



厦门大学

本科毕业论文

(科研训练、毕业设计)

题 目：基于 **ASM** 的人脸检测

姓 名：黄纯得

学 院：软件学院

系：软件工程

专 业：软件工程

年 级：2003 级

学 号：03369042

指导教师（校内）：段鸿 职称：讲师

二〇〇七年六月

基于 ASM 的人脸检测

[摘要] ASM (Active Shape Model) 主动形状模型是一种对在图片中的已知实物对象的进行识别定位的概率工具。这是当今很流行的一种基于灰度差别的对象识别算法，具有相当的通用性，可用于对各种形状相对稳定的物体进行识别，如常见的人脸识别。1995 年，Cootes et al^[1]提出 ASM，从那以后，ASM 已经发展和完善了很多年。ASM^[2]对图片中已知对象的检测是利用之前用希望在图片检测到的模型对其进行训练，然后在新给的图片中进行搜索，并与之前的训练样本作参照，找到尽可能匹配的模式。

ASM 是由一组轮廓模型集和一组可改变形状的模式组成，ASM 模式描述了一组手动标注的形状轮廓的集合（有的可以用程序标注）的典型变化，轮廓模型描述了每个形状轮廓点的灰度值的结构的统计情况，通过各个点的灰度值统计模型，我们可以在被搜索的图片上找到具有类似灰度值结构的点，通过足够准确的起始点，ASM 会利用轮廓模型（之前用来训练的数据模型）试图在新给的图像数据中寻找最佳的匹配模型，在搜索过程中，ASM 会应用全局变量限制来保证模式在一个可能的变化范围内变形以匹配新的数据中的模式。

目前有很多关于 ASM 的扩展研究，很多人在单纯的 ASM 基础上加入了一些新的算法以弥补 ASM 的一些不足，如 ASM 易受图片采集时的光照条件的影响。有人把 LBP (Local Binary Pattern)^[2]与 ASM 结合来强化 ASM 对光照变化的稳定性和健壮性。

有一种模式，叫 AAM(Active Appearance Model)^[3]，这种模式是 ASM 的直接扩展，它除了应用图片的灰度结构外，还利用了图像的颜色信息（也可以称为纹理数据）来定位欲识别的对象（比如一个人脸）。这种模式的匹配精度比较好，但与 ASM 相比，其执行一交搜索的时间要长一些。

[关键字] 主动形状模型 轮廓模型 灰度结构 AAM

Detect Face Using ASM

[Abstract] Active Shape Model (ASM) is a popular statistical tool for locating examples of known objects in images. It was first introduced by Cootes et al. ^[1] in 1995 and has been developed and improved for many years. ASM is a model-based method which makes use of a prior model of what is expected in the image. Basically, the Active Shape Model is composed of a set of profile models and a deformable shape model. The shape model describes the typical variations of an object exhibited in a set of manually annotated images and the profile models give a statistical representation of the gray-level structures around each model point. Given a sufficiently accurate starting position, the ASM search attempts to find the best match of the shape model to the data in a new image using the profile models. ASM has global constraints that allow the shape model to deform only in ways found in the training set.

There many researches try to combine the pure ASM with other approach or model to make the ASM model more robust since there are some native defeats born with the pure ASM model, i.e., the ASM model is easily effected by the illumination, some people refine the model by introducing LBP (Local Binary Pattern) ^[2] to the ASM model construction which is more robust to illumination.

A direct extension of the ASM approach has lead to the Active Appearance Model ^[3] besides shape information, the textual information, i.e. the pixel intensities across the object, is included into the model. The AAM algorithm seeks to match both the position of the model points and a representation of the texture of the object to an image.

Key Words: Active Shape Model Profile Model Gray-Level Structures AAM

目录

第一章 引言	1
第二章 ASM 算法的数学思想	2
2.1 数学背景.....	2
2.2 一个重要的数学工具 PCA (PRINCIPAL COMPONENTS ANALYSIS)	4
第三章 项目简介	6
3.1 项目意义.....	6
3.2 项目规划	6
3.2.1 数据采集工具开发	6
3.2.2 数据处理代码包开发	6
3.2.3 处理结果的预览	7
3.2.4 模型的整合	7
3.2.5 项目开发顺序	8
第四章 开发工具和 MFC 文档视图结构简介.....	9
4.1 开发工具简介	9
4.2 MFC 文档视图结构.....	9
4.2.1 MFC 简介.....	9
4.2.2 文档视图结构带来的好处.....	9
4.3 文档视图的对应关系	10
第五章 ASM 算法.....	12
5.1 建立模型	12
5.1.1 标注人脸	12
5.1.2 取点工具介绍	13
5.1.3 人脸向量的格式	14
5.1.4 对齐训练形状	15
算法一 向量训练形状对齐的算法	15
5.1.5 通过 PCA 分析建立模型	19
5.2 图像搜索.....	25
5.2.1 搜索的相关算法	25
算法二 搜索匹配人脸	25
5.2.2 搜索范围	27
算法三 寻找最佳匹配模型的迭代算法	28

5.2.3 搜索效果结论	29
第六章 程序实现	30
6.1 程序引言	30
6.2 程序模块相互依赖关系	31
6.3 程序的工具介绍	31
6.3.1 点采集工具	32
6.3.2 矩阵浏览器	32
6.3.3 线条连接设计器	33
6.3.4 模型浏览器	35
6.4 主程序主要类的说明	36
6.4.1 CSignalStudioDoc 类的说明.....	37
6.4.2 CSignalStudioView 类.....	37
6.4.3 CAmplifyRateManager 类.....	38
6.4.4 CMagicRect 类.....	39
6.4.5 CColorMenu 彩色菜单类.....	42
6.4.6 其它类	43
6.5 数学包的简要说明	43
6.6 图像处理模块的简要说明	43
结论	44
个人的心得体会	44
后继工作的展望	44
致谢语	45
参考文献	46
附录一	47
附录二	49

Content

CHAPTER 1 INTRODUCTION	1
CHAPTER 2 MATHEMATICAL BACKGROUND FOR ASM.....	2
2.1 MATHEMATICAL KNOWLEDGE	2
2.2 ONE USEFULL MATHEMATICAL TOOL PCA (PRINCIPAL COMPONENTS ANALYSIS).....	4
CHAPTER 3 CONSCISE INTRODUCTION OF THE PROJECT.....	6
3.1 SIGNIFICANCE OF THE PROJECT.....	6
3.2 PROJECT PLANNING.....	6
3.2.1 The Tool For LandMarks	6
3.2.2 The Development Of Data-Processing Package	6
3.2.3 Preveiw Of Processing Result	7
3.2.4 Module Integration.....	7
3.2.5 The Order Of Project Development	8
CHAPTER 4 INTRODUCTION TO MFC DOCUMENT-VIEW AND TOOLS.....	9
4.1 INTRODUCTION TO THE DEVELOPING TOOL	9
4.2 THE DOCUMENT AND VIEW ARCHITECTURE	9
4.2.1 Introduction To MFC.....	9
4.2.2 The Benefit Of Document And View Frame	9
4.3 4.3 THE CO-RELATIONSHIP OF VIEWS TO DOCUMENT.....	10
CHAPTER 5 ASM ALGORITHMS	12
5.1 BUILDING A SHAPE MODEL.....	12
5.1.1 Labeling The Training Set	12
5.1.2 A Consice Introduction To The Landmark Labeling Tool	13
5.1.3 The Format Of Face Vector	14
5.1.4 Aligning The Training Shapes	15
Algorithm One: Aligning The Training Shapes	15
5.1.5 Building The Molel Using PCA	19
5.2 IMAGE SEARCH	25
5.2.1 Concerning Algorithms	25
Algorithm Two: Searching A New Position	25
5.2.2 Searching Extension.....	27
Alogorithm Three: Iterative Fitting To New Points	28
5.2.3 Searching Conclusion	29
CHAPTER 6 PROJECT PROGAMMING	30

6.1 INTRODUCTION TO PROGRAMMING.....	30
6.2 CO-RELATIONSHIP AMONG MODULES.....	31
6.3INTRODUCTION TO TOOLS	31
6.3.1 Labeling Tool.....	32
6.3.2 Matrix Viewer.....	32
6.3.3 Lining Style Designer.....	33
6.3.4 Model Viewer.....	35
6.4 MAIN CLASSES OF THE MAIN PROJECT	36
6.4.1 CSignalStudioDoc Class.....	37
6.4.2 CSignalStudioView Class.....	37
6.4.3 CAmplifyRateManager Class.....	38
6.4.4 CMagicRect Class.....	39
6.4.5 CColorMenu Class.....	42
6.4.6 Other Classes.....	43
6.5 ABOUT MATHEMATIC PACKAGE.....	43
6.6 ABOUT IMAGE-PROCESSING PACKAGE.....	43
CONCLUSION	44
PERSONAL EXPERIENCE	44
FUTURE WORK	44
ACKNOWLEDGEMENT	45
REFERENCES	46
APPENDIX ONE	47
APPENDIX TWO	49

第一章 引言

在开始论文的正文之前，我们有必要明确一下这个课题的意义。ASM 作为一种基于已知模型在新的数据中寻找可能的与已知模型匹配的模型的识别算法，它不仅仅就是一种算法，还是一种建模方法。对 ASM 的学习研究不仅是为了了解 ASM 的工作原理，更重要的是学习这种模型是如何被提出来的，如何通过数学理论的应用来实现的。单纯的 ASM 在实际应用中并不十分理想，其实际效果和期望的效果存在差别较大，这就需要我们仔细地去研究是什么造成这样的差别，如何完善它等。

这篇论文是在实现 ASM 的算法后的经验总结和个人体会。算法实现基本上是在前人的理论基础之上，通过个人的学习研究和理解后一步步进行的。算法的实现虽然没有太多的创新，但在具体实现时，一些方面采用了比较灵活的实现方式。课题要达到的目的是，通过对 ASM 的学习和实践，掌握相关的建模方法和相关的数学知识的应用，同时在学习和实践过程中培养严谨调研的精神并提高动手能力，积累编程经验。

通过这个课题的实践，不仅扩充了个人的知识面、增加实践能力，同时也改变了我对现实的一些问题的认识，促进了思维方式的转变。ASM 是我大学以来，见过的最复杂，最有新意和最有挑战的一个课题。虽然算法的思想已经成熟，但是并没有现成的具体算法说明，这需要不断的学习的思考才能有更进一步的认识，才可能真正了解这个课题的意义所在。

个人认为，ASM 本身尽管并不完善，识别的效率并不是很理想。但 ASM 本身的提出，给人们指出了一条用来识别那些具有相对稳定形状的图形的新思路。这种思维方式可以应用在计算机的很多领域，比如，可以在人工智能，模式识别等领域。在科学研究的其它领域也许也有参考价值。

第二章 ASM 算法的数学思想

2.1 数学背景

ASM 的数学的基本思想源自线性代数中的解空间的数学理论，即在一个 n 维的空间中，任意一个点的位置可以由任意一组 n 维的线性无关的长度为 n 的向量线性表示，也就是可以由任意一组基础解系线性[4]表示 n 维空间中的任一点。

如下的线性运算，向量 $\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$ 可以由一个线性无关的同维数的向量组 $\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$

表示。形式如下图：

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} * b_1 + \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{pmatrix} * b_2 + \begin{pmatrix} a_{13} \\ a_{23} \\ \vdots \\ a_{n3} \end{pmatrix} * b_3 + \cdots + \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{pmatrix} * b_n$$

$$\text{即} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} * b_1 + \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{pmatrix} * b_2 + \begin{pmatrix} a_{13} \\ a_{23} \\ \vdots \\ a_{n3} \end{pmatrix} * b_3 + \cdots + \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{pmatrix} * b_n$$

ASM 人脸定位中运用了大量的代数运算，其中一个核心的思想就是：把每一张人脸都表示成 n 维空间中的一个点，也即表示成一个从原点 $(0,0,0,\dots,0)$ 出发的一个 n 维空间中的一个向量，然后从一堆的点向量中找出足够表示所有可能脸向量的一组类似的基础解系。之所以用“类似”，并不是说找出来的解就是代数中的完全的基础解系，而只是这基础解系中“主要”的一部分解，因为一般情况下，一张脸会用很多点表示，就会形成一个很高维度的向量（一般有上百个维），如果求其的完全解系，就需要很多的训练样本和大量的运算。

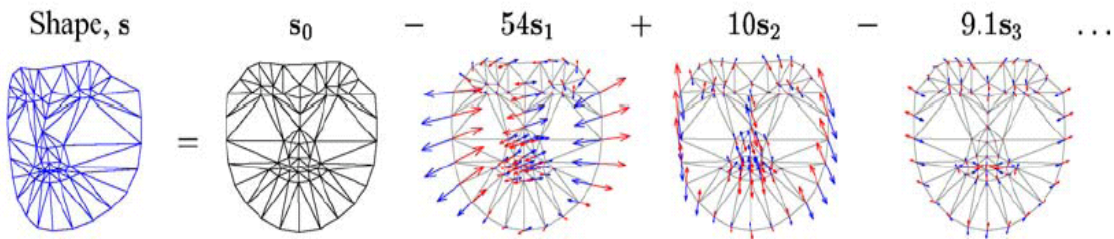
在找出基础解系后，我们就可以用它来表示所有可能情况下的人脸。只要改变一下各个解分量的系数就可以得到不同形状的人脸。当然，结果还是人脸。

在 ASM 中，其基础解系并不是直接解脸向量的解空间，而是先求出脸向量的平均值，也叫平均脸，然后求出各个训练的脸向量与平均脸的协方差矩阵，再求协方差的基础解系，这样，任何一张脸就表示成另一种形式，即：

$$S = \bar{S} + Q_s b_s$$

\bar{S} 即是平均脸向量， Q_s 为由协方差求得特征向量组成的一个类基础解系， b_s

是 Q_s 中各个向量分量的系数。之所以可以这样表示，是因为一个向量组中的任意一个向量都可以由这些向量组的平均值加上其与平均值的差。于是在人脸向量的表示也是适用的，我们可以形象一点地表示这种等式关系，如下图：



从图中我们可以看到， S_0 就是平均脸， S 即是要表示的新脸， S 可以用 S_0 加上方差矩阵的线性的成的一个差值。

2.2 一个重要的数学工具 PCA(Principal Components Analysis)

PCA[5]是一个很重要的数学分析工具。我们知道，在一个 n 维空间的基础解系一定是 n 个 n 维的线性无关的向量。在我们分析人脸向量的基础解系时，由于人脸向量的维度很高。我们经过求其协方差[6]的特征向量会有 n 个，同时可以得到 n 个特征值。由于是协方差的特征向量[6]，因此，由线性代数理论[4]我们可以知道，这 n 个 n 维向量是相互垂直的。我们可以用它们来表示任意脸向量与平均脸的差别。但 n 往往很大，如果我们直接把得到的 n 个特征值用来计算，会造成过大的计算负担，严重影响计算数度和效率，于是我们引进了 PCA 分析。

PCA 的基本方法是这样的：当我们通过计算得到协方差矩阵的 n 个特征向量和 n 个特征值后，我们把 n 个特征值从大到小排列，然后从第一个特征向量开始取，取 m 个特征向量，使得这 m 个特征向量的总和大于所有特征向量总和的一个百分值，如 98%：假如计算后并把特征值的大小从大到小排列后，我们得到 n 个特征向量：

$$\phi_1, \phi_2, \dots, \phi_n, \text{ 其对应的特征值为: } \lambda_1, \lambda_2, \dots, \lambda_n$$

我们假设

$$V_T = \sum_{i=1}^n \lambda_i,$$

我们取前面的 t 个特征值，使得：

$$\sum_{i=1}^t \lambda_i > p * V_T,$$

其中这为比例系数（一般 p 取 98%），然后把得到的 t 个特征向量组成一个新的矩

阵 $\phi\{\lambda_1, \lambda_2, \dots, \lambda_t\}$

在得到一个新和矩阵 $\phi\{\lambda_1, \lambda_2, \dots, \lambda_t\}$ 后，我们就可以用它来近似地表示所有的人脸模型与平均人脸的差别，也就是说，我们可以用平均人脸模型加上这个矩阵的一个线性组合来表示任何一张脸，通过 PCA 变换后，我们就可以减少计算的规模，和表示一张人脸所需的维度。

我们之所以用这样的少数的几个特征就可以比较准确地表示一个 n 维空间的一个向量，是因为人脸有一种比较稳定的，有统计规律的向量。其在 n 维空间的分布应该集中在一个特定的区域，并不是整个 n 维空间都有分布，我们正是利用人脸向量在 n 维空间的分布情况，来选取一个适当的坐标系。一个坐标系就是一个相互垂直的基础解系，我们选择的是协方差矩阵的特征向量作为一个坐标系，这个坐标系的一个轴是穿过所有人脸向量的组成的点雾集，即所有人脸向量在 n 维空间中的分布是集中在一块区域上，这个区域包含着所有可能的人脸向量的点，这些点看上去就像一团雾。其它的坐标轴都与那条穿过去雾的轴（不妨叫它主轴）垂直，它们是用来表示云雾与分布在主轴上的点的偏移，由于大多数的向量是集中在一个区域内，因此，在不十分严格准确的条件下，我们可以忽略一些不明显的偏移，就是这 PCA 分析法的目的，它在计算特征值的同时能得到一组对应的特征向量，特征值对应该的特征向量的大小决定其在空间中的重要程度，特征值越大，其表示的偏移就越大，就越不能忽略，因此，特征最大那个特征向量就是那条主轴。于是我们在上面的运算中，只用其中特征值最大的前 t 个向量就足够近似地表示所有人脸的可能变形情况。

第三章 项目简介

3.1 项目意义

基于 ASM 的人脸识别，是利用已有的 ASM 算法来实现的，算法完全是从底层开始，一点一点地做起，项目的最初选择到项目规划、项目实施和测试，整个过程是一个完整的调研学习过程。项目开始之前我认真阅读过相关论文和材料，并进行过项目开始前的一些必要的实验，为项目的顺利开展打下了坚实的基础，由于是采用 C++编写的，整个项目的代码比较大，又涉及到一些辅助工具的编码，整个项目的代码量超过万行，项目完整地提供了用于 ASM 数据采集，相关文件的生成，模型的演示，对比数据的查看等一系列工具。代码的编写规范，采用面向对象技术，模块化工程，并充分利用 C/C++语言的优势，写程序在大量的浮点运算时，仍能快速流畅。总体来说，通过项目的实践，让我检验了大学四年所学的知识，扩展了知识面，增强个人的耐心，培养了严谨务实的科研精神。

