	32.5703	30.9955	8.0216	75.37%
平均值	**32.9551**	**31.1443**	**7.9998**	**75.73%**

通过比较优化一和优化前的运行时间发现，优化后性能提升在 5%左右，由于该算法会在系统中多次使用，甚至要处理成千上万张图片，因此累积节省的时间还是相当可观。至此，我们对于数组处理这部分优化取得了良好的效果。

3.3.2　改进二：格式转换的优化

在完成了改进一之后，由于代码中仍然存在不合理的地方，特别是对于图像数据的多次格式转换，会造成矩阵的多次赋值操作，极大的影响性能，具体来说就是：在进行 BIF 算法处理之前，代码中首先将传入的图像数据转换成 VC 格式，然后进行处理，处理过程中再次使用 MATLAB 下标的代码，最终生成的格式为 VC 格式，在进行一次数据转换为 MATLAB 形式输出。整个过程中做了多次 MATLAB 格式和 VC 格式的数据转换，因此优化部分是：直接去掉数据格式转换部分，利用 VC 的下标而不是 MATLAB 的下标，最终结果做一次格式转换为 MATLAB 格式便于比较数据精度。总计减少了两次格式转换和循环中的下标变化。比较性能如表 3.1 所示。

从表 3.1 中可以看出，此优化极大的提高了算法的性能，减少了程序的运行时间，优化后的时间减少占 75%左右，已经占据了原时间的 3/4，故可以判断知道原始算法中格式转换花费了大量的时间，而我们此步完成的优化对于整个系统性能的提升起到了很大的作用，直接使得时间降低为原来的 1/4。

3.3.3　改进三：构造函数的优化

在完成了前面两部的优化之后，算法的核心部分已经能够达到很好的性能，进一步优化的空间很小。然而，由于在一次处理中，必须要对一定规格的图像调用 imgblockindex 函数来获得图像的索引，所以放置在 BifTransform 中，可以考虑到系统处理图像的时候都是针对相同尺寸的图像来进行，比如 32x32 或者 64x64，那么只需要调用一次该函数并及时的保存下相应的索引即可。为此，本部分的优化设计到构造函数的优化，即将取图像索引的函数 imgblockindex 函数放置在构造函数中，这样只需要在创建对象的时候调用一次构造函数来初始化该索引值，然后调用 BifTransform 函数传入图像数据对不同的图像进行处理，减少

了多次求索引的重复操作，提高了程序的运行效率。为了实现这个效果，增加了类成员 pos_all,pos_num,pos_buffer，TempDim 等数组成员来保存相应的信息，并增加了构造函数的功能，用于调用 imgblockindex 函数初始化这些类成员。具体实现如下：

```
memset(pos_all1,-1,POS_ALL_INIT * sizeof(double));
imgblockindex(ImgSize,blockSize,gapSize,pos_all1);
int dim_pos = 0;
while(pos_all1[dim_pos] != -1)//有效元素都不为-1
    dim_pos++;//获得实际存放有效数据的下标
TempDim[cross_scale] = dim_pos/sub_height/sub_width;//首先获得循环次数
pos_num[cross_scale] = dim_pos; //每一个 pos_num 都标记了大小
```

通过上面的这段代码和相应的循环实现，获得必要的索引信息，便于系统下一步的调用和处理。由于这部分优化仅在系统实现后才能体现，故这里并不针对 MATLAB 平台做比较，不过可以肯定的是，在大量数据的处理过程中，采用构造函数的处理方法，能够减少大量的重复工作，对于提升系统的总体性能具有很明显的作用。

3.3.4 改进四：格式转换函数模块的分离

此部分实际上只是针对优化二进行的一个改进，在优化二中，虽然已经去掉了相关的格式转换的工作，但是最后数据生成部分，由于需要和 MATLAB 进行比较，所以还是认为的增加的一个数据格式转换的步骤，也使得整个算法增加了这一个多余的环节。实际的系统由于采用的是 VC++平台，因此处理的都是 VC 格式的数据类型，并不需要 MATLAB 格式作为比较。但是，作为算法性能测试的一个重要组成部分，格式转换可以作为一个单独的函数模块，用于测试，比较和验证，并且该部分可以作用一个函数接口供外部程序来调用，因此这里选择分离出该模块，并用另外一个函数 DataConverse 来实现。由于类中已经保留了相关数组的信息，因此直接移植相关的代码，并做循环处理即可。

3.3.5 改进五：算法扩展性优化

在所有测试和改进中，算法都是在图像尺寸为[32,32]以及尺度 Scale 为 16 的情况下进行的，然而考虑到算法的移植性和适用性，为了在更广大的样本下能够使用，需要对该算法进行改进，以便处理更大的图像尺寸和其它的尺度。在图像处理中，输入图像越大，包含的信息越多，越有利于获得更佳的性能，因此考虑到将图像尺寸扩展到[64,64]甚至更大的尺寸上，而尺度 Scale 为 16 时已经足够大，仅仅需要考虑在其它较小尺度时的兼容性即可。为此，首先修改了相关的参数，扩大了预设的数组大小，并对程序的相关部分进行修改，利用 memcpy 取代循环赋值部分，进一步提高效率；将循环开辟空间放置在循环体外部以避免多次开辟

和释放空间的开销；对图像尺寸进行判断以便针对[32,32]或者[64,64]分别设置不同的 blockSize 来获得大小合适的特征维数；根据最终计算的结果设定输出参数的维数而不必预设固定大小的数组来准确界定最终生成特征维数……