3.2 项目规划

项目实施之前，本人对其做了简单的需要分析，通过对项目目的的理解，制定了实施步骤，使得前面的工作可以为后面的研究工作提供扩展支持。

3.2.1 数据采集工具开发

由于涉及到采样等操作，在训练和搜索工作开始之前，必须要有训练数据等，因此，工程是以开发数据采集工具开始。

3.2.2 数据处理代码包开发

在得到数据后，需要按算法的要求对数据进行处理，以得到有用的数据。因此在数据采集工具成功开发完成后，就需要编写相关处理算法，如数据对齐算法、代数的矩阵相关运算

中的协方差矩阵计算、特征向量和特征值的计算。只有这些数据得到正确的处理后，才能确实下一步工具是否有必要展开。

3.2.3 处理结果的预览

在得到处理数据后，必要验证数据是否和预想的符合，在一大堆的数据中，我们并无法直接看出这些数据是否正确，而等到整个项目完成再对其进行验证将很可能使得中间的工作没有一个可信的基础，并易引起后继工作中验证的困难，因此，有必要在这时候对得到的数据进行验证，于是在完成数据的采集和处理后的任务就是开发模型浏览器，对所得到的数据进行观察。

3.2.4 模型的整合

在完成数据的验证后，就应该把所有开发完成的工具，算法和数据整合在一起。并通过之前定义的数据交换约定，把必要的模块整合到一个框架下，并由框架进行有效的集成。具体实现时，这一步主要是把事先生成的相关数据集成在一个多文档视图结构下的 MFC 框架。框架负责数据的后期处理、显示和演示等。由于要把必要的工具，算法等整合在框架下，因此对框架各个模块的分工还要进行仔细的设计。如文档类负责数据的存取、处理，并为视图类提供相应的数据访问接口；视图类负责对数据进行显示，提供显示方式的设定和内容的选择等。

3. 2. 5 项目开发顺序

一个项目的开发在明确所有要开发的模块后，还应该确定模型开发的相互顺序。有的模型要同时开发，有的则要较其它模型先开发。在这个项目的开发中，由于模型的相互依赖关系，采用的开发顺序是一种类似流水线的方式，即一个欲开发的模块是在其依赖的模型开发完成后才开始本模块的开发。

项目开发的顺序由下至上，如下图所示：

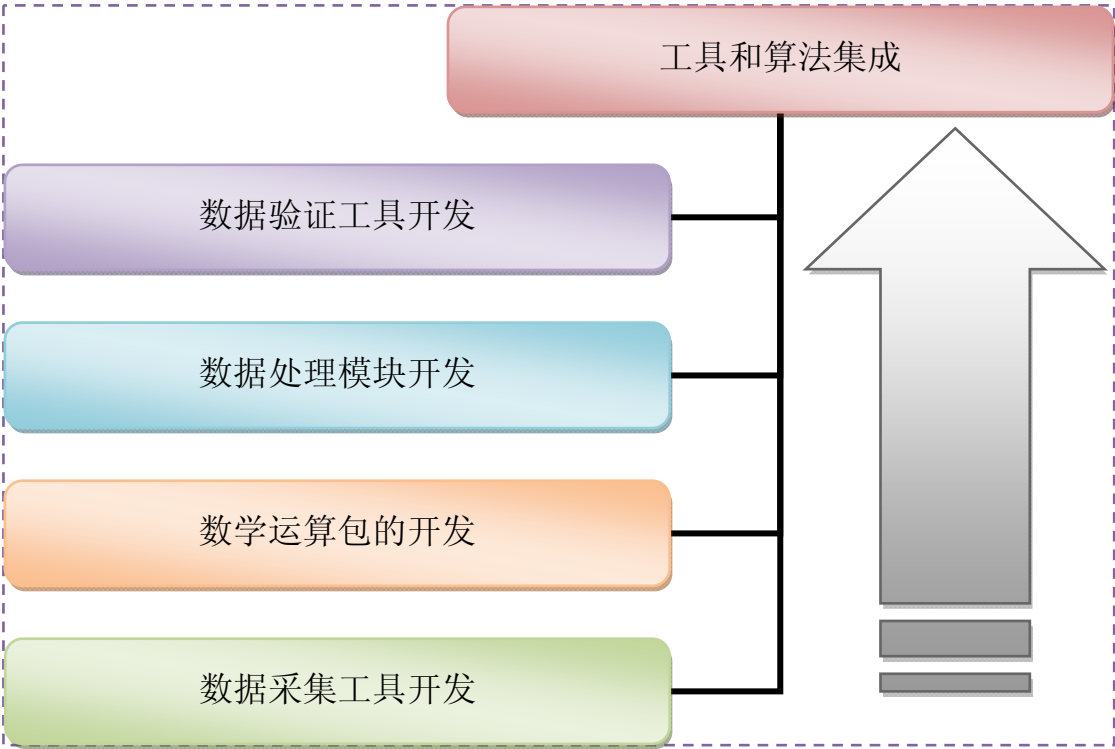


图 3-1 模块开发顺序图

从上面的顺序图中，我们可以清楚地看到项目的各个模块的开发顺序，数据采集工具最早被开发，其采集到的数据是其上面模块工作时所必要的输入数据。接下去是数学运算包的开发，这是因为上面的模块基本上都需要应用到这个模块提供的计算服务。项目开发的最后一步就是把所有这些子模块集成起来。

第四章 开发工具和 MFC 文档视图结构简介

这一章将介绍项目开发所涉及的开发工具的介绍和相关构架的概念。

4.1 开发工具简介

项目是在 Windows 平台下开发的，采用 Microsoft 的开发工具 Visual Studio.net 2005 开发，开发语言是 vc.net，开发结构利用的是 MFC 的文档视图框架。

Visual studio.net 2005 是一个集成的开发环境，其集成了微软的几种基于 .net 开发语言，如 vb, vc, vc#, vj#等，Visual studio.net 2005 是一个强大的集成开发平台，支持跨语言开发，团队开发测试等，是一个非常友好的开发平台，其调试功能十分强大，它的代码编辑器更是优秀，在用 .net 开发时，编辑器能有效也提供错误修正建议并智能纠错，使得代码的编写变得很人性化。本项目是采用 VC 中的 MFC 编写，充分发挥 MFC 传统的灵活高效优势，并且 MFC 是可以扩展到 .net 的，又可以充分发挥 .net 技术的新优势。VC 是 .net 下一种允许使用非安全代码和托管代码混合编程的语言，这也是 VC 的优势。

4.2 MFC 文档视图结构

4.2.1 MFC 简介

MFC 是 vc 下的一种架构，它是一种基础性平台，是一个模型。通过这个平台、这个模型，我们在上面进一步修饰，可以得到无穷无尽的新事物。MFC 的文档视图结构是 VC 中结构最复杂，内容最丰富的一种框架。目前绝大部分 windows 平台下运行下的大软件基本上也是采用这种结构，文档视图框架久经考验，是一种相当成熟的框架，其允许多模板，多文档和多视图，并提供了模板，文档和视图的数据交换接口，使用很方便。

4.2.2 文档视图结构带来的好处

a. 首先是将数据操作和数据显示、用户界面分离开。这是一种“分而治之”的思想，这种思想使得模块划分更加合理，模块独立性更强，同时也简化了数据操作和数据显示、用户界面工作。文档只负责数据管理，不涉及用户界面；视图只负责数据输出与用户界面的交互，可以不考虑应用程序的数据是如何组织的，甚至当文档中的数据结构发生变化时也不必改动视图的代码。

b. MFC 在文档/视图结构上提供了许多标准操作界面，包括新建文件、打开文件、保存文件、打印等，减轻了用户的工作量。用户不必再书写这些重复的代码，从而可以把更多的精力放到完成应用程序特定功能的代码上：主要是从数据源中读取数据和显示。

c. 支持打印预览和电子邮件发送功能。用户无需编写代码或只需要编写很少的代码，就可以为应用程序提供打印预览功能。同样的功能如果需要自己写的话，需要数千行的代码。另外，MFC 支持在文档视图结构中以电子邮件形式直接发送当前文档的功能，当然本地要有支持 MAPI（微软电子邮件接口）的应用程序，如 Microsoft Exchange。可以这样理解：MFC 已经把微软开发人员的智慧和技术溶入到了你自己的应用程序中。

4.3 文档视图的对应关系

一个文档可以对应多个视图，一个视图可以负责一个文档里的部分数据的显示，如下图所示：

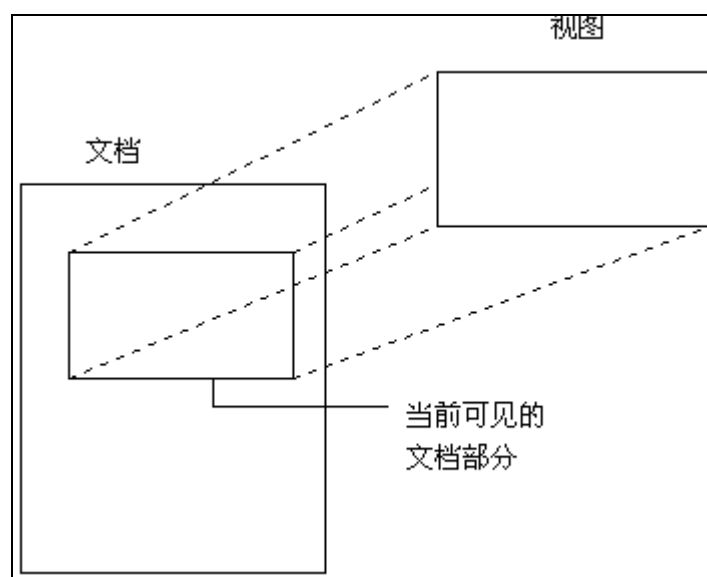


图 4-1 视图与文档的数据映射图

一个文档对应的视图可以同步更新数据的显示，文档负责协调视图对数据更新显示的工作，当用户修改了文档数据，文档就可以向与之关联的所有视图发送更新消息，通知各个视图更新显示其负责的数据。

视图与文档的数据的对应关系如下图所示：

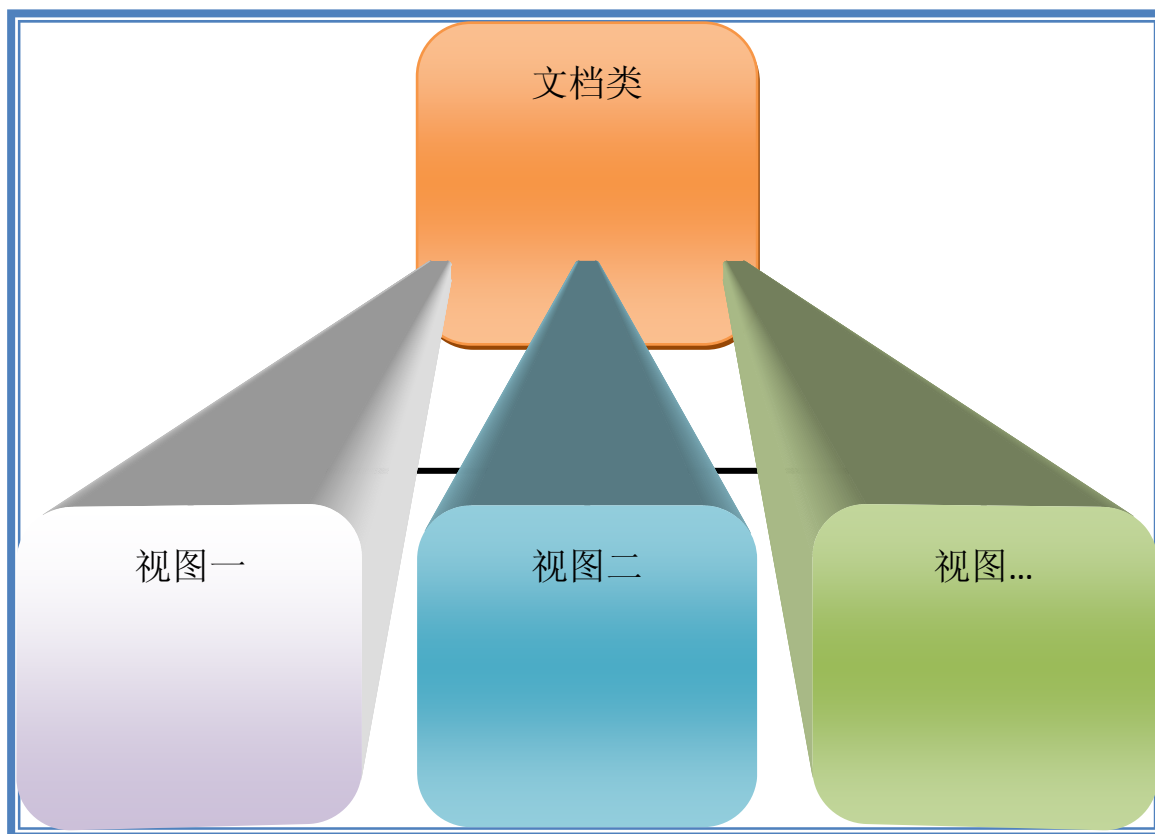


图 4-2 视图与文档的对应关系

上图表示不同视图可以负责文档中的不同数据，各种不同数据可以单独在一个视图显示，而这种对应关系使得数据的显示变得十分容易。

第五章 ASM 算法

本章将介绍 ASM 算法的实现过程，并会在介绍的过程中，介绍相关的工具和算法

5.1 建立模型

不同人的脸会由于不同表情，光照情况和人的姿势而有所变化，为了能利用 ASM 来定位人脸，就必需建立一种描述人脸典型变化的模型，这样的模型需要对人脸的变化进行概率统计计算，如计算平均模型和方差等。要得到这样的统计数据，就需要有一些人脸样本，这些人脸样本还应该包含尽可能的脸形变化，包括抬头，低头，左右旋转及不同的动作和表情。只有这样建立起来的模型才能比较准确地反应人脸的可能变化。

5.1.1 标注人脸

当我们采集到所需的人脸数据后，我们需要对其进行处理。人脸可以用 n 个点 (Landmark) 来表示， n 可以取不同的维度 d ，在人脸图片中 $d = 2$ 。此外，需要保证在每一张训练图片中所取得的点集能可靠有效地描述一张人脸，点的数量要足够以描述人脸的整体轮廓。

根据算法要求，我们在二维图像中，沿着人脸的轮廓边缘取点，取点方式如下图所示：

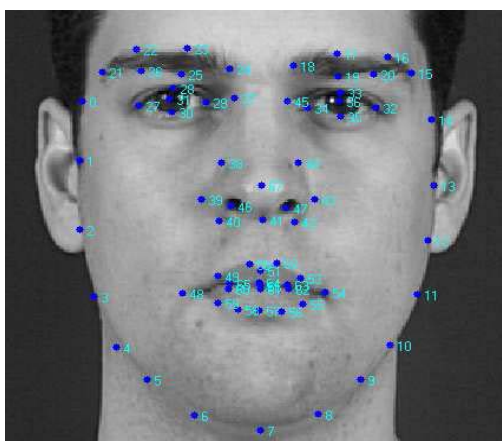


图 5-1 标注后的人脸

从图中我们看出，除了少数灰度边缘取的点间距比较大之外，在人脸的同一个部分，人脸外围轮廓，眉毛，眼睛和嘴巴的取点基本上是等间距的。

5.1.2 取点工具介绍

数据采集是整个项目实施的第一步，因此我们必需编码来实现一个取点的工具。这个工具能够把一组图片一一打开显示出来，并提供图形界面，让用户用鼠标手动地在图片上描点。所有的图片都用同一个坐标系统，工具必需能够设定所要取得的点的个数，并提示用户保存结果，还应该提供对点的编辑删除等操作。

下面简单介绍在项目中集成的取点工具，其工作界面如下图所示：

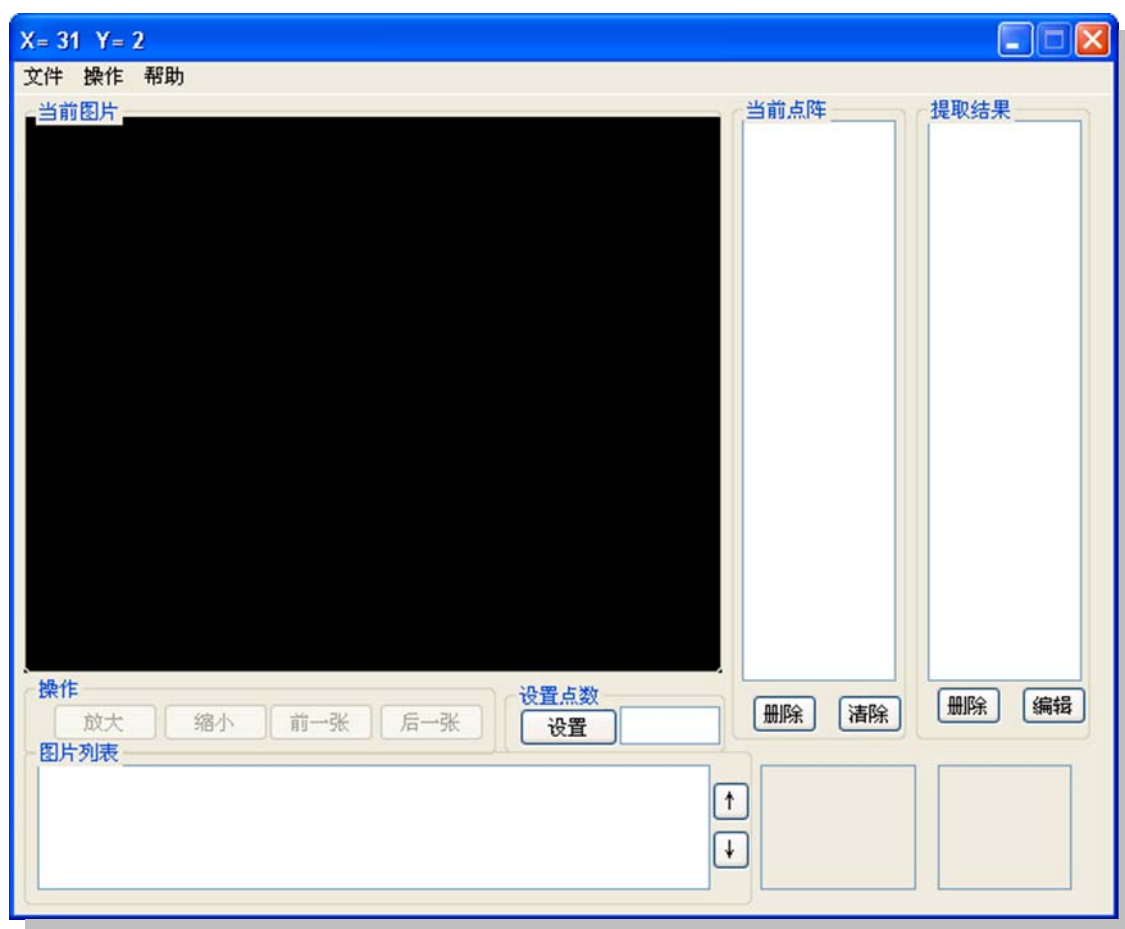


图 5-2 轮廓取点工具

从用户界面我们可以看到，黑色区域是用来显示图片及所得的取点的区域，我们可以从文件菜单中的“打开图片组”来打开训练样本图片，然后对图片一一取点。程序会在你取到足够点时提醒你保存点，工具提供多种文件格式，还能生成用于主框架打开的工程文件，工程文件把整个工具的所取得的重要数据保存成一个文件，包括所用图片，各张图片的左上角坐标，及对应该的点数据等，确保在主框架中处理时保持与取点时的一致性。其生成的文件还能再次用其打开，然后编辑，修改等。工具用友好、简洁、美观的用户界面。其还提供了

纠错功能等，当你打开的工程的点与图片不对应时，可以通过在图片列表调整图片的顺序来保证工程文件的图片与点数据的对应关系保持正确，然后再次保存，就能生成合格的工程文件，在主框架中打开后就能正确开始后继的工作。更多的细节可以参见工具在使用手册，工具开始工作后的界面效果如下图所示：

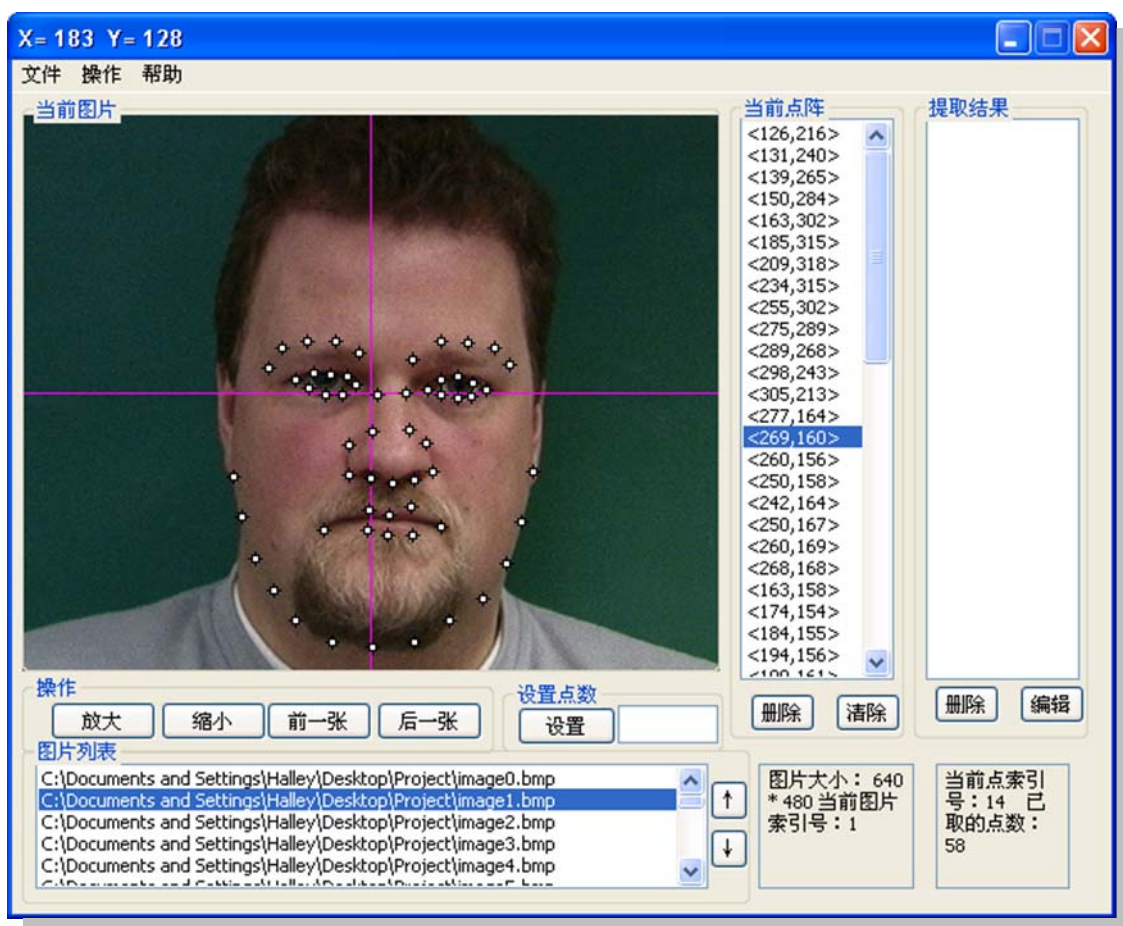


图 5-3 轮廓取点工具工作图

从图中我们可以直观看到这个工具还能显示对齐线，此外，你可以从操作菜单中选择要进行的操作，如修改点、选择点等。在选择点时，工具会显示点的索引号等信息，可以方便你找到相关的点，而不用自己去确定点的编号。

5.1.3 人脸向量的格式

利用上面介绍的取点工具，从每一张训练图片中我们都可以得到一组点集，我们假设共有 N 张图片，每张图片取 n 个点，由于是二维的图片，因此每个点的坐标可以表示成 (x_i, y_i) 。我们把每张图片得到的点的点坐标组成这样一个向量

$(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)^T$ ，向量的长度为 $2 * n$ ，向量中

(x_i, y_i) 为第 i 个点的 x 坐标和 y 坐标，我们把这样的向量叫做一个人脸向量，或人脸模型，我们后继的工作就是对这 N 个人脸模型进行相关处理。

5.1.4 对齐训练形状

我们得到一些人脸样本并把它们组成向量，我们可以把这些向量看成是在 $2 * n$ 维空间中的一个点，每个点到原点就组成一个从原点出发的维度为 $2 * n$ 矢量。但这些矢量长度不一，我们希望一个人脸模型应该是与模型的尺度，方向和位置无关的。因此，我们有必要对这些人脸向量进行一些规范化，我们把它们统一平移和缩放为重心在原点的单位向量。为此，我们必需把所有这些向量对齐到一个统一的参考向量，一般取所有向量的平均向量。我们利用 Procrustes[1] 算法来完成这些任务，这种算法对齐所有向量，使得所有向量

与平均向量的平方差达到最小，即 $(\sum |x_i - \bar{x}|^2)$ 最小。

算法一 向量训练形状对齐的算法

1. 移动所有人脸向量，使其重心在原点 $(0, 0)$
 2. 选择一个向量作为初始的平均模型（一般选择第一个人脸向量）然后将其缩放成单位向量，即 $|\bar{x}|=1$
 3. 把第一个平均估计作为默认参考帧，记为 \bar{x}_0
 4. 把所有的模型与当前的平均模型对齐
 5. 重新计算平均模型
 6. 把得到的新的平均模型与 \bar{x}_0 对齐，并重新对其变换，保证 $|\bar{x}|=1$
 7. 如果 \bar{x} 变化不是很大就停止，否则回到第 4 步
-

算法的第一步除去了模型因在不同的位置而影响对齐的结果。当所有的模型的重心都在原点后，第四步只要进行缩放和旋转就可以了，这一步是把所有模型与平均模型对齐。我们

假设一个变换 $T_{s,\theta}(x)$ 把 x 放大 s 倍，并旋转 θ 角度。至于如何对齐两个模型，见附录一。

我们假设 x_1 和 x_2 是两个重心在原点，通过选择适当的放大率 s 和旋转角 θ ，使 $|T_{s,\theta}(x_1) - x_2|^2$ 达到最小，我们就可以说这个变换把 x_1 对齐到了 x_2 。一般经过几次的迭代就可以达到很好的对齐效果，平均模型的求法很简单，只要把所有向量相加，然后除以向

量的数量就可以，即：

$$\bar{x} = \frac{1}{N} \sum_{i=1}^n x_i$$

把向量单位化用公式：

$$\bar{x} = \frac{1}{\sqrt{\sum_{i=1}^n (x_i^2 + y_i^2)}} \bar{x},$$

即把向量除以它的模长后即是单位向量。

下面两图片是对齐前后所有图像采集来的点的分布情况：

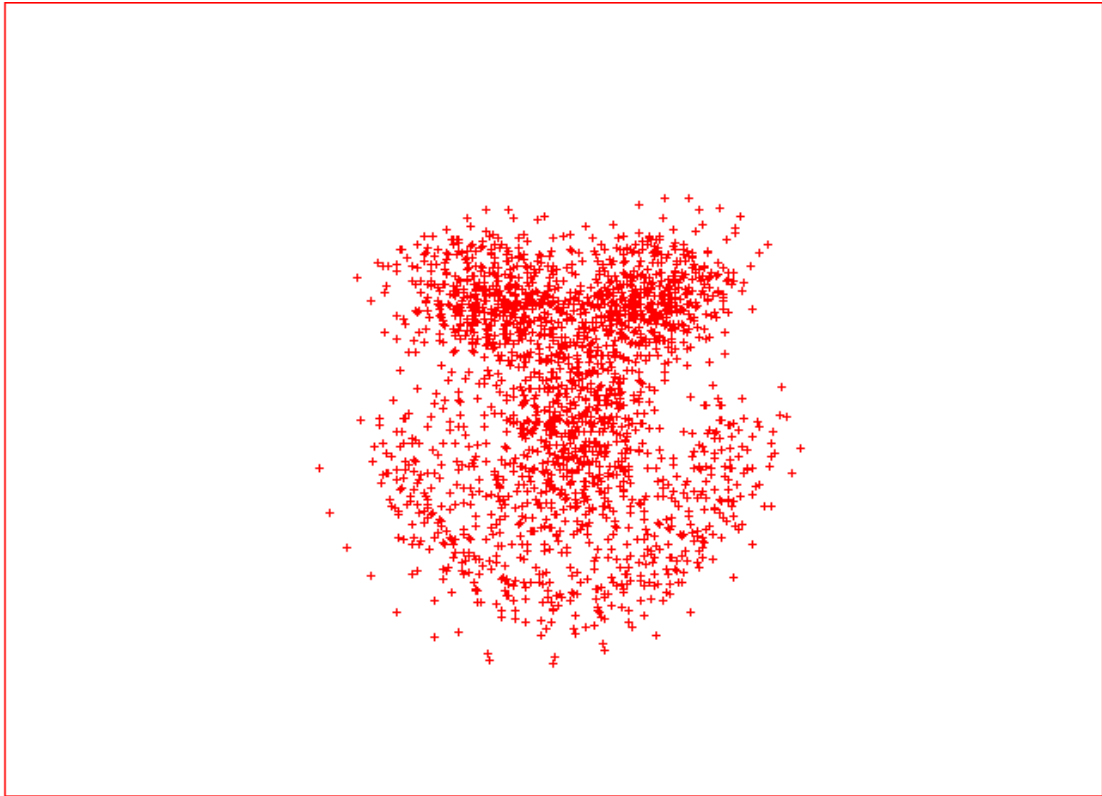


图 5-4 对齐前的模型点分布情况



图 5-5 对齐后的模型点分布情况

补充说明, 一种代数的投影变换

在把所有模型与平均对齐时, 我们引入了非线性关系, 即由变换矩阵引入的可能的旋转和缩放。为了改善这些向量的线性关系, 我们对起进行另一种变换, 就是把所有的向量投影到平均模型的切线空间。这是一种不改变向量方向的缩放, 这种变换是在算法一到了第四步后进行的。我们把所有的向量都乘以 $1 / (x_i \cdot \bar{x})$, 其中 x_i 为第 i 个向量。通过证明可以知道, 这个变换是把所有脸向量在 $2 * n$ 空间对应的点都移到了和平均向量垂直的平面上。通过坐标变换, 我们在用某个基础解系来表示这些点集, 若基础解系的几个分量和这个平面平行, 我们用来表示这些点所需的基础分量数就可以减少, 也就是增加这些向量的线性系数。如下图所示:

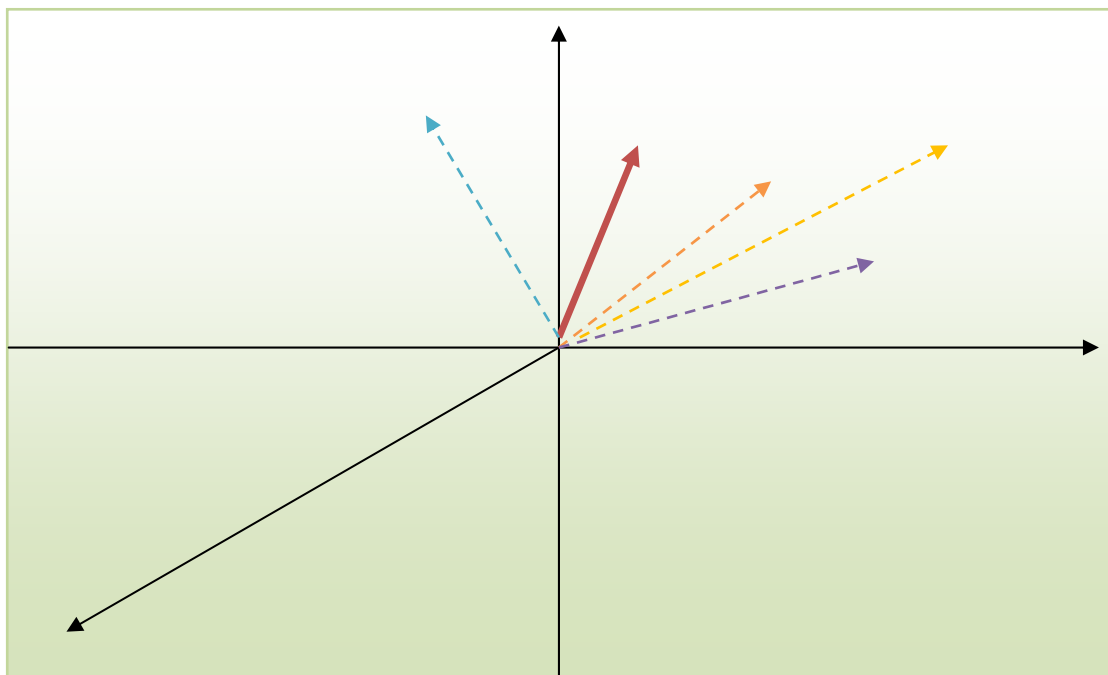


图 5-6 空间中未经过切平面投影变换的一组向量

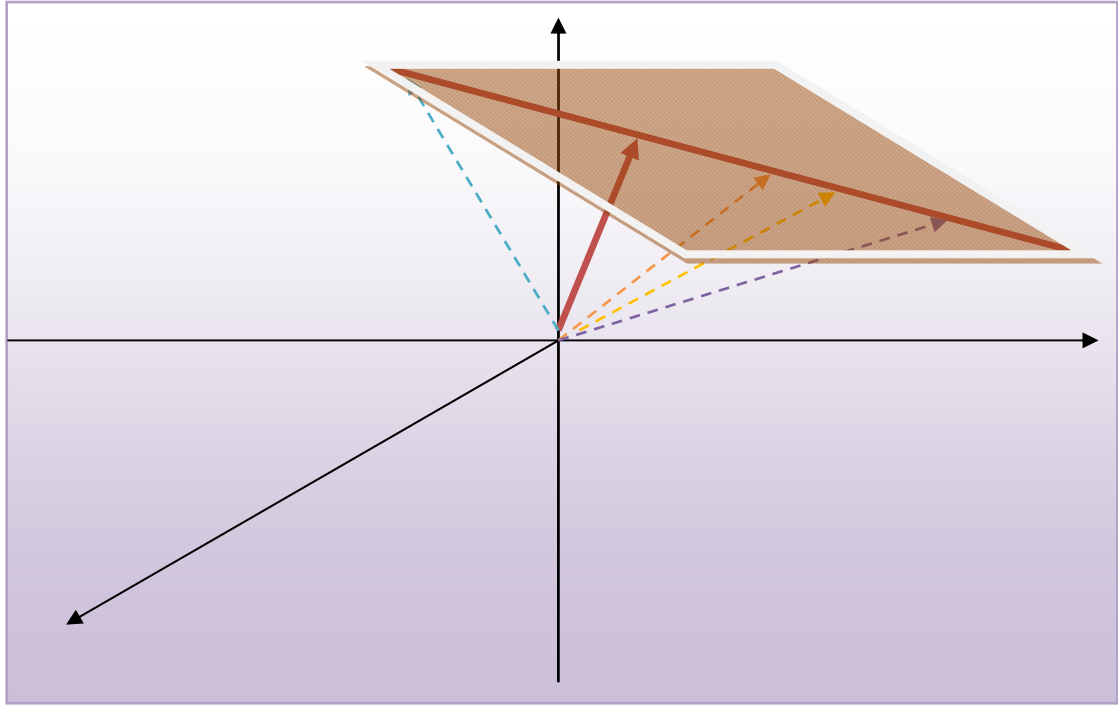


图 5-7 通过投影到平均向量的切线空间后的向量

上两个图中，实线表示的向量就是平均向量，而虚线表示其它与之对齐的向量。从上面两个图的对比我们可以看到，在变化前，所有的向量对应的点并不都在同一平面上，而通过上述的投影变换后，它们都移到同一个平面上。这个平面与平均向量是垂直的，所以也叫它切线空间。这样就能增加它们之间的线性相关程度。线性程度的增加有助于后继工作中减少运算量。

5.1.5 通过 PCA 分析建立模型

之前的工作已经得到了平均模型，而由概率统计和高等代数的知识我们可以知道，任何一个人脸模型都可以表示成平均模型加上一个偏差。因此，我们下面的工作就是找到模型能反映模型之间差别的数学模型。因为人脸向量是高维的向量，数学中能反映高维向量差别情况的模型是协方差矩阵，于是我们下面就计算这些训练样本的协方差矩阵 S ：