经过一系列的变化，最终算法表现出良好的健壮性和可移植性，能够在输入图像大小为[32,32]和[64,64]图像，尺度 Scale 小于 16 时获得良好的数据结果，最终比较算法产生数据和实际 MATLAB 中产生数据的误差，如图 3.1 和 3.2 所示。其中图 3.1 指示了在图像大小为 32x32 时 MATLAB 和 VC++两者所产生结果的误差，图 3.2 指示了图像大小为 64x64 时的 MATLAB 和 VC++两者产生结果的误差。

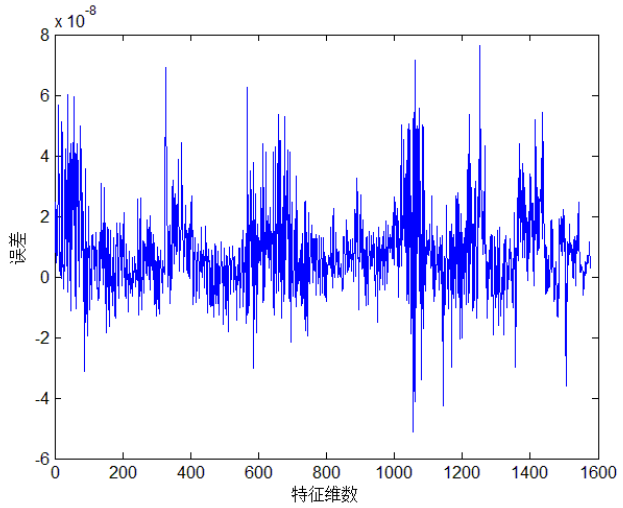

图 3.1 图像大小为 32x32 时 MATLAB 和 VC 的误差比较

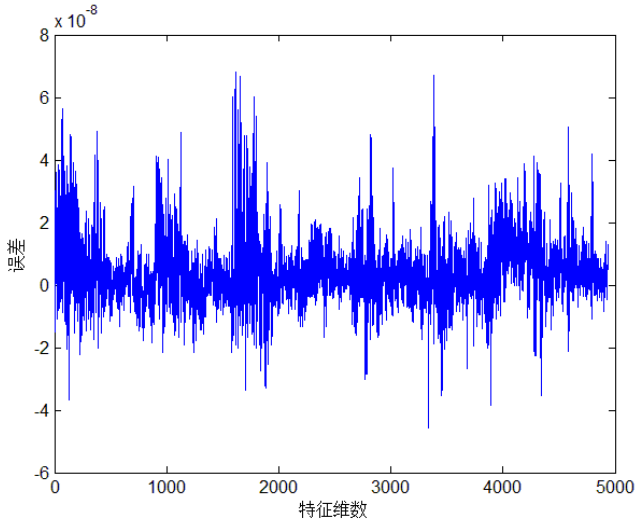

图 3.2 图像大小为 64x64 时 MATLAB 和 VC 的误差比较

4 人脸年龄估计系统的实现

4.1 系统组成

人脸年龄估计问题是一个复杂的问题，其思想是要从图像获得人脸区域中的有用信息，并根据该信息进行分类处理，获得年龄。本文提到的系统包含以下几个部分：图像采集和捕捉，人脸检测，人眼定位，特征提取，支持向量机处理，分类器分类。该系统既能够对单帧的静态图像进行年龄估计，又能够通过摄像头捕捉图像并实时估计年龄。在图 4.1 中给出了该系统的流程图。

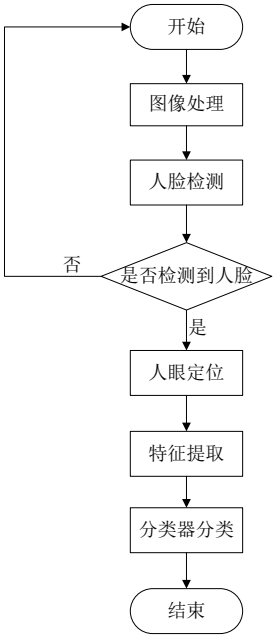

图 4.1 人脸年龄估计系统的流程图

根据处理流程可以知道，系统的大概工作流程包括如下步骤：

- 人脸检测：对输入的图像进行人脸检测，以确定图像中是否包含人脸。如果包含人脸，则给出人脸所在的区域位置。
- 人眼定位：根据已经检测得到的人脸区域进行眼睛位置搜索，得到眼睛的准确位置后，重新对人脸图像进行切割和预处理，获得校准人眼后的区域。
- 特征提取：根据前面获得的面部图像，进行图像的预处理，包括直方图均衡，归一化处理，然后将产生的结果作为输入，利用 BIF 算法处理，获得最终的面部特征数据。
- 分类器分类：将上一步获得的特征数据进行降维处理，具体就是利用之前训练好的模型参数，通过矩阵的乘积运算降低维数，达到降维的效果。然后将