1.求协方差矩阵

$$S = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T$$

(其中 N 为训练图片的数量)

2. 求特征值和特征向量

利用数学模型我们可以求得 S 的特征值和对应的特征向量，并按特征值的大小，从大到小排列特征向量：

$$\phi_1, \phi_2, \dots, \phi_n$$

其对应的特征值为：

$$\lambda_1, \lambda_2, \dots, \lambda_n, \text{ 其中 } \lambda_i > \lambda_{i+1}。$$

3. 利用 PCA 的思想和相关算法，我们得到一个变换矩阵 $\Phi\{\phi_1, \phi_2, \dots, \phi_t\}$

4. 利用前面几步得到的结果，我们就可以用得到的矩阵和一组参数 b 加上平均模型来表示大致表示任一人脸向量 x ，即：

$$x \approx \bar{x} + \Phi b \quad (*)$$

其中 b 是一个 t 维的向量 ($t < nd$, d 在此为 2, n 为一个图片中所取的点的数量)，反过来，如果我们有一个新的脸向量 x ，我们就可以通过这个式子来得到其对应的参数 b ：

$$b = \Phi^T (x - \bar{x}) \quad (**)$$

从这个表达式我们也可以看到，任一个新的向量都可以通过改变参数了，并代入(*)就可以近似表示这个向量。这里要说明的一点是，向量 $b = (b_1, b_2, \dots, b_t)$ 中的每一个分量都具有相对的独立性，即其变化不受其它分量变化的影响。这样一来，任何一个变量都有可能在一个相当大的范围内变化，这样就不能保证通过(*)匹配得到的新的 x 是我们想要的人脸形状。因此，有必要对这些参数的变化范围进行约束，常用的方法是把 b_i 的范围限制在 $(-3\sqrt{\lambda_i}, +3\sqrt{\lambda_i})$ 之间，其中 λ_i 为 Φ 第 i 个分量，当 b_i 小于这个范围的下限时，就让它等于 $-3\sqrt{\lambda_i}$ ，而当其超过上限时，就让它等于 $+3\sqrt{\lambda_i}$ 。通过这样的变化限制，我们就可以

保证得到的新模型是一种可能的人脸模型。下面演示由项目集成的一个模型浏览工具，它能把上述变换结果直观地显示出来。

在项目主程序中，从工具菜单中打开模型浏览器，就可以浏览已经处理过的数据所能生成的新的模型，模型浏览工具的启动界面如下图所示：

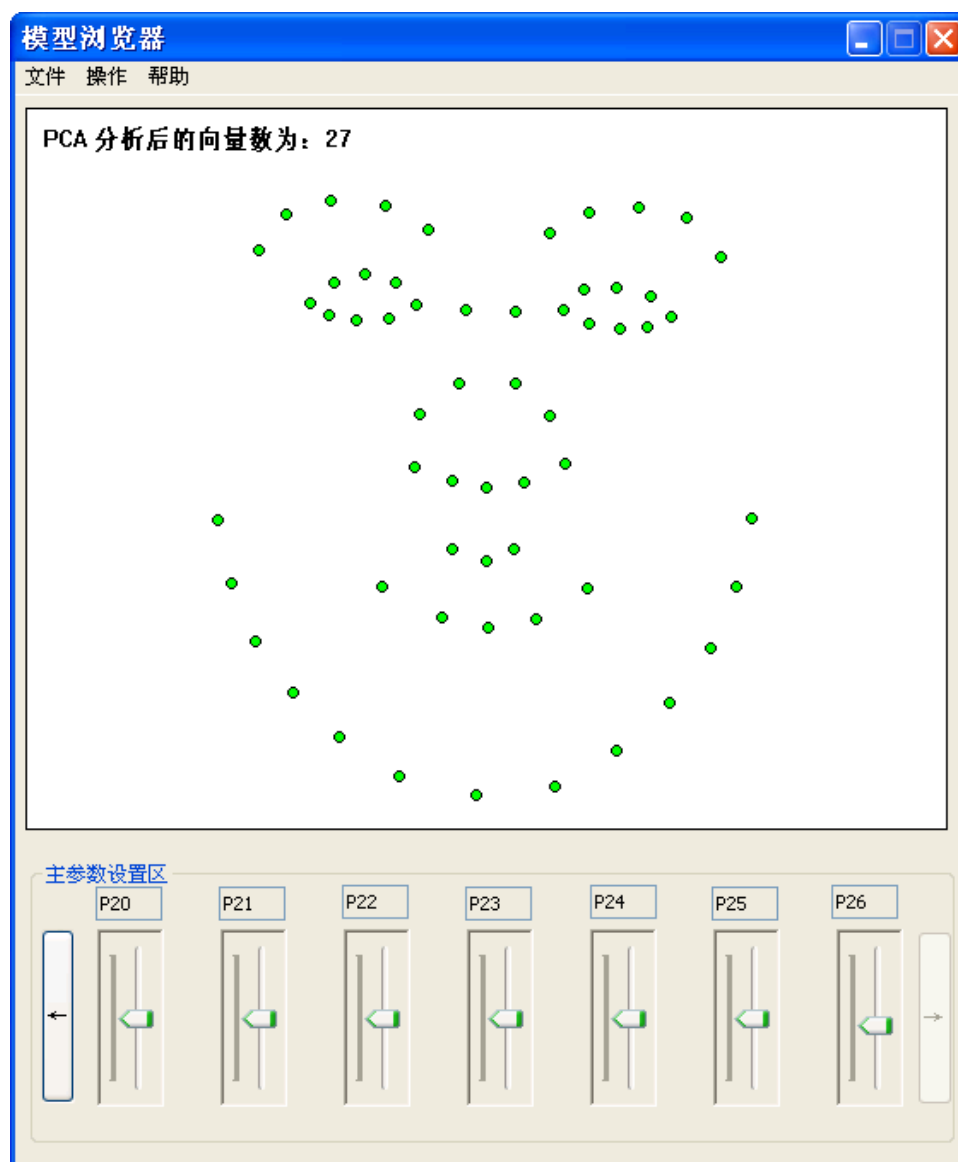
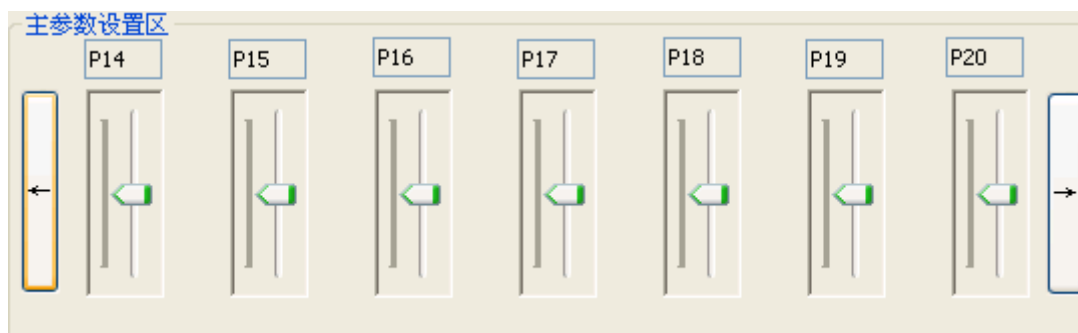


图 5-8 模型浏览器

从图中我们可以看到，这是浏览器把平均模型映射到显示区域后的初始效果。工具下面的一排滑杆控件是用来改变对应 b 的分量参数 b_i 的大小。在这工具中，为了编程的方便，每个滑杆上方都有标注其当前对应的参数名。如当前第一个滑杆对应第 20 个参数，即 b_{19} 。又由于我们通过 PCA 得到的主要的特征向量个数会比较多，如当前得到是数量是 27 个，我们不可能用 27 根不同的滑杆分别与这 27 个向量的参数对应。为此，工具采用了滑杆共享的方法，

从上面的界面我们可以看到，在这排滑杆的两侧，各有一个方向按钮，通过按动它们，我们可以选择当前要改变的 7 个参数。下图给出了另一个可改变的参数的范围：



5-9 参数控制滑杆组

从上图可以看到，现在会受改变的参数范围是从第 15 个参数到第 21 个参数，其它范围外的参数都为零，更详细的使用说明参见使用文档。

为了让我们清楚地感受这是张人脸轮廓，我们还要导入一种由另一个工具生成的文件。这种文件定义这些人脸模型中的点是如何用线段相连的，我们暂时不去考虑这个文件如何生成的，这里直接给出加载这种文件后浏览器的显示区的变化。项目给出了两个显示模型，一种标准型，另一种具有 3D 效果的 3D 模型，两种不同模型加载后的效果如下面的两个图所示：

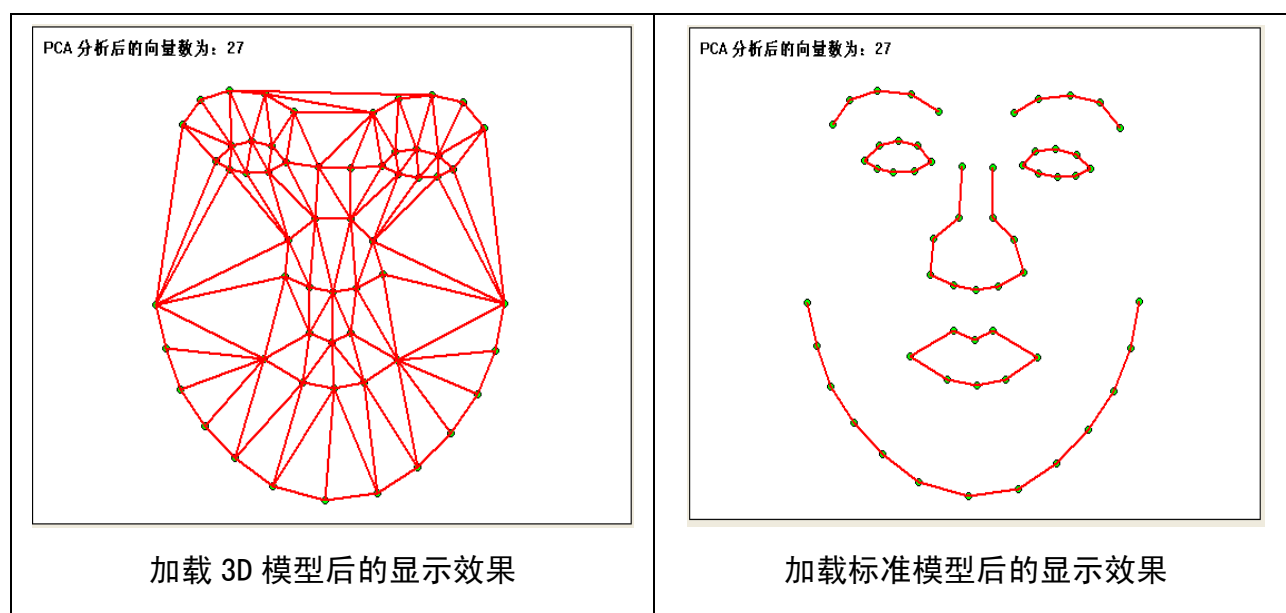
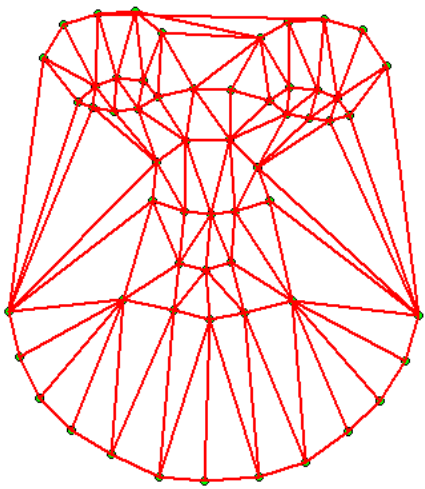
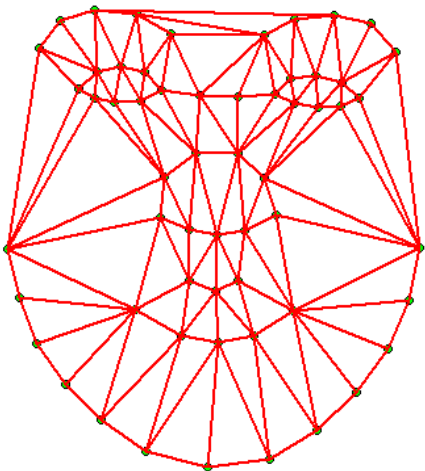
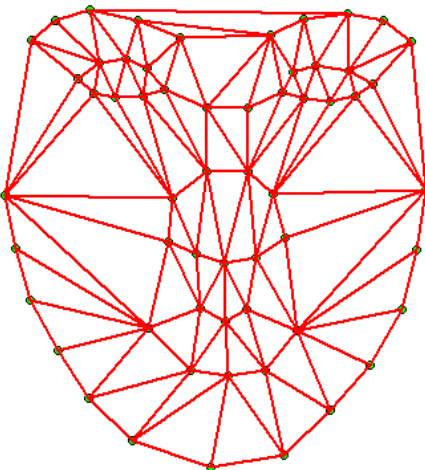
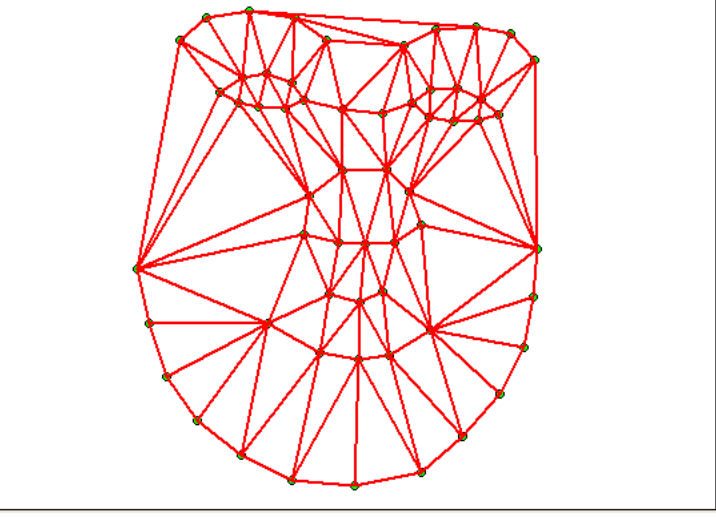
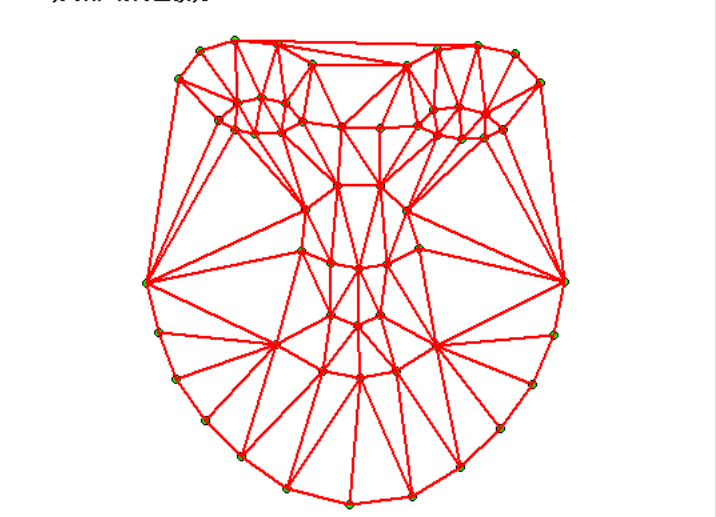
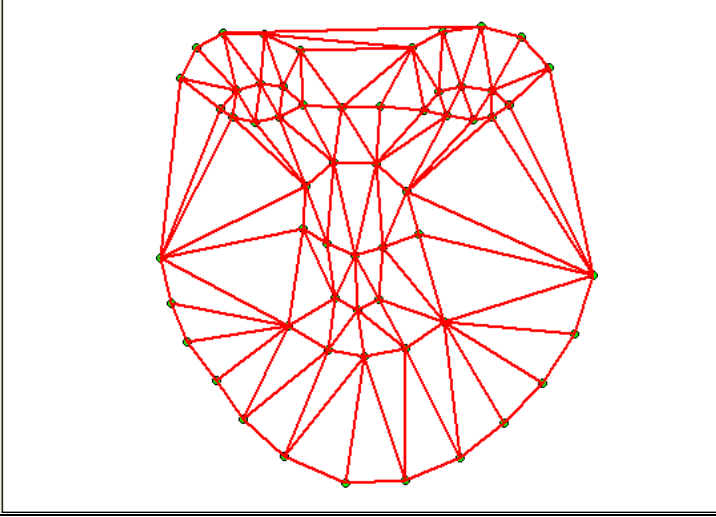


图 5-10 加载不同的连线文件的显示效果

我们下面采用 3D 显示模型来演示相关参数改变对 x 的影响，我们每次只改变一个参数，其它参数置零，并以表格的形式给出结果。

	第一个参数对比，参数 P26	
$-3\sqrt{\lambda_i}$	<p>PCA 分析后的向量数为：27</p> 	这个参数的改变使得人脸模型向上抬头
0	<p>PCA 分析后的向量数为：27</p> 	平均模型
$+3\sqrt{\lambda_i}$	<p>PCA 分析后的向量数为：27</p> 	低头

	第一个参数对比，参数 P23	
$-3\sqrt{\lambda_i}$	<p>PCA 分析后的向量数为：27</p> 	这个参数的改变使得人脸模型向右旋转
0	<p>PCA 分析后的向量数为：27</p> 	平均模型
$+3\sqrt{\lambda_i}$	<p>PCA 分析后的向量数为：27</p> 	头向左旋转

动态的变化请参见程序的演示。

5.2 图像搜索

这一节，将会介绍如何在一张新的图片中搜索出人脸并利用已有的模型与之匹配，并会给出相关算法，和工程相关工具。

5.2.1 搜索的相关算法

当我们有一张并没有标注的人脸相片时，我们需要利用 ASM 自动地利用训练后的模型来匹配这张人脸。要利用 ASM 来搜索和匹配一张人脸要事先定位这人脸的大致位置，这样的位置我们叫初始位置，这个位置的准确程度往往会影响搜索的结果。如果没有确定一个相对准确的初始位置，很可能会搜索不到人脸，就无法完成匹配。因此我们有必要在匹配前手动或利用人脸定位算法来找到相关的位置。

算法二 搜索匹配人脸

- 1.在已知的人脸的大致位置加载平均模型，缩放平均模型，使得加载的模型大致能包含新的人脸。
 - 2.在加载的模型的每一个点周围搜索最适合的点，当所有的点都找到一个新位置后，我们得到新和一个模型 x' 。
 - 3.更新参数 (X_t, Y_t, s, θ, b) 得到一个可能的最佳模型，这个模型与 x' 尽可能相合，再利用等式(*)得到一个新的可能模型。
 - 4.一直重复直到匹配成功。
-

上述的算法告诉我们如何去匹配一个人脸，但在第 2 步如何找到最佳的点，这并没有一个最佳的算法，如何确定一个点是最佳位置，要在什么尺度的范围内搜索，这些都需要进一步地讨论和反复的实验。

ASM 的创始人 Cootes[1]提出了一种搜索的方案，他的思想是：从我们训练的样本中学习各个点的灰度分布梯度模型，即在我们前一步建立模型，同时也建立所取的 n 个点的灰度梯度模型。这个模型的建立将在下面介绍，点的灰度梯度模型就是在一个点附近的一定

惊讶范围内的灰度分布情况，我们在所有的训练的图片都已经手工标注后，就可以建立这样的模型。

首先，我们把每张图片所标注的点用线段相连，就得到一个轮廓，然后在每一个点取一长度为 n_p 的一段小线段：线段的中心在这个点上，这条小线段的方向一般为角平分线，或切线的垂线，在这里，为了实现的简单的简便，程序的实现是采用角平分线，在没有角平分线的点，直接取其相关线段的垂线，如下图所示：

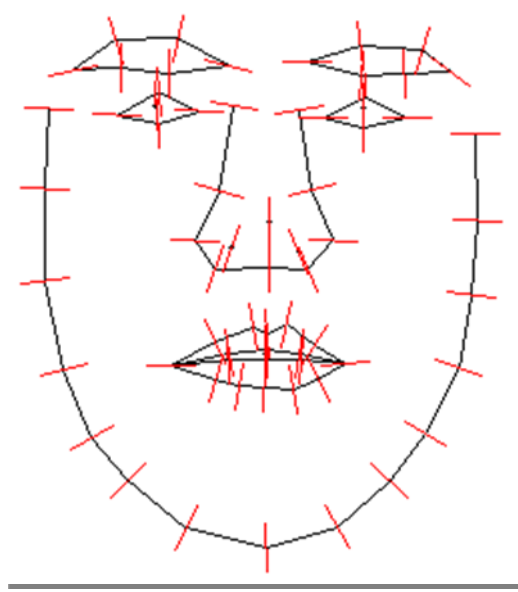


图 5-11 训练样本的提取方向

通过提取每张图片的数据，我们就可以对这些数据进行适当的处理，因为每一个点都能得到长度为 n_p 的数据，我们可以把这个数据看成维度为 n_p 的向量，并叫它为点模型向量（profile）。由于我们得到的是绝对灰度值，为了减小光照情况对数据的影响，我们采用相对灰度值来保存得到的数据：

$$g_{ijk} = I_j(y_{i(k+1)}) - I_j(y_{i(k-1)})$$

其中 g_{ijk} 表示第 j 张图片的第 i 个提取线段上的第 k 个点对应的结果， I_j 表示第 j 张图片， y_{ik} 表示第 i 条提取线段的第 k 个点，下一步就是把所得到的数据进行规范化，用下面的方式对其处理：

$$g'_{ij} = \frac{g_{ij}}{\sum_{k=1}^{n_p} |g_{ijk}|},$$

即把每个图片的每个点对应的数据都用这个等式进行转换。

下一步就是计算平均模型，平均模型是从所有图片的数据中进行处理后得到的，每一个点都有一个平均模型。第 i 个点的平均模型用下面的式子计算得到：

$$\bar{g}_i = \frac{1}{N} \sum_{j=1}^N g'_{ij},$$

即各张图片编号相同的点的数据分别相加后求平均。

和之前建立人脸模型的方式相似，这里我们在建立点的模型时，也计算每个点的协方差矩阵，并对其求逆。

$$S_{g_i} = \frac{1}{N-1} \sum_{j=1}^N (g'_{ij} - \bar{g}_i)(g'_{ij} - \bar{g}_i)^T$$

这个我们就为每一个点建立一个灰度模型。

在搜索过程中，给出一个新的点灰度向量 g_s ，这个点的合适因子用下面的函数来给出：

$$f(g_s) = (g_s - \bar{g}) S_g^{-1} (g_s - \bar{g})^T \quad (***)$$

函数值越小的点就越可能是最佳的新位置。

5.2.2 搜索范围

在实际的搜索中，我们在加载平均模型后，在平均模型的每一个点上，和在训练时提取的方式一样在，提取一段长度为 L ($L > n_p$) 的数据，并对其进行相同的变换处理。一

个点在寻找一个最佳新位置时，将其对应的平均模型沿这长为 L 的数据段进行比较，共有 $(L - n_p + 1)$ 种可能情况，利用等式 (***) 进行因子计算，因子最小的那个点就是最佳点，其位置在当前平均匹配向量的中心。匹配过程可以用主程序的匹配对照接口来显示匹配情况，下面的图示给出了几个点对应的数据匹配情况，并用直方图显示数据的大小尺度：

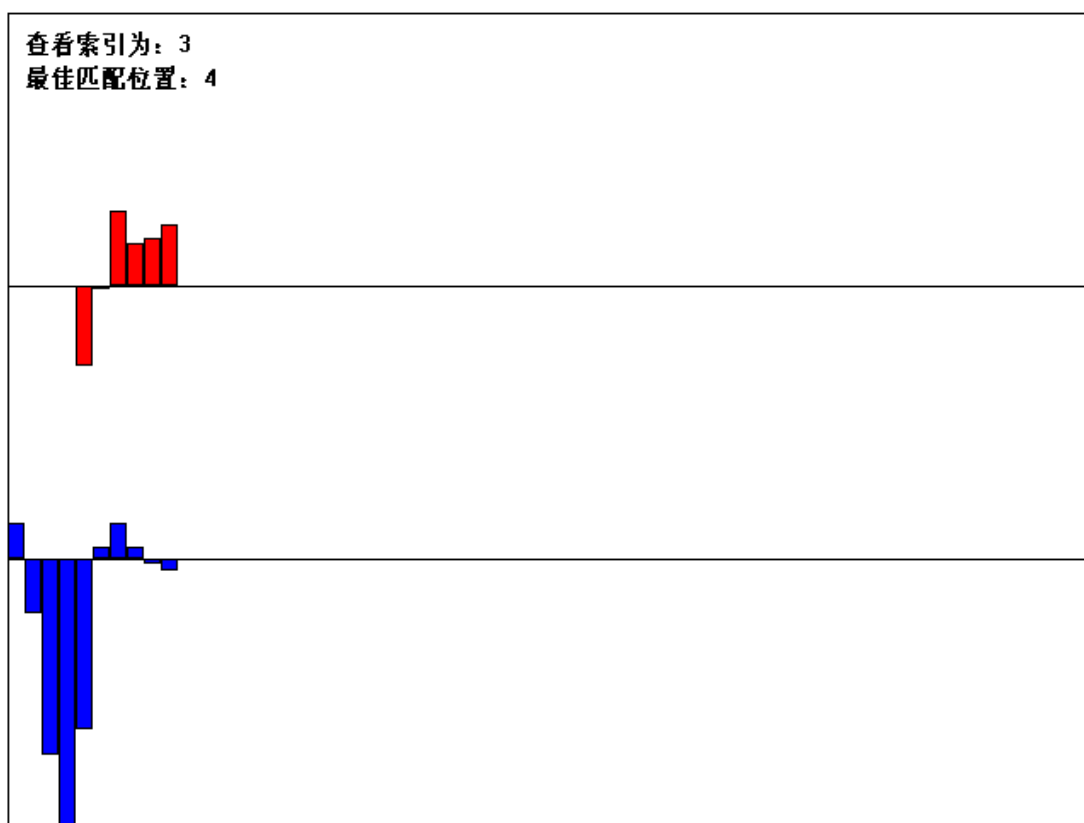


图 5-12 搜索数据平均数据的对比图

上图显示是一个搜索实例在一次提取对比数据时，索引为 3 的点的搜索过程，其匹配结果是在索引为 4 的点找到最佳匹配，训练时的长度为 5，搜索长度为 9，由于等式 (***) 是非线性运算，所以并不能直接从图看出其为什么在那个位置是最佳位置。从这个结果，我们就可计算索引为 3 的点应该移动到的新位置。当所有点都完成一次新位置的查找时，我们得到一个新模型 Y ，之后还必需通过之前建立的模型计算一个与新模型最匹配的模型等。下面的算法是用来寻找一个最可能的模型来与新得到模型 Y 匹配：

算法三 寻找最佳匹配模型的迭代算法

1. 把参数 b 初始化为零。
-

2. 利用等式 $x = \bar{x} + \Phi b$ 生成一个模型
3. 利用附录一的算法找到参数 (X_t, Y_t, s, θ) 使得 x 映射到 Y
4. 利用这些参数，把 Y 反映射到 y ，即 $y = T_{X_t, Y_t, s, \theta}^{-1}(Y)$
5. 把 y 投影到平均模型的切线空间，即把 y 乘以 $1/y \cdot \bar{x}$
6. 更新参数 b 来与 y 匹配， $b = \Phi^T(x - \bar{x})$
7. 对 b 的变化进行约束。
8. 当 b 变化不是很大时，即已经匹配，否则返回第 2 步

有了这个算法，我们只要每次提取搜索样本后，调用这个算法，生成给新模型匹配模型，然后就可以在新模型的基础上再次提取搜索样本，再次搜索和匹配直到生成的匹配模型变化不是很大，就表明算法已经稳定匹配。到此，人脸的搜索算法和思想基本已经介绍完成。

5.2.3 搜索效果结论

利用 ASM 算法实现的程序在搜索人脸时，受到几个因素的影响比较严重，一是初始位置，另一个是点模型的维度和搜索时的长度，这些因素可能导致搜索到的结果无法接受，所以原始的 ASM 的匹配并不是很好，但其思想是对的，而且通过改进是完全可能的。如在其基础上利用纹理信息，可以改善搜索结果，这就是 AAM (Active Appearance Mode), 也可以改变其搜索匹配的方式，如采用区域匹配而不采用线段匹配的方式，加入 LBP 等等。当然，比较直接的方法就是增加模型的维度，也可以增加训练的数量，这样可以带来更好的稳定性和搜索效果，在这 ASM 基础上进行改进是方法是很多的，有待进一步的研究。（搜索实例请参见附录二）

第六章 程序实现

本章主要介绍程序实现的有关细节，更多的细节请参见源代码。

6.1 程序引言

程序采用 MFC 的多文档，在 Visual Studio .net 2005 开发实现的，可以同时打开多个文件和工程。程序的用户界面简洁，操作方便，提供了菜单栏和工具条快捷操作，并支持鼠标操作，程序还应用到了动态链接库技术对程序进行了扩展。程序包括三个工程，在开发集成环境中如下图所示：

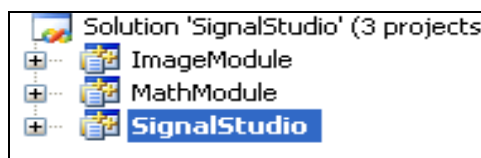


图 6-1 项目的工程视图

项目的主要工程是 Signal Studio 工程，这个工程实现了项目实现所需的工具，而且是其它模块相互关联集成的主程序，程序运行的主界面如下图所示：

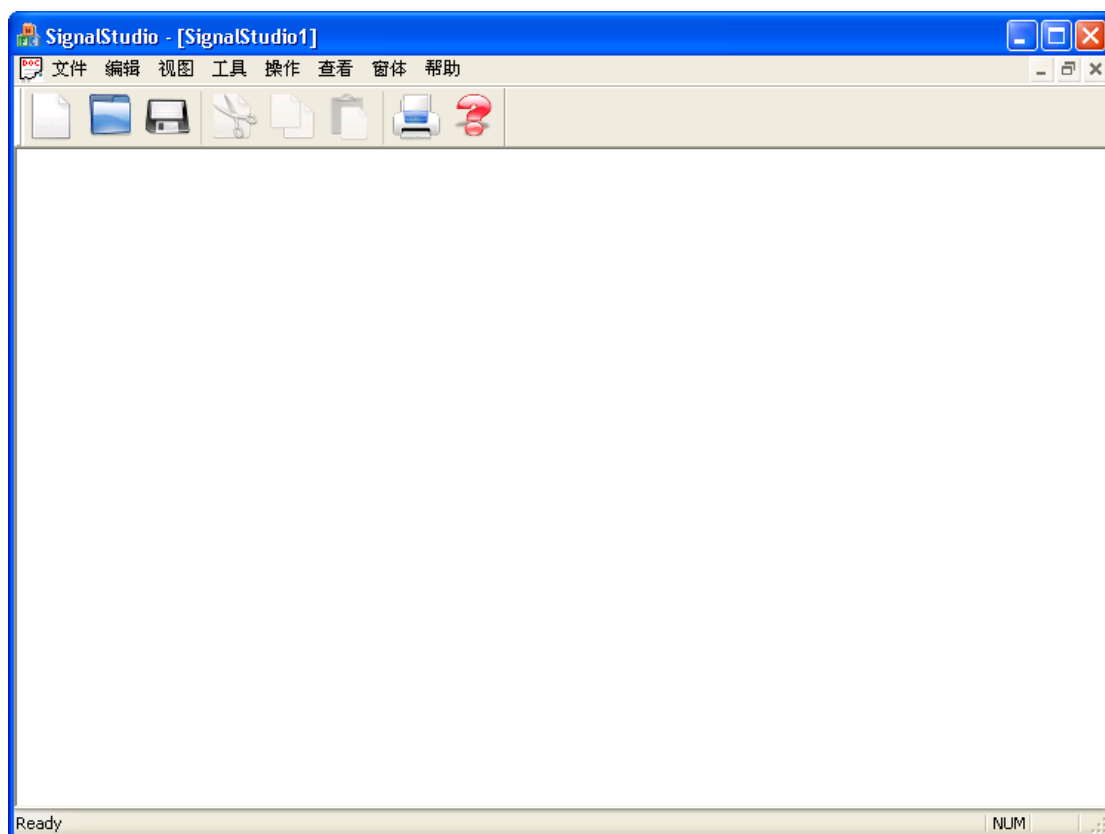
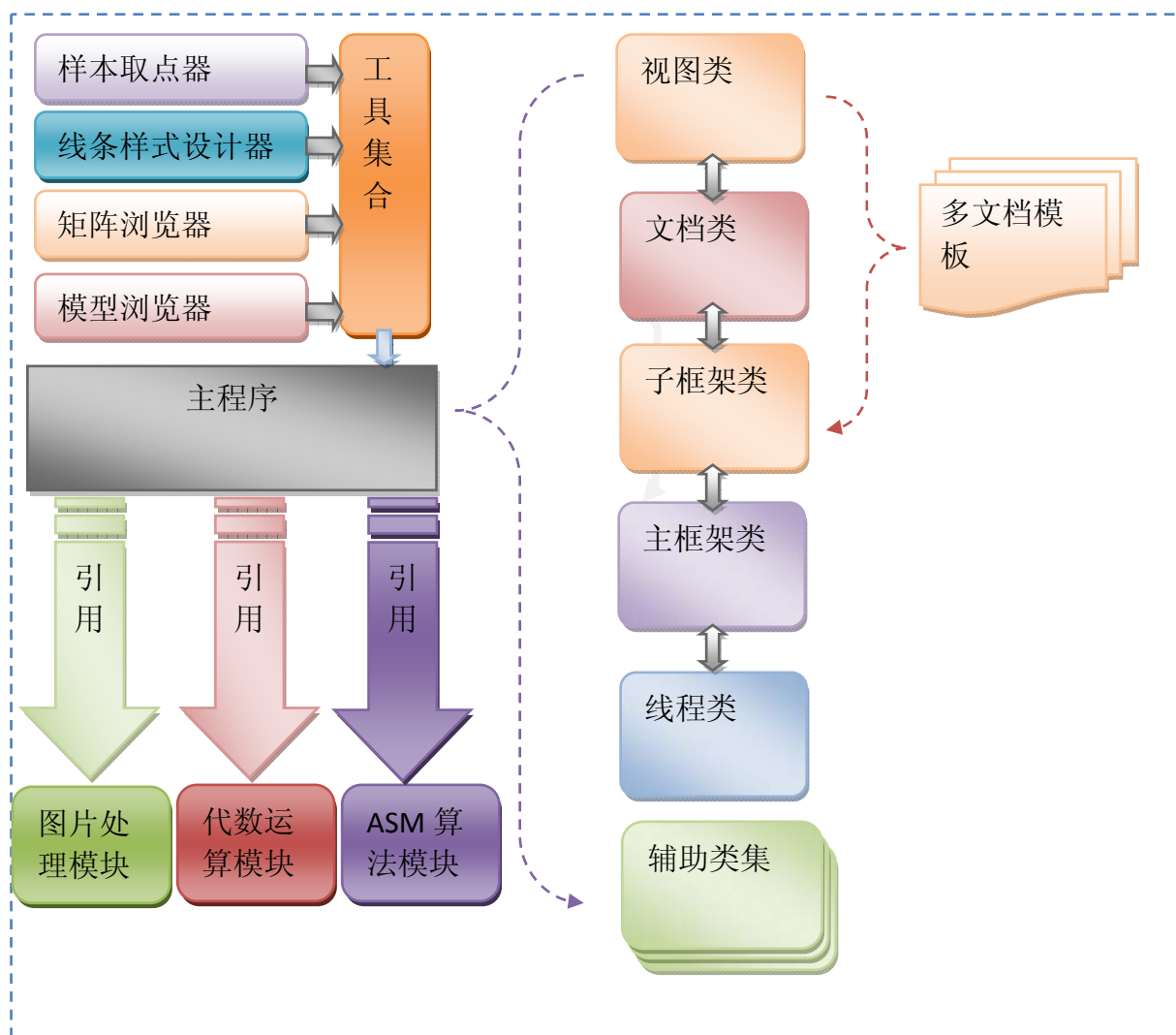