降维后的特征作为参数，传入训练好的分类器中进行分类，从而获得最终估计的年龄值。

最终，采用 Visio 2007 绘制系统的 UML 状态图，如图 4.2 所示。

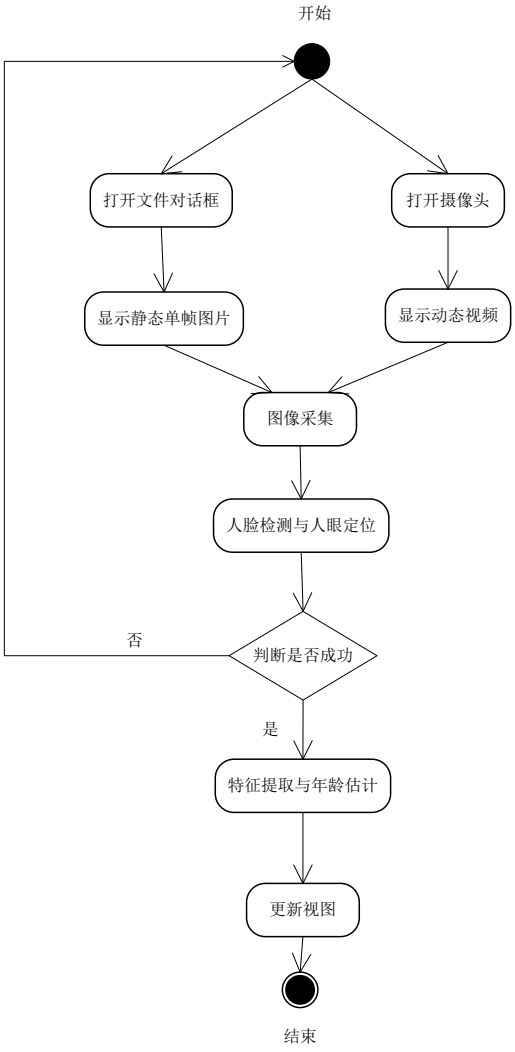

图 4.2 系统的 UML 状态活动图

4.2 系统模块介绍

本节主要对系统的主要功能模块进行介绍，尤其是算法中没有涉及到的人脸检测，人眼定位和支持向量机处理部分。

4.2.1 图像采集和人脸数据库

该原型系统首先需要从文件中读取 bmp 格式的人脸图像，或者从摄像头捕捉

的视频中截取到人脸图像，完成初期的图像采集工作，便于后续进行人脸检测等步骤的进行。同时，由于后期年龄判决阶段需要用到分类器，因此还需要选择合适的人脸数据库，通过大量的面部图像进行训练，获得相应的分类器，以便为后期使用该分类器打下基础。

该原型系统既能够对单帧静态图像进行年龄估计，又能够利用摄像头实时捕捉人脸图像进行年龄估计。为了获得性能良好的分类器，需要采集大量的已知实际年龄的人脸样本并进行训练。在本系统中，主要使用了两个年龄数据库，分别是 FG-NET 人脸数据库和手动采集的数据库。FG-NET 人脸数据库包括 1002 张不同年龄和性别的欧美人脸，自定义数据集则包括 6597 张不同年龄和性别的亚裔人脸。

在实验中，采集切割相应的图片以获得人脸区域，然后利用这些样本，通过 LIBSVM 训练即可获得分类器。实验中对这两个数据集分别进行训练。前者直接使用 FG-NET 中的 1002 张截取人脸后的图像训练，虽然数据集较小，但是便于进行算法正确性的验证，及早修改或者完善系统的代码，通过将样本图像估计的年龄和实际年龄比较来检验算法正确性。后者则筛选掉数据集中部分图像，并融合前面 FG-NET 的图样，一起进行训练，这样一则可以使得模型更适宜于亚洲人脸估计，二则可以利用大规模数据增加边缘年龄的样本数，使得幼儿和老人的估计效果更好。通过这两个分类器模型的建立，保证系统正确高效的运行。

4.2.2 人脸检测

本系统使用一个基于 Adaboost 的人脸检测器，从而较好的解决了复杂光照和姿态变化等问题。图 4.3 给出了此人脸检测模块的框架结构。

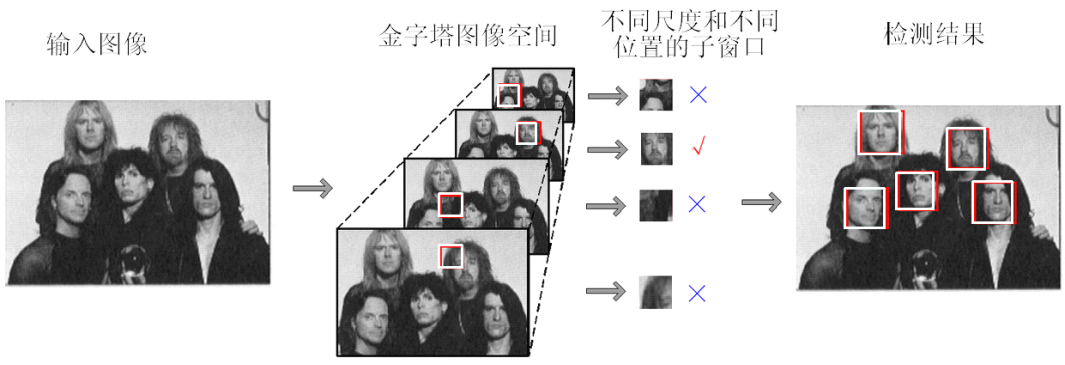

图 4.3 人脸检测算法示意图

大量的实验结果表明，通过分类器的组合可以极大地提高单个特征或分类器

的区分能力，Boosting 就是一种经常采用的组合分类器方式。而它的一个演化算法 AdaBoost 已经用于人脸检测，并能够在快速处理图像的同时达到和其他方法可比的精度。因此，本原型系统采用了 AdaBoost 方法来训练分类器，通过组合一些弱分类器来形成一个最终的强分类器。

4.2.3 人眼定位

人眼定位是绝大多数人脸分析任务中不可缺少的一项预处理过程。人眼定位的结果对于人脸图像归一化和面部关键特征点定位等都有着极其重要的意义。本系统使用的人脸定位方法为双向Cascaded AdaBoost 的眼睛定位方法，该方法的主要过程如图4.4所示。