图 6-2 程序运行的主界面

从图中我们可以看到程序运行时的工作界面，没有太多的参数设置接口，因为程序采用了工程文件写入的方式来完成对程序工作时的初始化。

6.2 程序模块相互依赖关系

项目由多个模块构成，主程序又是由几个分块组成的，程序的各个类相互关系如图所示：



上图展示了基本的模块依赖关系和类的其它关系。一个多文档模块是由一个子框架类，一个文档类和视图类构成。一个这样的程序可以包括多个模板，可以创建多个模板实例，工具集是由多个工具组成，其被主程序引用，并主程序的一部分，其它数据处理模块也被主程序引用。

6.3 程序的工具体介绍

本节将简要介绍程序所带工具的情况，一些工具的有关信息之前已经有所介绍，这里主要是介绍一些编程相关的细节的介绍

6.3.1 点采集工具

在主界面的 工具菜单中点击“轮廓提取器”就会弹出工具的工作窗口，而主界面会自动隐藏，这样做是为了方便工具在使用时不受主界面的状态影响，前面已经简单介绍过这个工具了，这里不多赘言。

6.3.2 矩阵浏览器

写这个工具的目的并不是单纯的用来查看程序运行过程中生成的相关矩阵，这个工具提供了一个接口，接口的定义如下：

```
ShowMatrix(double *Mat, int row, int col, int wide);
```

这个接口的参数说明如下：

第一个参数是指向数据区的指针，从类型上我们可以看到，这个接口支持的数据类型是 double 类型的，第二个参数为矩阵的行数，第三个为矩阵的列数，第四个参数为要显示的字长，即一个 double 类型的数要用多少位来显示有了这个接口，我们也可以查看一个一维数组，只是把行数或列数设为一就可以。

当然，你也可以通过菜单来打开一个文件，以数字的形式显示出来。

程序的显示部分是一个多功能编辑类的对象，CMyRichEdit 的一个实例，是从 CRichEdit 类派生而来的，程序的工作界面如下：

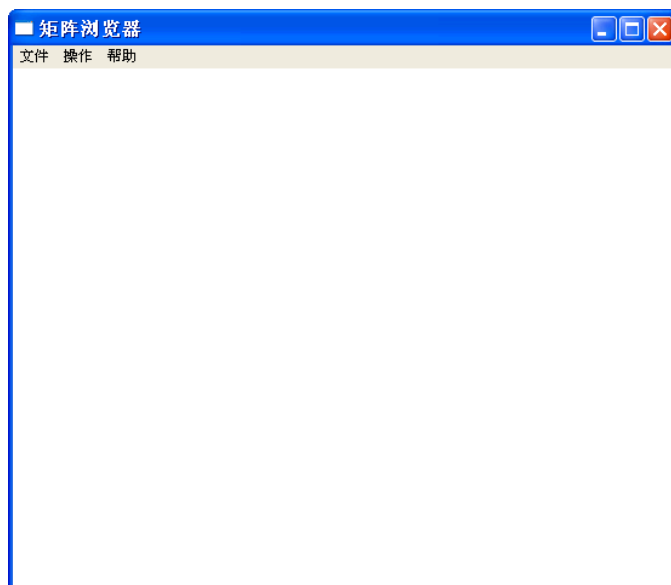


图 6-3 矩阵浏览器的工作界面

6.3.3 线条连接设计器

我们在验证我们得到的特征向量是否可以通过线性组合，然后再加上平均模型来表示可能的人脸模型时，曾经介绍了模型浏览器，其显示点的连线时可以用标准的方式来连接各个点，也可以用有 3D 效果的连线方式来连接。之所可以这样子，是因为我们利用了点线条文件（格式为 *.ls）来设置如何连接的。这种文件就是通过线条连接设计器来生成的，可以在主界面的工具菜单中点击“显示设置器”来启动这个工具，这个工具提供了一个接口，可以由主程序直接调用来设置其初始点的模型，也可以通过手工方式打开一个平均模型来设置其初始化的点模型，其工作界面如下图所示：

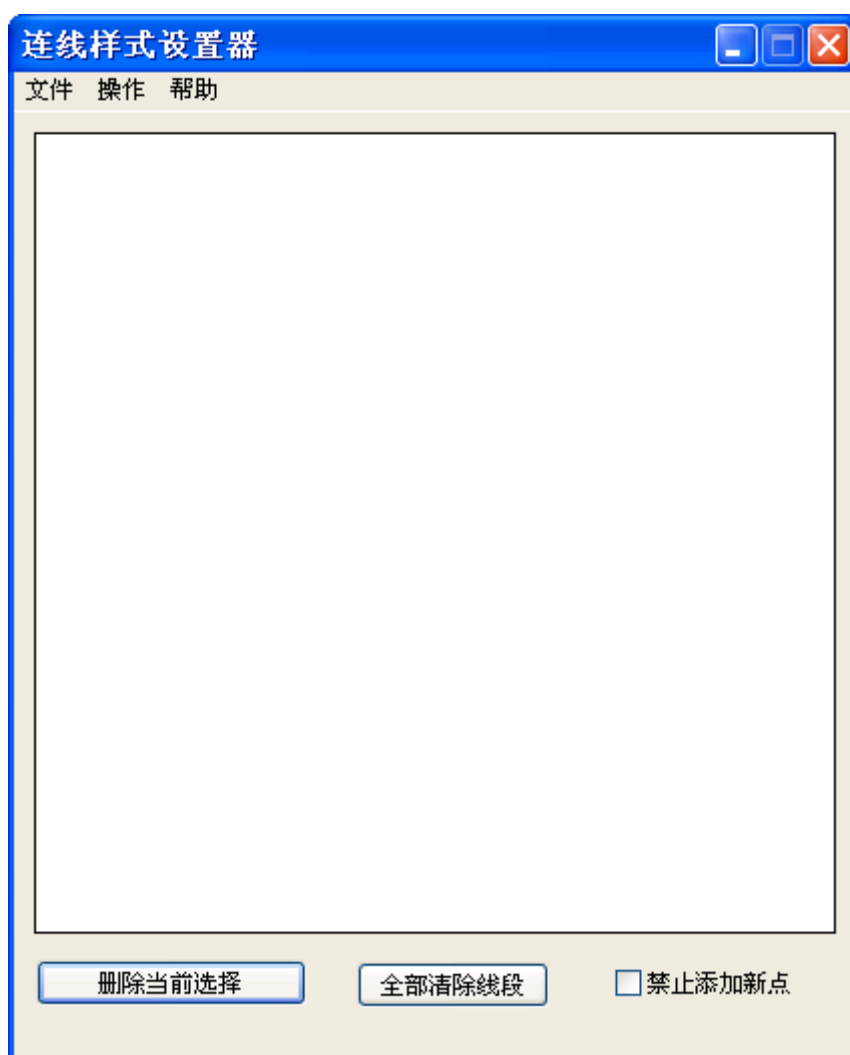


图 6-4 连线设置器的工作界面

（图中主程序没用调用接口对其设置）

程序加载初始文件后的运行界面如下图所示：

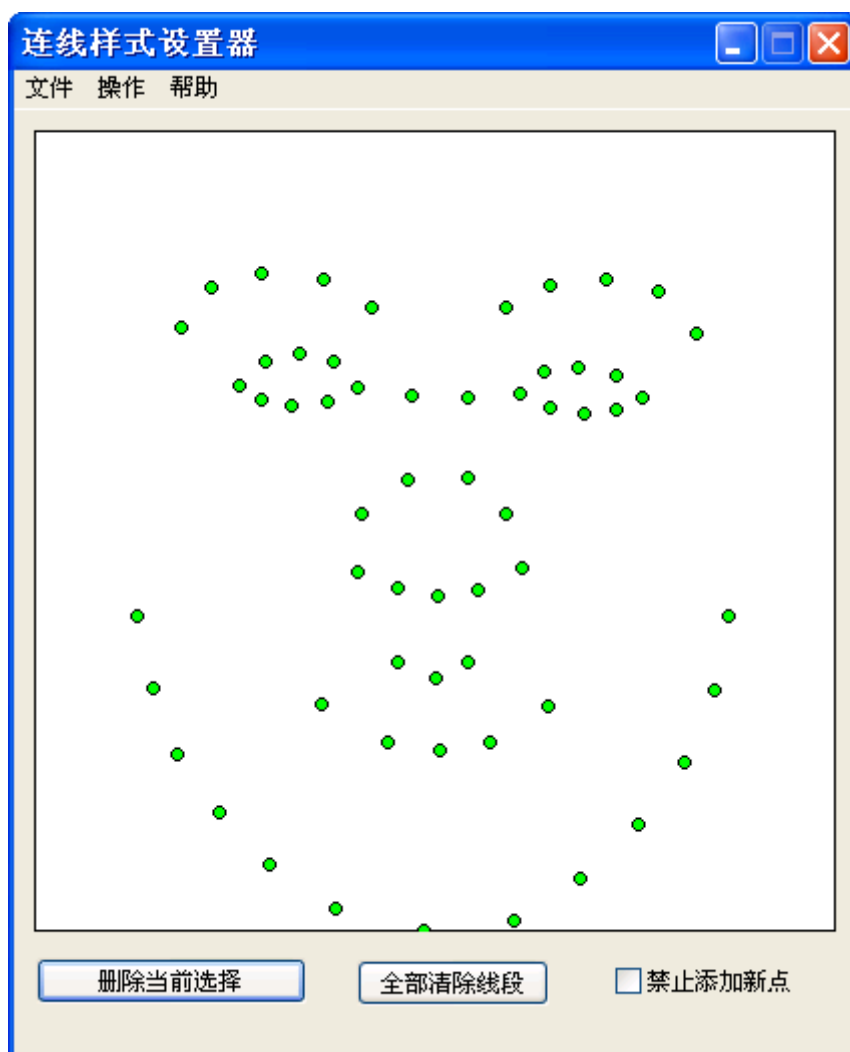


图 6-5 加载平均模型后的工作界面

从界面中我们可以看到程序在设置初始点模型后就会将其显示出来，以绿色的点描绘出来。之后我们就要手工地来选择一个点，当一个点被选择时，就会一直闪烁，一会红色，一会绿色，这样做是为了方便人的查看。当有一个点被选择后，就可以选择另一个点，程序就会在这两个点之间用一条线段把它们相连，然后后面的点就会成为被选择的点，也就会闪烁。可以再点另一个点又生成成为一条线段，当我们不选择任一个点时，在白色区域点一下就可以。生成的线段也是可以选择的，选择后的线段也会同样的闪烁，可以用菜单或上图所示的按钮来删当前所选的线段。也可以清除所有线段。这个工具在实现时经过了很多困难，因为可以完全采用鼠标操作，实现起来的一些细节采用了一些技巧性比较强的算法来判断鼠标下面的线段的点，高效快速，工具经过几次连线后，界面效果如下图所示：

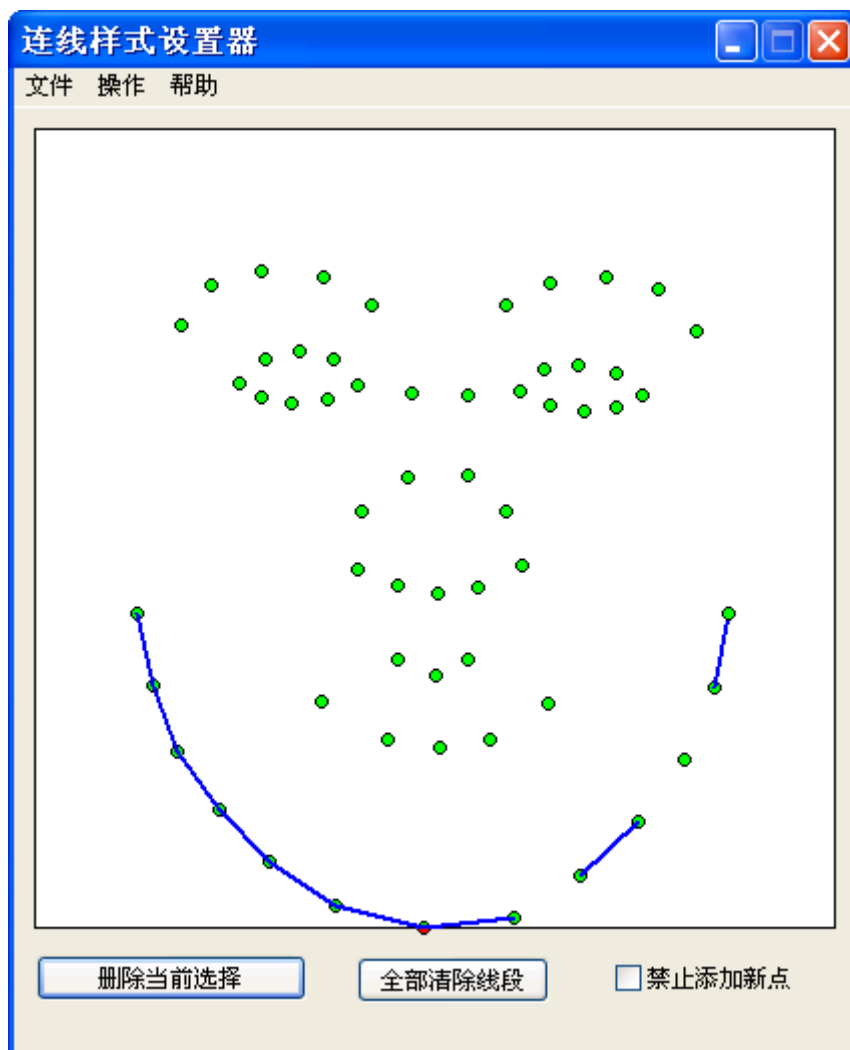


图 6-6 连线样式工具的一个工作状态界面

此外，我们用来提取训练样本的数据时，要用来判断一个点相邻的点是那些，有没有角平均线等，都需要一个点关系模型文件来设置，这种文件的模式为(*.lps), 这种文件也是由这个工具生成的，工具提供了若干种其它格式的文件生成。

6.3.4 模型浏览器

这个工具可以直观地演示我们建立的模型的可能变形，可以用于演示，其利用一个接口来对工具初始化，接口的定义如下：

```
SetConfigure(double *mean, double *vector, double *value, int len, int count);
```

这个接口的参数意义分别是：

第一个参数是指向平均模型的指针，第二个参数是指向经过 PCA 分析后的主要特征向量，第三个参数是指向主要特征向量对应的特征值的指针，第四个参数是向量的长度，第五个参

数是主要特征向量的个数。在主程序完成对平均模型的计算后，就可以打开个这工具进行浏览，这个工具没有提供文件设置功能，可能在后续的工作加入。

6.4 主程序主要类的说明

程序由很多类组成，在主程序中的工程视图中，我们可以清楚地看到这个工程中的类的数量和类名，如下图所示：

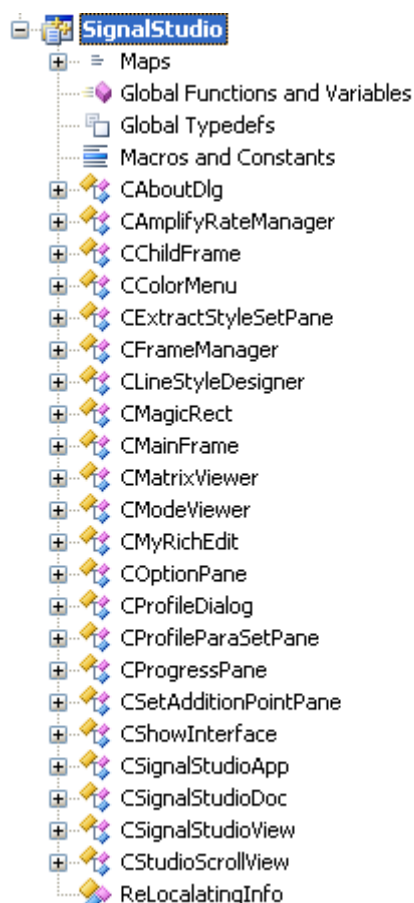


图 6-7 程序的类视图

从上个类视图我们可以看到，主程序是由很多个类构成的，其中有几个类是比较重要的，其它类有的是由向导生成的，在程序中没有经过太多修改，如 CChildFrame 类, CAboutDlg 类，有的类是完全自己写的，如 CMagicRect 类, ReLocalatinginfo 类等，还有的类是利用别人写的好的类，如 CMyRichEdit, CColorMenu 类等。下面对一些主要的类和比较特别的类做一下简单的介绍。

6.4.1 CSignalStudioDoc 类的说明

CSignalStudioDoc 类是一个文档类，在文档视图结构中，文档类就是那个负责处理数据的类，包括读取和保存等数据操作。这个类在主程序中被选为 ASM 实现的类，没有对 ASM 相关算法进行独立的封装，不过如果把 ASM 的相关算法进行封装，程序的结构会更好些，这些工作留在以后再做。这个类负责 ASM 的所有相关算法的实现和数据的保存等，并提供一组接口，可以供视图类调用以实现数据的显示。这个类的头文件就有近 400 行，源文件也有 3000 行以上，是整个项目中代码量最大的类。这样大的代码量是事先没对 ASM 进行独立封装造成的。从这个项目的实践总结，可以感受到面向对象程序设计中分块封装的重要性。

这个类还提供了内存的图形绘制（一个内存 DC 对象），可以导出为各种格式的图片。视图可能通过获取这个绘图对象来显示当前的操作结果，这个绘图对象是一个统一的绘图板，可以用来显示当前要搜索的图片，或训练样本，也可以显示训练结果与搜索数据的对比等。之所以这样做是为了方便管理，避免内存的大量占用和不同对象引用时的可能错误。对内存数据的所有绘制都是在这个对象进行的，包括图像的处理等。文档类提供了对这个对象了操作的接口的封装。

6.4.2 CSignalStudioView 类

这个类不是直接从 CScrollView 类继承过来的，而是从 CStudioScrollView 类继承而来的，CStudioScrollView 相当一个接口类，仅定义一组接口，在其子类中必须实现的，这样就可以实现用父类指针可以访问子类的对象。这样做是出于文档类可能需要多个视图来显示不同的内容的考虑。项目是建立在原本一个框架之上的，这个框架本来是想用来实现一个信号处理的基本平台，这平台提供有关信号处理的工具，新的模块注册接口和动态添加移除菜单工具条等服务。但后来由于经验不足，遇到诸多困难，就没有规划完成。因此这个项目是建立在一个并不完整的框架上的，即保留了原本规划过后的一些特征，又不是一个完整的信号框架。接口类的声明代码如下：

```

class CStudioScrollView : public CScrollView
{
DECLARE_DYNCREATE(CStudioScrollView)
protected:
    CStudioScrollView();           // protected constructor used by dynamic creation
    virtual ~CStudioScrollView();
protected:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual void OnInitialUpdate(); // first time after cone struct
//定义接口，以让所有子类实现，两个接口分别是返回当前视图的图片和其数据区的起始地址
public:
    virtual HBITMAP GetCurrentImageBitmap();
    virtual BYTE* GetCurrentImageDataBuffer();
DECLARE_MESSAGE_MAP()
};
    
```

视图类实现以对文档对象内的一些数据的显示，这些数据是经过处理后再显示，而不是直接就能在视图中描绘出来，视图类提供了对文档对象的用户操作接口，如鼠标事件，键盘事件的事件处理等，文档视图结构大大简化了程序的开发复杂度。

6.4.3 CAmplifyRateManager 类

这个类完全是自己编写的，编写这个类源于在对视图图片进行缩放时，在原来实现方式的基础上，由于浮点数的精度问题，会造成图片在经过多次缩放后无法在同样放大率的情况下得到等大的图片，而且难以有效地管理放大倍数。因此写了这个类用来管理图片的缩放比率。这个类的声明好下：

```

class CAmplifyRateManager:public CObject
{
private:
    CArray<double,double> rateList;
    int index;
    double value;
public:
    CAmplifyRateManager();
    void AddRate(double rate);
    void RemoveRate(int index);
    void RemoveAll();
    void Reset();
    double GetNextRate();
    double GetPreveiwRate();
    double GetCurrentRate();
};
    
```

从类的声明中可以看到，这个类提供了一组获取和改变放大倍数的接口，类的初始化时会自动添加一组预定义好的放大比，并按小到大的顺序排列，还提供了添加和移除放大率的接口，这些接口在实现时都会重新排列数组中的放大值。我们可以在视图中的查看菜单中的放大与缩小图片查看来改变图片显示时的大小，也可以通过快捷键 **Ctrl + up** 或 **Ctrl + down** 来改变图片的显示大小，这放大与缩小都有这个类的对象的参与。

6.4.4 CMagicRect 类

首先让我们在程序中查看一下这个类的一个对象是怎么的一个样子的，当我们打开主程序时，在视图是左上角移动鼠标，我们就可以看到它，如下图所示：

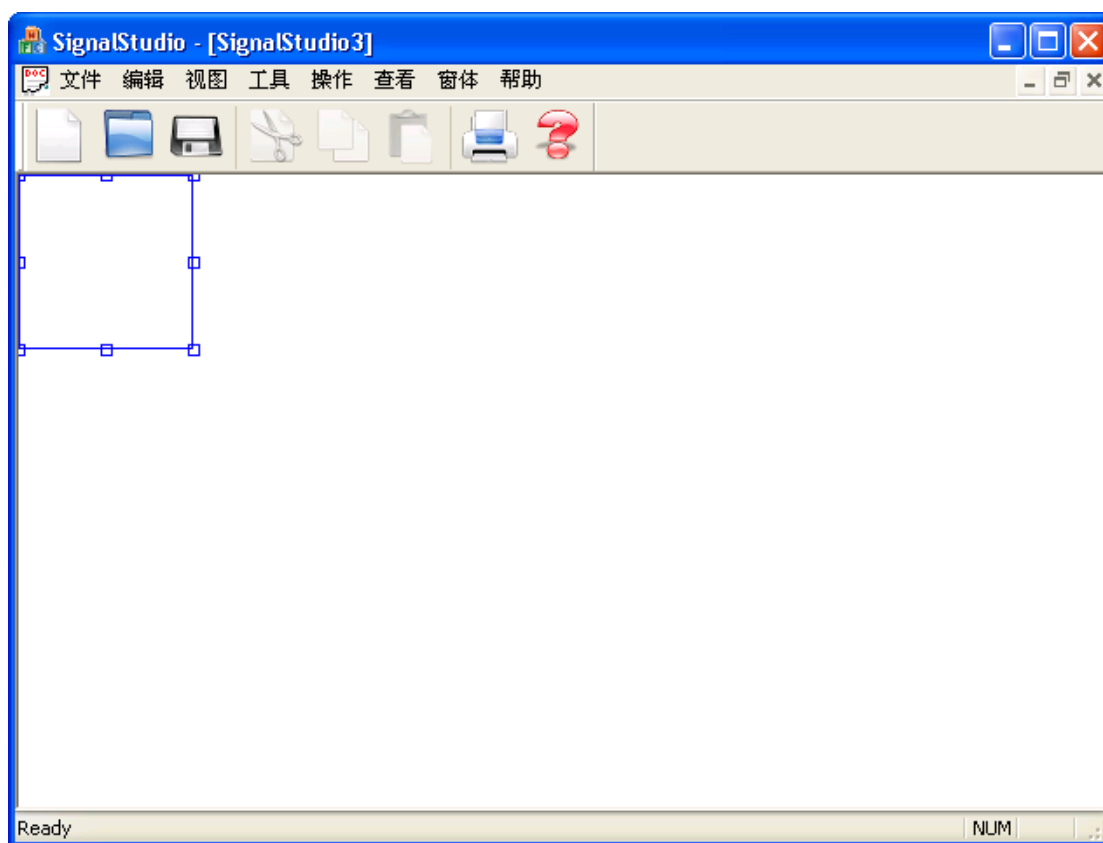


图 6-8 魔术框类在程序模型还没计算时的效果图

这个类的对象是一个我们在其它程序中经常能看到的一种矩形框，这个对象的周围有八小小矩形，四个在直角处，四条边中间各有四个。我们可以通过这些小矩形，就可以改变大矩形的大小。四个角上的小矩形拉动，就可以沿纵横方向来改变大矩形的大小，从边上的小矩形拉动，只能改变横向或纵向大矩形的尺度。实现这个魔术框类的目标是为了比较准确地定位模型在搜索匹配时的初始位置，达到比较好的搜索效果。当我们已经建立好模型后，拉

动这个魔术框就可以动态地看到放大和缩小后的初始模型，然后在确定大致的模型后，就可以通过文档对象提供的接口来设置搜索的初始模型。

训练后移动和改变魔术框的效果如下图：

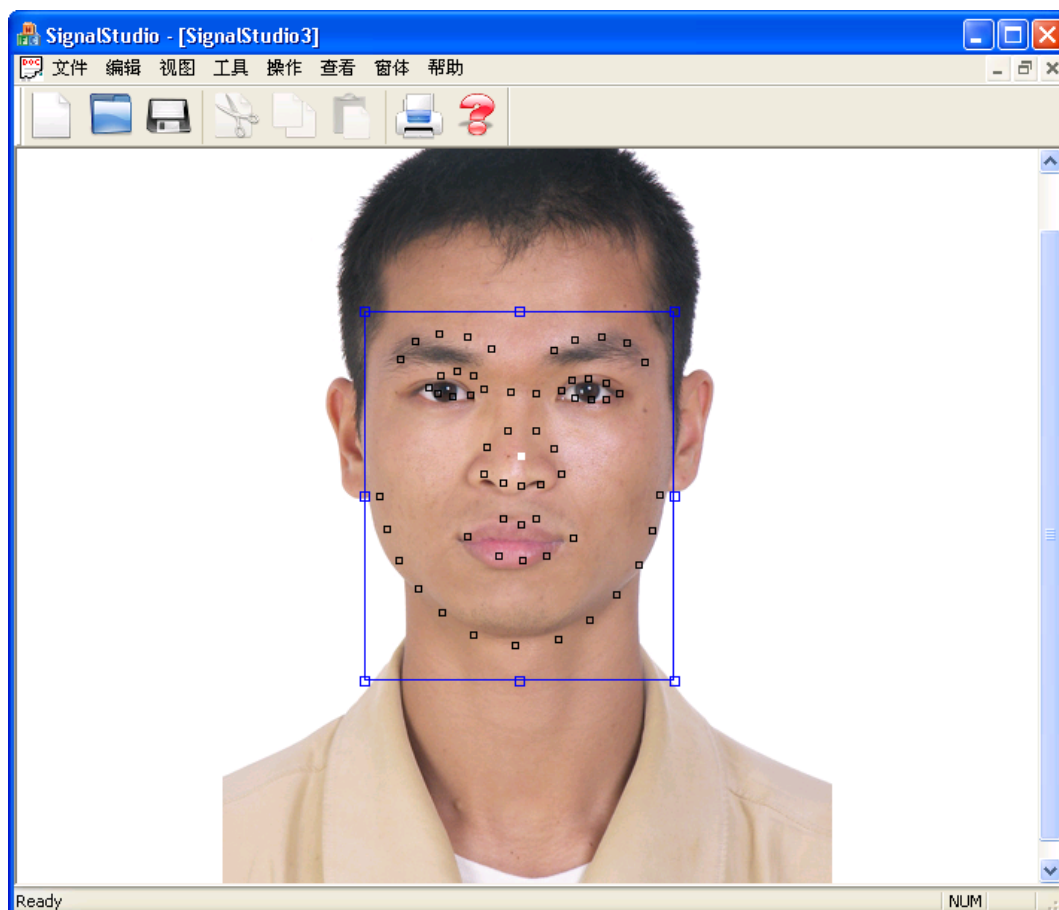


图 6-9 魔术框在程序运行时的效果图

魔术框类的声明好下：

```

#define SIZE_FROM_TOP_LEFT 0
#define SIZE_FROM_TOP_RIGHT 1
#define SIZE_FROM_BOTTOM_RIGHT 3
#define SIZE_FROM_BOTTOM_LEFT 2
#define SIZE_FROM_TOP 4
#define SIZE_FROM_BOTTOM 5
#define SIZE_FROM_LEFT 6
#define SIZE_FROM_RIGHT 7
#define DRAG_LARGE_RECT 9
class CMagicRect : public CObject
{
public:
    CMagicRect(void);
    CMagicRect(CRect rect, CDC* pDC); //初始一个位置
public:
    ~CMagicRect(void);
private:
    COLORREF m_FrameColor; //边框的颜色
    COLORREF m_FillColor; //填充的颜色
    COLORREF m_HotColor; //鼠标经过时的颜色
    int m_nRadius; //选择小圆圈的半径
    CWnd* m_pWnd; //当矩形大小改变时，向这窗口发送消息
    UINT m_nMsg; //以送的消息
    CRect m_Rect; //大矩形的参数
    BOOL m_bMouseDown; //鼠标按下的标示
    CRect Rects[8]; //保存八个小矩形
    int m_nLastHotIndex; //最近一次的索引
    CSize off; //鼠标偏移
    int m_nTypeFlag; //当前处理的类型
    CRect m_OldRect;
public:
    void Draw(CDC* pDC); //显示本对象
    BOOL RegisterMsg(CWnd* wnd, UINT msg); //注册一个消息
    void InforMouseMsg(UINT nFlags, CPoint point); //通知一个鼠标消息
    void UpdateRects(); //重新计算小矩形的位置
    int FindCircleUnderMouse(CPoint); //查找鼠标下面的小矩形，返回其索引
    CRect GetWorkRect(); //返回工作的区域
    CRect GetInvalidateRect(); //返回需要重绘的区域的大小
    
```

由于这个类不是窗体类，为了其能与窗体进行交互，不仅其能从窗体类那获得鼠标事件的消息，类中的成员：`m_pWnd` 和 `m_nMsg` 是用来保存窗体的指针和与窗体交互的消息号，当

魔术大小位置发生改变时，其就会向 `m_pWnd` 发送 `m_nMsg` 消息，窗体得到消息后，就会做出相应的处理，如重要定位模型的初始位置等。

这个类的实现涉及到的细节比较繁琐，也需要一些技巧，由于时间的关系，这个类有待进一步的完善。

6.4.5 CColorMenu 彩色菜单类

这个类是利用别人已经写好的代码移植到程序中的，主要是用来改变视图的背景色而引入的，在视图中鼠标右键，就会有一个菜单弹出来，点击“设置视图背景色”就可以看到这个彩色菜单，这个类的对象在程序运行时的效果如下图所示：

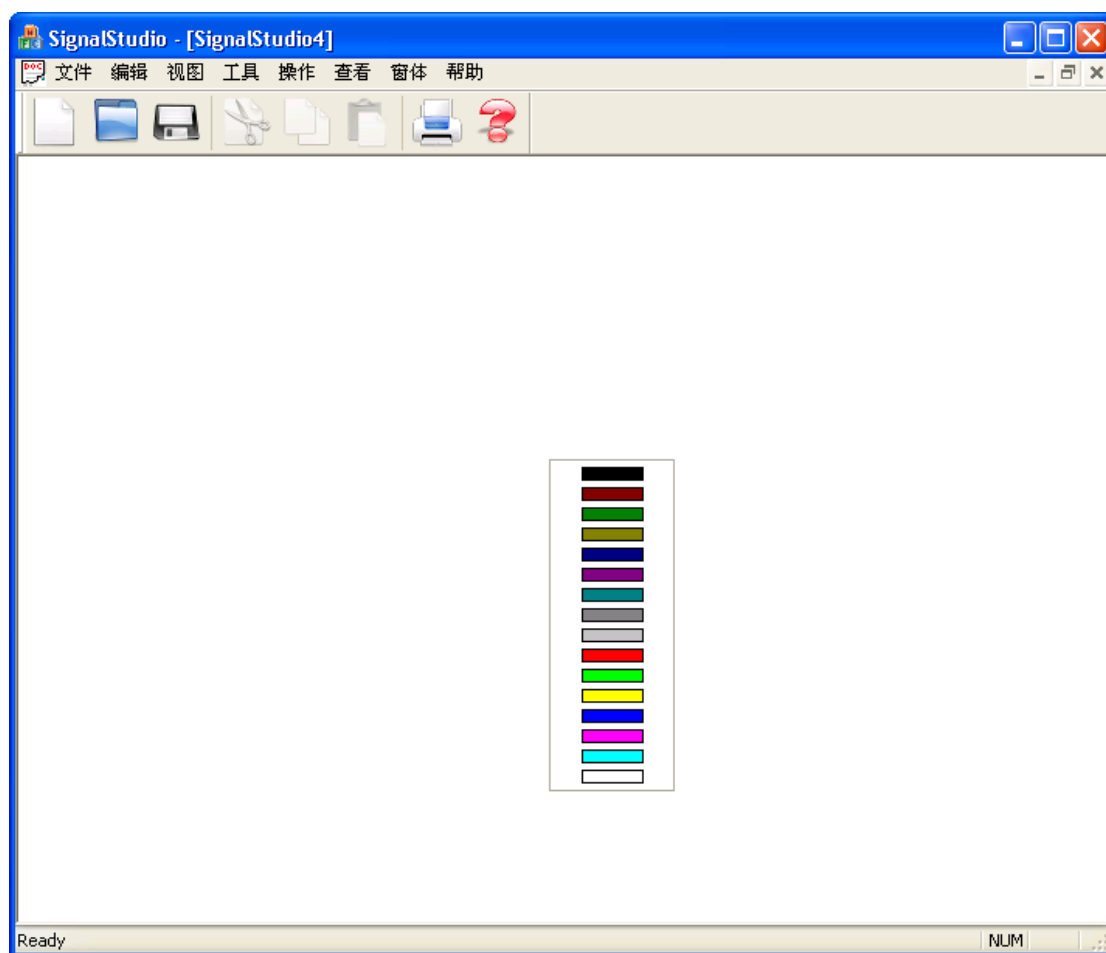


图 6-10 菜色菜单运行时的效果图

这样的菜单能比较直观地让用户选择想要的背景色，使用方便。在 .net 下要实现这样的效果会相对简单一些，而在 MFC 下实现起来比较麻烦。

6.4.6 其它类

CSignalStudioApp 类是主程序的运行类，是一个线程类，在这个类中声明了一些参数设置窗口类，以供全局使用，类没有做太多的修改。

还有其它的工具类我们在之前已经介绍过了，其中 CProfileDialog 类就是提取训练数据的工具类，它的代码量也比较大超过两千多行，是项目开始进行时第一个实现的工具。CMatrixViewer 类就是矩阵浏览类，CLineStyleDesigner 就是线条连接样式设计工具类，CModeViewer 类就是模型浏览器的工具类。还有其它的一些类就不做介绍，详细的实现请参见源代码。

6.5 数学包的简要说明

程序中运用了比较经典的一些代数算法，如求矩阵的逆阵，对称阵的特征向量和特征量，还有求解线性方程组等，这些运算应用了比较多的数学理论。主程序没有对这些算法进行实现，而是直接利用一个开源数学包 lapackpp，lapackpp 的官方主页 (<http://sourceforge.net/projects/lapackpp/>) 提供了源代码的下载，并有使用说明文档。这一个高效的代数算法集合，在计算一个 116 维的对称阵的特征值和特征向量，并对其进行排序所用的 CPU 时间非常小，程序进行调用时，几乎感觉不行有任何停顿。这个算法集合也是采用面向对象的方式编写的，使用非常方便。

6.6 图像处理模块的简要说明

程序处理模块是由一系列处理算法组成，在这个项目中，所起的作用是可以忽略的。但是，整个项目考虑到是用于学习研究的，把这个模块也加进来，会方便演示和研究。这个模块还有不断扩展中，新的处理算法也随着知识的不断扩充而不定时地加入，模块编译生成的是一个动态链接库，可以用于所有的 windows 平台下开发的程序中

结论

个人的心得体会

通过对 ASM 算法的学习和研究实践，我从中学到了很多很有用的知识，对研究对象的建模方法和技术多了一些了解，对数学知识的应用有了更多的认识。课题对我来说比较有挑战，之前在选择课题时，并没有太多的信心把它完成，只是出于困难问题的好奇心，和自身的挑战精神。但随着工作一步一步有序的进展，我看到了完成工作的可能性，每一步工作的完成都增强了我的信心。在学习过程中，遇到很多问题难题，曾有几次要放弃其中的一部分工作，但经过大量的在线查询和在图书馆的查阅，不断地解决遇到的问题。这个过程都是学习和探索的过程，同时，也让我更加深刻地体会到学习能力，探索精神以及想象力在某种程度上比知识重要。通过艰苦的编码和代码不断的完善，不仅在项目上取得进展，同时也锻炼了自己的一些素质，我能更加冷静思考，更专心地去学习新知识。不仅在收获了知识，也锻炼了品格，受益匪浅。

后继工作的展望

项目完成了 ASM 算法的所有基本算法，完成了相关的工具开发，并最终能够进行搜索。项目还额外提供相关数字图像处理的算法和实现，可以对项目进行扩展。通过实验观察，发现 ASM 算法对初始位置的依赖很大，有待进一步改进和引入新的强化算法，这类算法可通过各种方法实现。