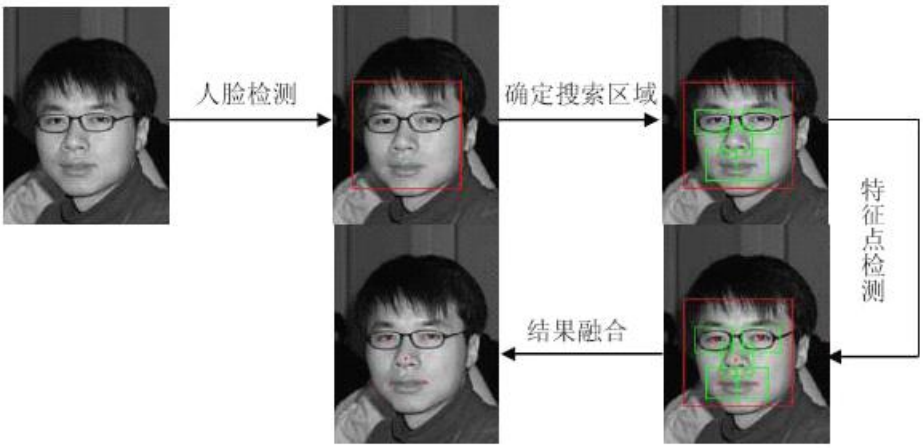

图 4.4 人眼定位示意图

双向 Cascaded AdaBoost 的眼睛定位方法首先在输入图像中通过人脸检测算法给出人脸图像的头框的位置，然后在头框的范围内确定各特征点所在的区域。通过在区域内对特征点的检测，最后通过对应的特征点的检测器的输出结果融合得到特征点的最终位置。特征点检测器的构成借鉴了 AdaBoost 算法在人脸检测上的应用，它通过在训练集上选择 Haar 特征，构成 AdaBoost 分类器来完成特征点的检测，其中正例样本是以对应特征点位置为中心的矩形框，反例样本是人脸图像中远离对应特征点区域取出的同等大小的矩形框。

4.2.4 支持向量机处理

这里分离器训练我们主要使用 LIBSVM 来实现，LIBSVM 是一个简单、易于

使用和快速有效的 SVM 模式识别与回归的软件包，它不但提供了编译好的可在 Windows 系列系统的执行文件，还提供了源代码，方便改进、修改以及在其它操作系统上应用；该软件对 SVM 所涉及的参数调节相对比较少，提供了很多的默认参数，利用这些默认参数可以解决很多问题，并提供了交互检验(Cross Validation)的功能。

LIBSVM 使用的一般步骤是：

⑴ 按照 LIBSVM 软件包所要求的格式准备数据集；

⑵ 对数据进行简单的缩放操作；

⑶ 考虑选用 RBF 核函数；

⑷ 采用交叉验证选择最佳参数 C 与 g ；

⑸ 采用最佳参数 C 与 g 对整个训练集进行训练获取支持向量机模型；

⑹ 利用获取的模型进行测试与预测。

结合以上研究内容，我设计了如下的解决问题的方案：根据系统的工作流程，先完成前三个模块的初步的函数功能设计，对人脸检测，人眼定位和特征提取的流程进行深入了解；然后完成这三个模块的具体实现，并独立测试，测试成功后，进行系统测试，最后验证系统的年龄估计的正确率。

4.3 类设计与实现

在系统实现的过程中，为了对每一个模块分别进行处理，需要使用相关的类来实现不同的功能，比如为了实现图像的处理和存储，使用 CFaceImage 类来实现图片的读取，拷贝，灰度处理，捕捉，保存；为了实现图片框的功能，使用 CFaceWnd 类来实现图片的增加，删除，清空，并响应其中左键双击，右键单击的动作。本节中，通过绘制相关的类视图，描述相关类之间的调用关系和依赖关系。图 4.5 给出该系统的工程对应的总体类关系视图，其中 CMy001Dlg 对应着该年龄估计系统工程的主类，它包含这 CFaceImage 类型的指针成员 pImage，CFaceWnd 类型的指针成员 m_wndAddFace，以及 CMyVideoCapture 类型的指针成员 m_pVideo。这三个主要的类分别控制着系统的图像处理部分、界面显示部分和视频处理部分。pImage 用于保存待处理的图像，可以调用图示的各种方法实现图像处理的相关工作，m_wndAddFace 用于保存待显示图像的相关信息和界面响应函数，m_pVideo 用于保存待处理视频的相关信息，可以调用图示的各种方法来实现视频处理工作。最终保证整个系统在各个类之间的配合下高效的运行。

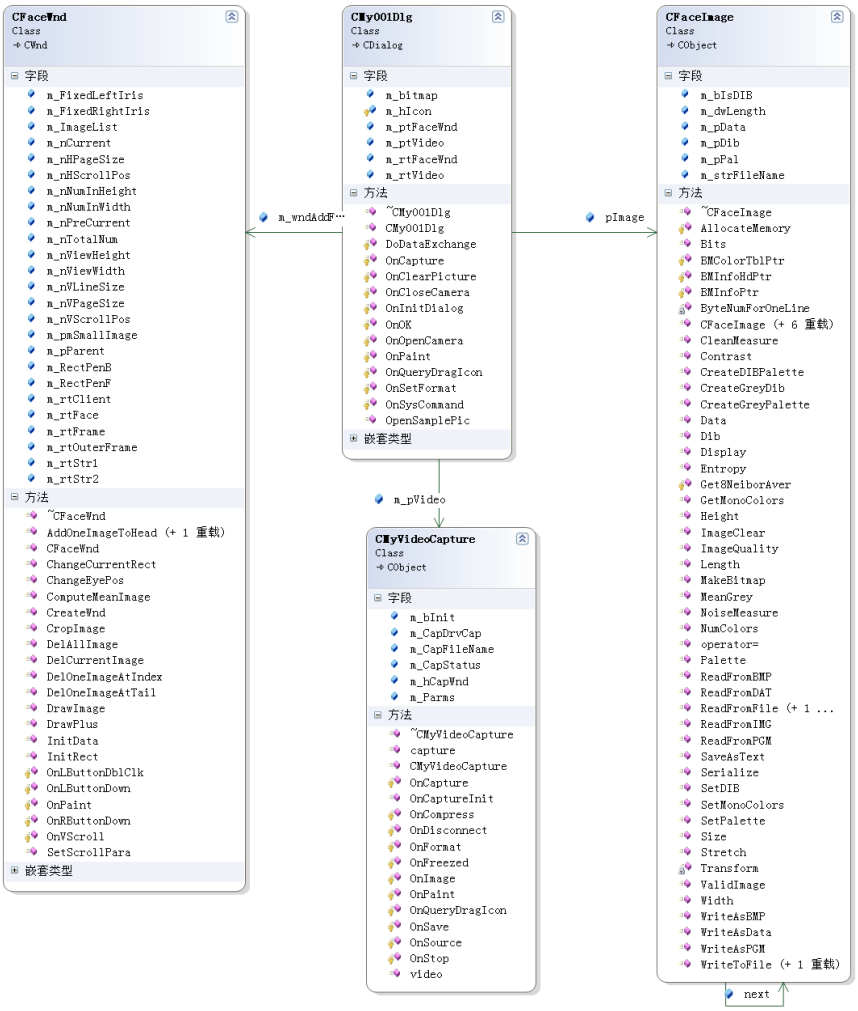

图 4.5 系统的类关系图

下面详细的介绍几个主要类的功能：

4.3.1 CAgeEstimatation

功能描述:用于实现年龄估计的类。由于对于每一张图片，都需要经过人脸检测，人眼定位，特征提取和分类器分类，所以分别将这些部分封装在该类中，并用不同的函数来实现之，实现了功能的模块化处理。而 CAgeEstimatation 类则将这些模块有效的结合起来，使整个年龄估计的流程有条不紊的进行下去。类成员及函数说明如下，相关的类关系图如图 4.6 所示。

```
class CAgeEstimatation
{
public:
    BIF * m_pBif;
    CFaceImage * pCtrImage;        //用于保存缩放后的图像
    CFaceImage * pSmallImage; //用于保存显示的图像
```

```
    CSVMPredict * m_pSVMpredict;        //SVM 训练模型
    CFeatureExtract* m_pFeaExt;         //特征提取模型
    void FeatureExtraction();       //特征提取并估计年龄
    void FaceResize(CFaceImage *pImage);        //用于将原始图像缩放保存
    CFaceImage * pImage64;          //用于保存截取后并定位好的 64x64 图像
    double EstimateAge;             //保存估计的年龄值
    BOOL FaceDetect(CFaceImage *pImage);        //人脸检测，并保存面部图像
    CAgeEstimatation();             //初始化模型参数
    virtual ~CAgeEstimatation();
};
```

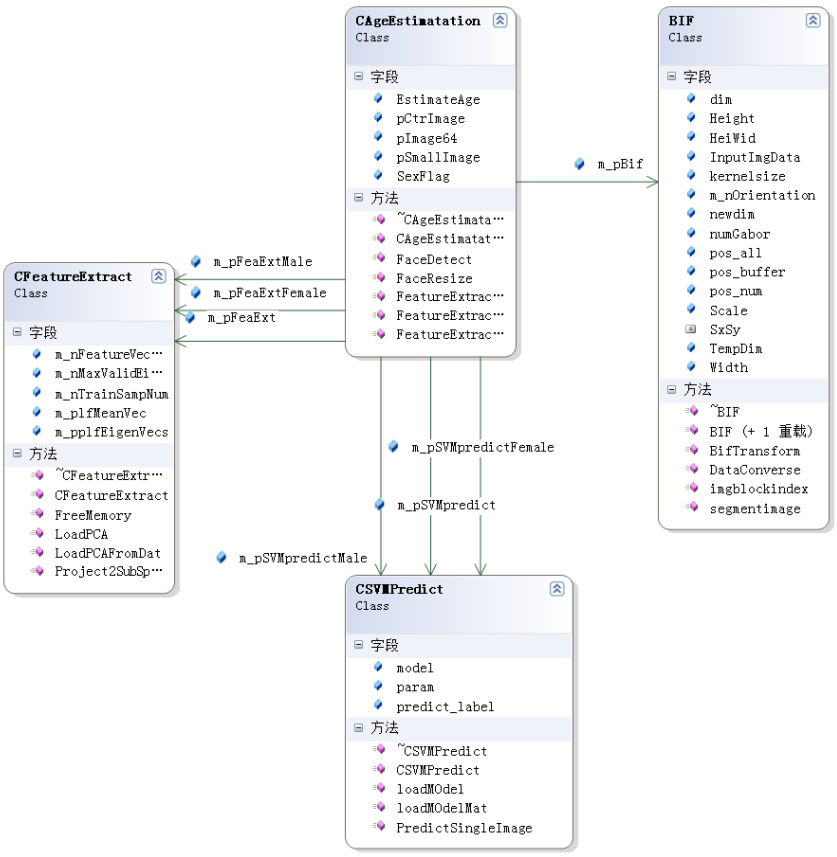

图 4.6 年龄估计类 CAgeEstimatation 的类关系图

从图 4.6 中可以看出，年龄估计类包含 CFeatureExtract 类型的指针成员 m_pFeaExt，CSVMPredict 类型的指针成员 m_pSVMpredict 和 BIF 类型的指针成员 m_pBif，它们分别实现特征模型的建立、SVM 分类器的建立和 BIF 算法的特征提取。该类首先通过 BIF 类的 BifTransform 函数提取完特征，然后利用 m_pFeaExt 读取特征文件，并利用该特征文件中的特征矩阵，调用

Project2SubSpace 进行降维处理，获得 SVM 分类器的输入特征，最后调用 m_pSVMpredict 的子函数 PredictSingleImage 来估计年龄，保存相关的估计值。