由于工程项目时间仓促，没有对整个项目进行一个完整的设计和摸索。在未来可能的工作中，希望能应用软件工程的架构思想，实现一个简单而灵活的框架，使得项目具备可扩展性和对底层模块驱动的注册支持，实现一组可能更换算法的接口。不断实现新的强化算法并往框架中注册。进一步可能的工作是实现 AAM 的扩展模板，使框架具备 AAM 和其它相关算法的演示和实验，并在实验结果在可接受的前提下，去实现算法的实时识别搜索。

致谢语

在即将毕业的这段时间里，我还能在大学最后的时光里学习到那么多知识，学会更加稳重和坚持，有太多的人要向他们表示我最真挚的谢意。

很高兴在大学里能遇见我们可爱温和的段鸿老师，他的认真务实的教学精神深深地感染了我，他治学的严谨负责同样让人敬佩。在大学最后的一学期中，他在学习和为人方面给我巨大的帮助和指导，并和我分享人生的经历，见解和体会。他为人谦虚谨慎、虚怀若谷、博学多才，是同学们的良师益友，在此我要感谢他对我的无私帮助和支持。

还要感谢厦大计算机科学系的吴瞰华师兄，他在项目开发过程中给我的很大的帮助。吴瞰华师兄是一个可爱而温柔的学长，在项目之初，是他百忙之时挤出时间耐心地给我讲解 ASM 的算法思想和算法实现。在项目进入关键的阶段时，是他再次抽出时间耐心地和我讨论算法实现的细节，并给我提出了很多建设性的意见，同时他也给了我很多鼓励和启示。

在此还要感谢养育我的爸爸妈妈，是他们在身心疲惫的时候给予我无限的温暖和关怀，是他们的理解和支持给了我无限的勇气和毅力。

此外，还要感谢所有关心我的朋友和亲爱的同学，特别是那些在我经历人生不平常时期时不断地给予关心鼓励的好朋友，在毕业设计这段时间和我一起去机房学习的同学。

还有很多很多的人我想对他们说谢谢，有我认识的，还有不认识的，很感谢他们无私的指导和关怀，他们的善良热情和可爱让我感受到这世界处处有好人，处处都有温暖，是他们让我更加懂得珍惜、懂得感恩，还有责任。

青春永不言败，在未来的日子里，我会更加努力学习，通过自己的努力和对理想的不懈追求来回报这个充满关爱和温暖，而需要更多关爱和温暖的社会和所有关心的我人们。

参考文献

- [1] T.F. Cootes, G.J. Edwards, and Taylor C.J. Active appearance models. [R] In 5th European Conference on Computer Vision, Berlin, Germany, 1998. Springer.
- [2] Jean Keomany of Swiss Federal Institute of Technology (EPFL), Lausanne, S'ebastien Marcel of IDIAP Research Institute Martigny. Active Shape Models Using Local Binary Patterns[R], February 2006
- [3] G. J. Edwards, C. J. Taylor and T. F. Cootes, Wolfson . Interpreting Face Images using Active Appearance Models Image Analysis Unit[R]. Department of Medical Biophysics, University of Manchester, of Manchester M13 9PT, U.K.
- [4] 蓝以中. 高等代数简明教程[M]. 北京大学出版社
- [5] Lindsay I Smith. A tutorial on Principal Components Analysis[R]. February 26, 2002
- [6] 盛骤 谢式千 潘承毅. 概率论与数理统计[M]. 浙江大学. 高等教育出版社
- [7] 孙辉. 经典与现代的结合: 在 MFC 中集成 RAD .NET 框架. 网络链接 <http://www.microsoft.com/china/MSDN/library/langtool/VCPP/Radne>
- [8] Tom Archer, Nishant Sivakumar . Extending MFC Applications with the .NET Framework[M]. publisher: Addison Wesley. December 26, 2003
- [9] Jack Dongarra, Roldan Pozo and David Walker of National Institute of Standards and Technology, Oak Ridge Nation Laboratory and University of Tennessee, Knoxville. LAPACK++ V. 1.1 High Performance Linear Algebra Class Reference Guide. <Http://sourceforge.net/projects/lapackpp>. April 1996.
- [10] Jack Dongarra, Roldan Pozo and David Walker of National Institute of Standards and Technology, Oak Ridge Nation Laboratory and University of Tennessee, Knoxville. High Performance Linear Algebra Users Guide. <Http://sourceforge.net/projects/lapackpp>. April 1996

附录一

算法:对齐两个向量

由线性代数我们知道，对一个重心在原点的向量缩放 s 并旋转 θ 可以用一个矩阵来实现，即

$$T_{s,\theta}(x) = \begin{pmatrix} S \cos \theta & -S \sin \theta \\ S \sin \theta & S \cos \theta \end{pmatrix},$$

从矩阵内容关系我们可以看简化一下变化矩阵，

设

$$S \cos \theta = a,$$

$$S \sin \theta = b,$$

于是变化矩阵变成

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix},$$

现在假设我们要把 x 对齐到 x' ，我们先对 x 做变换：

即

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$

设函数

$$E(a,b) = \|T(x) - X'\|^2,$$

变换后，这个函数的值可以达到最小，即两个模型的差别最小化，我们利用高等数学的知识把等式变换为：

$$E(a,b) = \sum_{i=1}^n (ax_i - by_i - x'_i)^2 + (bx_i + ay_i - y'_i)^2.$$

分别对 a 和 b 求偏导得到下面的两个方程：

$$\sum_{i=1}^n ax_i^2 + ay_i^2 - x_i x_i' - y_i y_i' = 0$$

$$\sum_{i=1}^n bx_i^2 + by_i^2 - x_i y_i' + y_i x_i' = 0$$

通过求解就可以得到 a 和 b 的表达式：

$$a = (\sum_{i=1}^n x_i x_i' + y_i y_i') / (\sum_{i=1}^n x_i^2 + y_i^2) = x \cdot x' / |x|^2$$

$$b = (\sum_{i=1}^n x_i y_i' - y_i x_i') / (\sum_{i=1}^n x_i^2 + y_i^2) = (\sum_{i=1}^n x_i y_i' - y_i x_i') / |x|^2$$

把 a 和 b 代入变换矩阵就可以求出变换后的模型。

注：这里的 x, y 都是模型的 x 分量和 y 分量，即由一个模型的所有 x 坐标和 y 坐标组成的分量向量：

$$(x_1, x_2 \cdots x_n), (y_1, y_2 \cdots y_n)$$

附录二

搜索实例：

下面给出一次搜索的实例。

搜索条件：

训练长度为五个像素

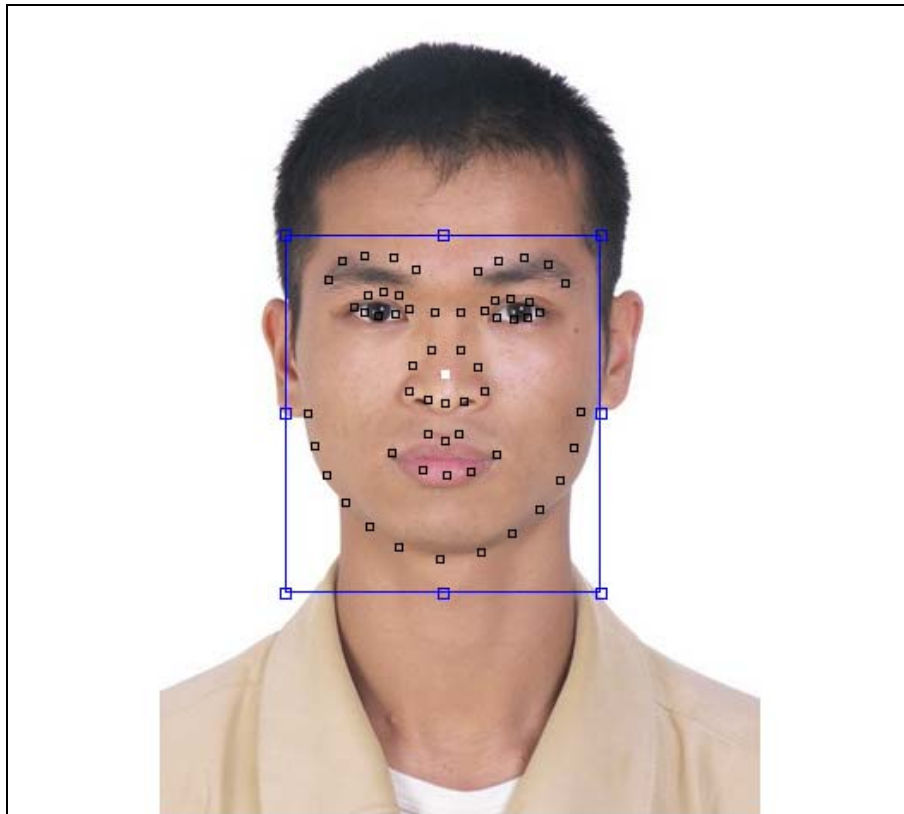
搜索长度为九个像素

训练用的样本数为 37 张人脸

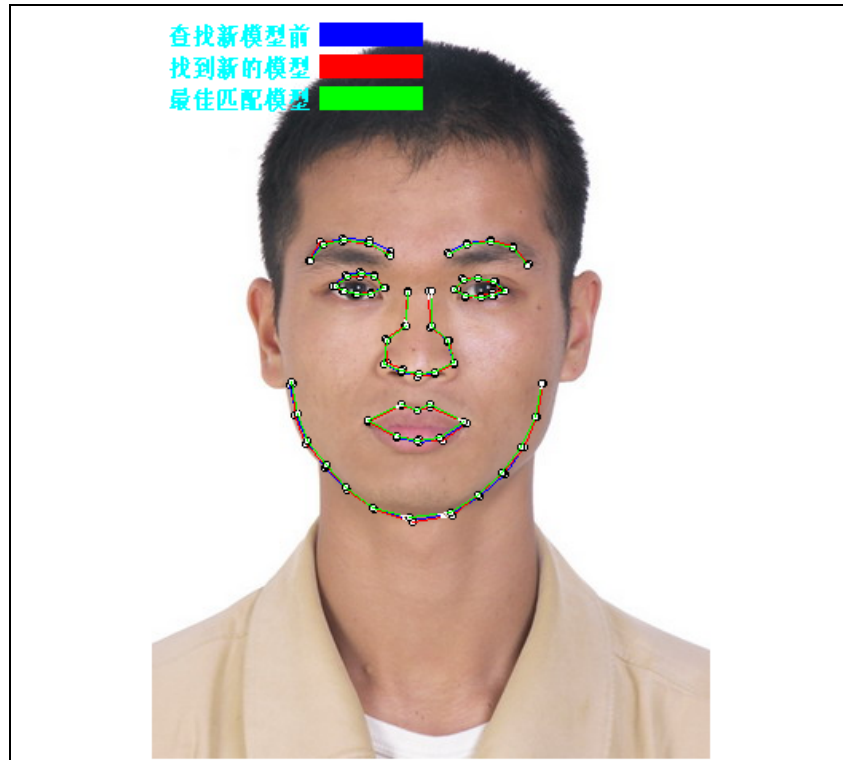
用于搜索的人脸，本人照片。

照片的拍摄条件与训练样本取样的光照条件差别比较大。

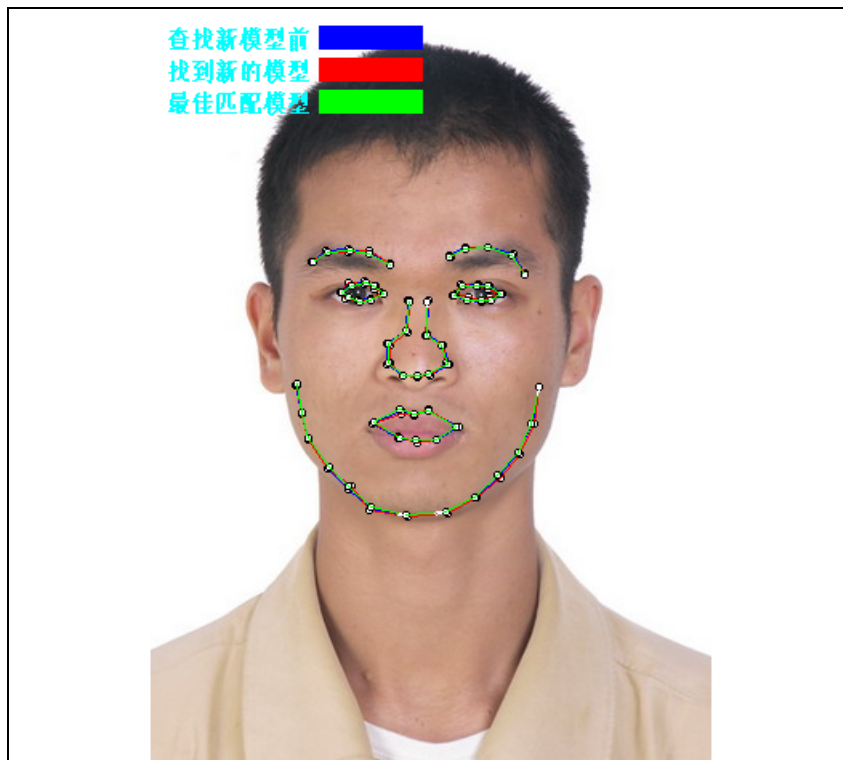
搜索过程：



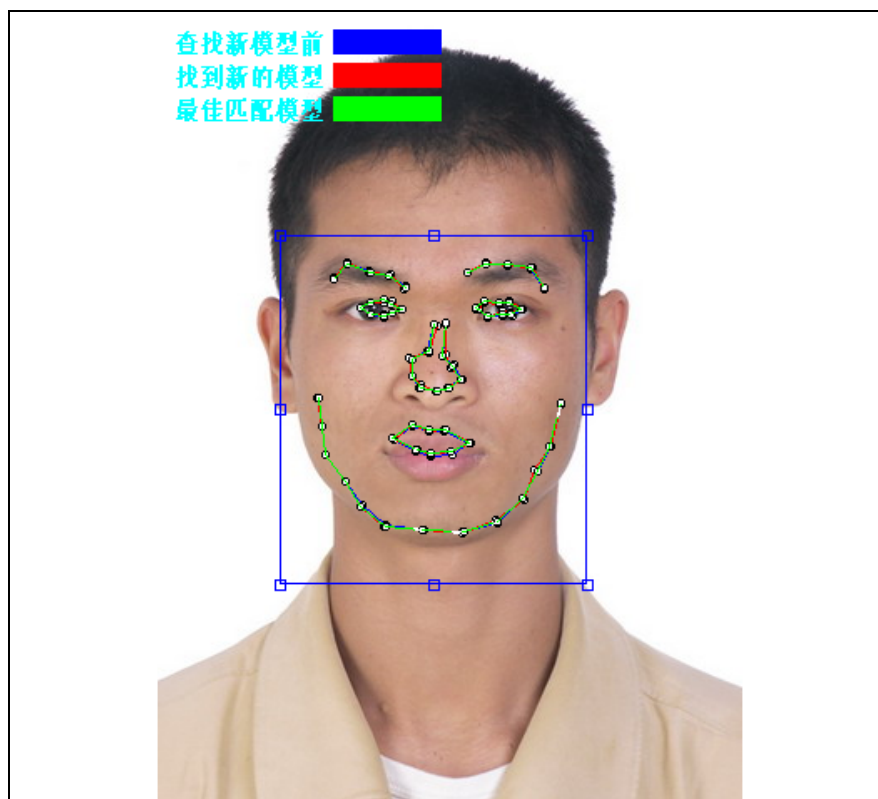
初始位置



一次搜索迭代



迭代搜索五次



迭代搜索十五次

结论：从上面的一次实验中，我们可以看到，ASM搜索的结果并不是很理想，主要的原因有以下几个：

1. 训练的样本不够，才37张人脸，需要有更多的训练样本。
2. 训练时的数据长度和搜索长度不够，无法搜索到人脸的轮廓边缘。
3. 搜索时用人脸的光照情况和训练用的样本的光照情况差别较大，有必要采用一些强化算法。

这个实验主要目的是为了说明ASM先天的一些不足，有待于以后的研究工作对其进行完善和改进。

毕业论文任务书

(以下由学生填写)

题 目：基于 ASM 的人脸识别

目标要求：

通过对 ASM 的学习研究和实践，掌握相关的数学知识的工具的应用，了解 ASM 的建模思想的途径，了解 ASM 的相关扩展算法并对其的扩展等，能过编程实现 ASM 对人脸的识别，匹配人脸的轮廓，并通过实践来深入了解相关的算法的工作原理和提高编程技巧，并培养调研精神等。

支持条件：

支持硬件：奔腾 4，CPU 主频 2.4G，内存 512MB 的个人电脑

操作系统：Windows XP professional sp2

开发环境：Visual Studio .net 2005

指导教师（签名）_____职称_____学生（签名）_____

分阶段进度安排

阶段	起讫时间	计划完成内容
1	2007 年 3 月 15 日-4 月 15 日	阅读相关论文资料，并查阅有关文献
2	2007 年 4 月 6 日-4 月 17 日	理清程序设计思路 and 开发步骤
3	2007 年 4 月 18 日-5 月 25 日	完成程序的编码和基本测试
4	2007 年 5 月 26 日-6 月 5 日	进行系统的进一步完善和文档的编写
5	2007 年 6 月 7 日-6 月 11 日	进行文档的完善和程序的最后调试

注：一般可分为资料文献搜索、拟定方案（提纲）、试验或初稿、定稿等阶段

教师分阶段指导记录

第一阶段：

阅读相关文献，确定课题

第二阶段：

阅读相关论文资料等，把握大体需求

第三阶段：

进行程序的设计和规划

第四阶段：

开始编码和调试，并定时汇报进度

第五阶段：

完成文档的编写和审核，提交初稿，修正不合格的方面

论文评语	<p>拟评成绩_____ 指导教师（签名）_____ 职称_____</p> <p>年 月 日</p>
论文评阅	<p>评阅成绩_____ 评阅教师（签名）_____ 职称_____</p> <p>年 月 日</p>
答辩记录	<p>演示成绩_____ 答辩成绩_____</p> <p>答辩小组组长（签名）_____ 职务（称）_____</p> <p>年 月 日</p>
总评	<p>成绩_____ 学院负责人（签盖）_____</p> <p>年 月 日</p>