4.3.2 CMyVideoCapture

功能描述：用于实现摄像头的捕捉。在该系统中，为了实现实时捕捉视频并对捕捉的图像进行年龄估计，需要利用到视频编程的知识，为此了解了 vfw(video for windows)的编程知识，并封装了相关的函数和功能，实现了摄像头设备的检测、初始化、捕捉视频、关闭等各个功能。由于这部分功能比较独立和简单，所以不再赘述。

```
class CMyVideoCapture
{
public:
    CMyVideoCapture();
    virtual ~CMyVideoCapture();
    void capture();
     static    UINT video (LPVOID param);
    HWND m_hCapWnd;// 预示窗口
    BOOL m_bInit;// 捕捉器初始化
    CAPDRIVERCAPS m_CapDrvCap; //定义驱动器性能
    CAPSTATUS m_CapStatus; // CAPSTATUS 结构，定义捕捉窗口当前状态
     CAPTUREPARMS m_Parms;   // CAPTURE PARMS 结构，定义捕捉参数
     CString m_CapFileName; // 捕捉文件名称
    BOOL OnCaptureInit(HWND hwndParent);
protected:
     void OnPaint();
     HCURSOR OnQueryDragIcon();
     void OnFormat();              //设置摄像头格式，如 320x240 或者 640x480
     void OnSource();              //选定摄像头设备，识别设备。
     void OnCompress();      //设置视频的压缩方式
     void OnCapture();             //捕捉视频
     void OnFreezed();             //暂停捕捉视频
     void OnImage();               //将当前的视频中的图像保存
     void OnStop();            //停止摄像头工作
     void OnDisconnect();          //断开摄像头的连接
     void OnSave();            //保存摄像头视频。
};
```

4.4 系统优化与改进

在整个系统的实现过程中，为了对界面和功能都实现更多更强大的功能，也为了提高系统的健壮性和可移植性，系统经过了多次的改进和优化，不论是效率上还是功能上，都经过了从简单到复杂的过程。下面重点介绍几个优化和改进的部分：

4.4.1 提高人眼标注的精度

在第一次估计年龄时，系统会自动调用相关的人脸检测和眼睛定位来进行脸部图像的切割，但是，该算法并不能够保证识别的准确性，仍然会出现识别不准确的情况，为此需要手动进行标注。在程序中，我们通过编写类 CDlgChangeIris 来实现用鼠标选中人眼并进行标记这一功能，并可根据新的坐标来重新进行年龄估计，大大提高了系统的灵活性。同时，针对于测试时部分小图像进行人眼定位的时候不够准确，容易因为较小的位置变化而带来较大的年龄波动这个问题，我选择了将图像进行放大，这样，相对而言，人眼区域更大，更容易找到中心位置，从而获取的坐标更准确，估计更好。具体处理就是，定义放大系数：

```
#define   MAGNIFYRATIO  3 //定义在输入为 64x64 的时候图像的放大系数
#define   NORMALMAGRATIO 2   //定义普通输入图像的放大系数
```

然后在图像特征提取完毕后，对原始图像根据尺寸的按照上面的比例进行相应的放大。比如在完成 FeatureExtraction 后，编写如下代码构造新的放大图像：

```
int width = NORMALMAGRATIO * m_pAgeEst->pCtrImage->Width(); //定义新的宽和高
int height = NORMALMAGRATIO * m_pAgeEst->pCtrImage->Height();
CFaceImage *pLargeImage = new CFaceImage(CSize(width,height));//构造并放大
ResizeImage(m_pAgeEst->pCtrImage->Data(),m_pAgeEst->pCtrImage->Width(),m_pAgeEst->pCtrImage->Height(),pLargeImage->Data(),width, height);
```

同时，修改函数 CFaceWnd::ChangeEyePos()，使得每次在改变人眼位置的时候刷新并更新相关的年龄信息，调用 CAgeEstimatation 中的相关函数估计新的年龄值，实现优化后的估计。

4.4.2 减少图像处理的误差

由于图像需要经过多个步骤才能进行最终的估计，而每一个处理过程中都可能因为处理方法的不同而导致误差，最终影响估计的结果，因此，考虑从中间环节入手来进行优化。这里通过对大量的图片进行测试知道，如果图像中人脸的区域占据很少，甚至不足 1/10 的话，很容易造成人脸检测时切取的人脸图像不准确，比如包含了太多了人脸外部区域的部分，这样在最终进行估计的时候，会因

为数据的不准确而导致出现很大的误差。为此，需要考虑如何对这种图像进行处理，以便能够更精确的捕捉到合适的人脸，然后估计年龄。根据系统流程，图像首先会进行压缩存储，存储为 320x240 格式的图像，然后根据宽高比来显示在左侧的视频捕捉框中，而后续的处理时，会采用放大后的 640x480 图像进行切取人脸和估计，这样就存在一个多次处理图像数据的损失和失真问题。在原始图像缩放到 320x480 时，已经损失了部分数据，而再次放大后，会对缩小后的该图像拉伸，这样图像就会粗线锯齿状，从而导致切割的时候图像也不够清晰，影响精度。为此，接下来的任务就是考虑如何减少这些处理步骤中的误差损失。

(1) 方法一：坐标数据的重新处理

由于误差较大是由图像放大所导致的，因此为了减少这部分误差，可以考虑不使用放大后的图像进行年龄估计，而是使用原始的缩小后的 320x240 图像进行估计，为此，需要对标记获得的人眼坐标位置进行相应的变换，因为该坐标位置是针对放大后图像的坐标。通过修改 CDlgChangeIris 类的 OnLButtonDown 函数和 OnRButtonDown 函数中相关部分的坐标的取值，直接进行人脸截取，估计年龄。

```
//    SrcKeyPnts[0] = m_ptLeftIris;      //原始办法中直接使用放大的人眼坐标
//    SrcKeyPnts[1] = m_ptRightIris;
SrcKeyPnts[0] = CPoint(round(m_ptLeftIris.x/2.0),round(m_ptLeftIris.y/2.0));//坐标处理
SrcKeyPnts[1] = CPoint(round(m_ptRightIris.x/2.0)+1,round(m_ptLeftIris.y/2.0)+1);
//    CFaceImage* pGreyImage = new CFaceImage(*m_FullImage);//原始办法使用放大图像
CFaceImage* pGreyImage = new CFaceImage(*m_pAgeEst->pOrignalImage);//使用原始图像
```

最终通过对整个系统的其他部分进行处理和改写后，实现了这种优化方法的版本，并通过对图片进行测试和估计，发现不再存在图像出现锯齿状的失真现象，也比之前估计的结果更加准确，达到了一定的优化效果。

(2) 方法二：图像的重新处理

通过对整个系统的处理流程进行分析，可以发现，上面进行估计时候误差的产生，实际上是由于对图像的处理方法的不同。不论是压缩还是伸长，图像都会丢失原始的重要信息，因此在进行图像处理的时候，需要尽可能少的进行相关的变化。考虑到摄像头部分可以设置 640x480 的视频格式，而且在较大图像尺寸上丢失信息较少，所以选择对原始系统进行修改，重新设计界面和相关代码，将系统从 320x240 的平台上移植到 640x480 的平台上，这样既可以利用到摄像头采集更大的图像，便于捕捉精确的面部信息，又可以使得较小图像不用压缩，减少信

息丢失，提高估计精度。

此部分的具体实现就是在上面的基础上修改相关的图形界面参数，初始化代码，并去掉相应的放大部分，直接采用压缩图像进行年龄估计，主要涉及图像处理相关函数的改写和界面的重构，因此这里不再赘述。

4.4.3 容错处理

整个系统在实现的过程中，随着功能的日渐复杂和模块的逐渐增多，总会出现各种各样的错误，有时候甚至会出现导致程序的崩溃，因此，有必要对相关的模块和部分加入错误控制的代码，以便增加程序的健壮性，保证系统能够处理各种意外情况。在这里，错误处理主要包括两个方面：界面部分和算法部分。

由于系统采用 VC++ 6.0 编程，使用 MFC 编写图形界面，因此，不仅需要正确使用相关的 API 函数，而且要了解消息响应机制。在初始版本的系统实现中，由于没有对对话框的关闭和取消按钮进行响应，导致选择图片时如果取消或者关闭对话框会出现断点错误，为此，通过了解函数的调用过程，添加响应代码，并对于是否选中图片进行判断，使用 assert 断言来进行出错控制，减少界面部分的错误。

对于算法部分，由于整个系统的模块很多，函数调用关系复杂，因此，在初期遇到这方面的 BUG 时很难定位错误之处。但是，后来通过对函数的返回值设置操作码，显示正确和错误状态，并对调用过程中返回值进行判断，及时的输出错误信息，从而实现了健壮性的提高。举例来说，比如在初期程序经常崩溃，通过调试定位到文件读写函数和人脸检测函数经常出错，为此加入错误控制代码：

```
if(FALSE == pImage->ReadFromFile(dlg.GetPathName()))  //读取文件失败
{
    return ;
}
```

另外，在人脸检测中，由于用到了外部 DLL 库，因此如果没有检测到人脸，也需要相应的处理，在本系统中，由于精度的原因，很可能在人脸检测中没有检测到人脸，因此，使用该部分代码进行错误控制尤为重要，能够极大的减少崩溃的可能性。

4.5 系统实例（一）

本部分是利用该系统分别对静态单帧图像和实时捕捉的视频图像分别年龄估计。图 4.7 显示了系统对静态单帧图像的年龄估计结果，图 4.8 显示了系统对实时视频图像的年龄估计的结果。

从结果可以看出，本系统对于大部分图像能够估计出较为正确的年龄，达到

了较好的实验效果，体现了年龄估计算法的良好性能。

图 4.7 静态单帧图像的年龄估计结果示意图

图 4.8 实时采集图像的年龄估计结果示意图

4.6 系统实例（二）

在系统功能日渐完善之后，整个系统的程序优化部分已经很难获得进一步的提高，但是，为了获得更好的估计性能，结合论文[16]中提高的关于性别分类对年龄估计性能的巨大提升作用，我决定更改系统在年龄估计方面的流程，完成另外一个基于性别的年龄估计系统。

具体来说就是，原始的估计流程是，训练出适合男性和女性的统一分类器，

然后根据这个统一的分类器，直接用估计的特征来估计出年龄值。但是，鉴于男女衰老的速度不同，可以采用分别训练出男女分类器的方法，先估计性别，然后利用性别来分类器针对图像进行估计。首先提取到特征，然后根据该特征，利用性别分类器，估计出性别，再根据该性别，选择不同的年龄估计的分类器，重新利用特征来估计出年龄。

最终，系统原型估计的结果示意图如图 4.9 所示。

图 4.9 基于性别分类的年龄估计系统

从图 4.9 中可以看出，该系统能够对图像中的人脸进行性别估计和年龄估计，并且，在性别估计失败或者人眼标注不准确的时候，通过手动选择性别和标注人眼，实现对图像年龄的更加准确的估计。

5 总结

本文在对基于生物特征的年龄估计算法进行理论描述后，利用 MATLAB 和 VC++ 6.0 分别实现了 BIF 年龄估计算法的动态链接库的实现和年龄估计系统的构建，然后对算法部分和系统实现部分分别给出了具体的实现和改进，最后，还提出了一种新的利用性别分类的年龄估计方法，并给出了系统实现的原型。总而言之，利用生物特征算法，在经过人脸检测，人眼定位，特征提取，分类器分类等处理后，能够对人脸图像进行较为准确的年龄估计，该算法具有较好的性能。

系统目前在进行年龄估计时虽然对大部分图像能够较为准确的估计出年龄，但是，对于部分图像估计的效果并不理想。分析原因在于，图像处理和人脸检测部分的功能可以进一步提升，获得更为准确的人脸区域，目前系统对于部分图像识别效果并不理想，甚至会出现不能识别的问题，而且灯光对于识别人脸和年龄具有较大的影响。因此，如何对图像进行预处理，较少甚至去掉光照的影响，以及提高人脸检测部分的精确度，是我们下一步需要改进的地方。

另外，由于分类器的选择对最终估计的结果有很大的影响，因此如何获得较为准确的分类器，设计合理的估计流程也是一个可以改进的地方。在系统实例(二)中，我们构建了利用 BIF 算法的基于性别分类的新系统，该系统不同于原始的估计流程，但是获得了较好的实验结果，体现了在算法流程方面的提升空间。同时，注意到估计时候的年龄跨度比较大，所以下一步考虑对年龄进行分段，然后针对不同的年龄段分别训练出分类器，以减少在估计时候年龄跨度较大的问题。

机器自动人脸识别的研究已经开展了五十多年，目前的技术水平已经有了巨大的进步，也已经有了一些成功的应用。但是，面对复杂的实时监测环境以及无数不可预测的条件，各种形式的自动人脸识别的系统都显得有点力不从心，在研究的同时，许多学者又在考虑自动人脸识别技术究竟可以达到何种水平或者说其极限在哪里？这些问题在短时间内无法解决，但并不意味着随着科学技术的进步，总有一天高适应度，高正确检测率的机器自动人脸识别系统不会出现，这将是值得人脸识别的研究者认真思考和奋力追求的目标。

致谢

首先衷心地感谢我的毕业设计的指导老师马丙鹏老师。在三个月的时间里，马老师通过言传身教让我学到了许多重要的思想，养成了良好的学习习惯。马老师对学生的严格要求，让我的毕业设计能够顺利完成。我还要感谢实验室的王天江老师、庞秀梅老师等其他老师的关心和帮助，给予我毕业设计很多的建议和帮助。另外，实验室师兄和师姐也给了我许多的帮助，在此表示诚挚的感谢。最后还要感谢我的家人在大学四年里对我的支持和鼓励，没有你们我的学业不可能这么顺利的完成，谢谢你们！

参考文献

[1] H. K. Young, V. L. Niels. Age Classification from Facial Images. In: Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition. Seattle, Washington, U. S.A.,1994:762–767.

[2] J. Hayashi. Age and Gender Estimation Based on Wrinkle Texture and Color of FacialImages. In: Proceedings of the 16th International Conference. Washington, DC, U.S.A., 2002:405–408.

[3] A. Lanitis, C. Draganova, C. Christodoulou. Comparing Different Classifiers for Automatic Age Estimation. IEEE Transactions on Systems, Man, and Cybernetics-partB, 2004, 34(1):621–628

[4] M. Nakano, F.Yasukata, M. Fukumi. Age Classification from Face Images Focusingon Edge Information. In: Proceedings of The eighth International Conference on Knowledge-Based Intelligent Information and Engineering Systems. Wellington, New Zealand, 2004:898–904.

[5] S. K. Zhou, B. Georgescu, X. S. Zhou, et al. Image Based Regression Using BoostingMethod. In: Proceedings of the Tenth IEEE International Conference on Computer Vision. Beijing, China, 2005, 1:541– 548.

[6] X. Geng, Z. H. Zhou, Y. Zhang, et al. Learning from Facial Aging Patterns for Automatic Age Estimation. In: Proceedings of the fourteenth ACM International Conference on Multimedia. Santa Barbara, CA, 2006:307–316.

[7] Guodong Guo, Yun Fu, Charles R. Dyer, et al. Image-based human age estimation by manifold learning and locally adjusted robust regression. IEEE Trans. Pattern Anal. Mach. Intell., 2008, 17(7):1178-1188

[8]胡斓,夏利民.基于 Boosting RBF 神经网络的人脸年龄估计. 计算机工程,2006,32(19):119–201.

[9] 胡斓,夏利民.基于人工免疫识别系统的年龄估计.计算机工程与应用,2006,26:186–188.

[10]D. Hubel, T. Wiesel. Receptive fields, binocular interaction and function architecture in the cat’s visual cortex. Journal of Physiology, 1962,160:106–154.

[11] T. Serre, L. Wolf, T. Poggio. Object recognition with features inspired by visual cortex. In: Proceeding of IEEE Conf. on Computer Vision and Pattern Recognition . Washington, DC, U.S.A., 2005:944-1000

[12] T. Serre, L. Wolf, S. Bileschi, et al. Robust object recognition with cortex-like mechanisms. IEEE Trans. Pattern Anal. Mach. Intell., 2007,29(3):411–426.

[13] M. Riesenhuber , T. Poggio. Hierarchical models of object recognition in cortex. Nature Neuroscience, 1999,2(11):1019－1025.
[14] G. Guo, Y. Fu, C. Dyer, et al. Image-based human age estimation by manifold learning and locally adjusted robust regression. IEEE Trans. Image Proc., 2008,17(7):1178–1188.
[15] G. Guo, G. Mu, Y. Fu, and T. S. Huang. Human age estimation using bio-inspired features. In: Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition,. Miami, FL,U.S.A.,2009:112-119.
[16] G. Guo, G. Mu, Y. Fu, et al. A Study on Automatic Age Estimation using a Large Database. In: Proceedings of the twelfth IEEE International Conference on Computer Vision. Kyoto ,Japan,2009:1986-1991