快速人脸检测系统的设计与实现

内 容 摘 要

学科专业：计算机应用技术　　**研究方向**：图像处理

指导教师：程小平　　**研 究 生**：沈乐君

人脸识别可以广泛应用到身份验证、信息安全、电子商务、人机等诸多领域。随着社会日益增长的巨大需求、计算机软硬件技术的成熟和人脸识别研究的日趋进步，人脸检测逐步受到重视。特别是复杂背景下的人脸检测研究，已经逐步成为人脸识别系统是否能够真正实用的关键。

对于人脸检测的研究，已经经历了由简单到复杂，由静态图像检测到视频实时检测的发展，特别是最近几年出现的基于 Haar 特征的人脸检测，真正实现了在 PC 机上的人脸快速检测。本文研究了基于 Haar 特征的人脸检测算法之后，发现虽然它具有具有很强的实时性，但在复杂背景下的人脸检测率，错检率等指标不是很理想。

所以，本文研究了图像中的肤色，外形，边缘能量，背景等特征，提出利用肤色概率模型、形态学算子、轮廓过滤、边缘启发式搜索、背景概率分布、阴影检测等手段改进原算法。试验证明，和原算法比较，本论文提出的方案总体性能得到提升，并且提高了检测率和降低了错检率。

另外，本文创新地提出了基于背景模型和头肩几何特征的人脸粗定位技术，缩小了人脸定位的范围，提高了性能。同时，讨论的各种理论和实现方法，对视频监控、车牌识别等其它机器视觉应用系统，具有借鉴价值。

作者网站：www.shenlejun.cn

关键词：Haar 特征，肤色模型，连通区域，Canny 边缘检测，DirectX，背景消除，阴影检测

Design and Implement of Rapid Face Detection System

Abstract

Face recognition can be widely used in such fields as personal identification, information security, e-business, video surveillance, multi-media, etc. With the demanding of society and business, developing of computer hardware and software and progress of face recognition research, face detection become more and more important. It's a key of face recognition application, especially in real world.

The research of face detection advanced from simple image process to real-time video stream. Haar feature based face detection made PC can see human face via camera. Our experiments show that the Haar feature based face detection algorithm can truly detect face rapidly, but face detection rate and false negatives rate is not satisfying, especially videos in normal environment.

Other face features, such as skin color, contour and edge energy are used in our face detection system. skin model, contour selection, canny edge pruning, background subtraction and shadow elimination are utilized in order to improve the system performance, raise face detection rate and cut down false negatives rate.

In this paper, a face detection system using DirectX technology may be useful for other computer vision system, such as video surveillance and vehicle plate recognition.

Key word: Haar feature, skin color model, connective region, Canny edge detection, DirectX, background subtraction, shadow elimination.

目录

第一章 概述

1.1 人脸识别

人脸识别是图像分析和理解领域最重要的应用。对它的研究可以追溯到 20 世纪 60 —70 年代，对它的研究经过了几十年，不断的完善[3][19][34]。

研究人脸识别的国外主要国际会议。除了传统的国际会议，如 CVPR（Conference on Computer Vision and Pattern Recognition）、ICIP（International Conference on Image Processing）之外，是在过去数年中，还出现了人脸识别的专门国际会议，如 AFGR(Automatic Face and Gesture Recognition) AVBPA(Audio and Video-Based Person Authentication)。

人脸识别成为科学家们研究的热点，其根本原因是巨大的社会需求、它应用潜力大、应用范围广。其应用领域如：

- 身份鉴定：护照、身份证等自动鉴定。
- 使用许可：机要部门准入、自动提款机 ATM 使用许可。
- 智能监控：入店行窃和嫌疑犯追踪、无人值守下的报警。
- 多媒体：人脸建模和编码，人机交互。图像检索。

具体举个例子：我们现在要从 ATM 当中提取现金，就需要 PIN（personal id number），还有一个密码。通常我们的密码要么复杂并且容易遗忘、要么简单容易被猜测出来，导致帐户被非法提款。如果我们可以识别出人脸，那么，我们就可以判断出在 ATM 机前的人是否是持卡者本人。这样即使信用卡被盗窃也无需担心，因为唯一的密码就是持卡者本人。银行和其他安全机构对此极为感兴趣。

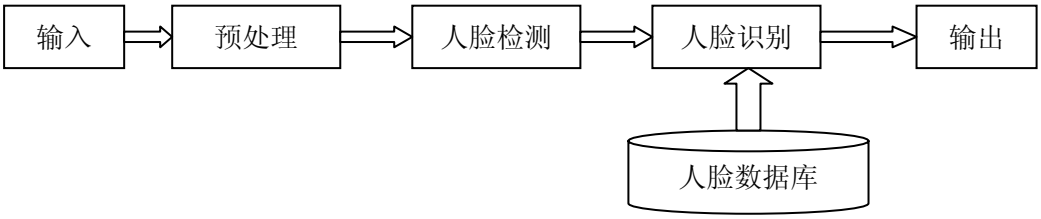
人脸识别作为模式识别、图像分析和理解领域的一个长期的研究热点，除了社会上的巨大需求之外，还有其他几个原因：

- 其他的生物特征相比，例如指纹、虹膜和 DNA，不需要被测试者主动配合。能做到主动识别，适用于各种应用环境。
- 图像样本采集自然并且容易，不需要复杂设备。已经有成熟产品。
- 和声音特征比较，人脸特征更加丰富。
- 大量的神经科学研究表明，人脸识别在灵长类动物的大脑皮层中有相应

的位置和非常重要的地位。在计算机领域研究人脸识别，对人们理解人脑的工作方式、研究人工智能都有非常重要的意义。

基于上述理由，我的毕业论文将人脸识别作为主要研究方向。其中，一个完善的现代人脸识别系统，必不可少的环节就是人脸检测。

简单来说，所谓人脸的检测（定位），就是在静态图像或视频(动态图像)中标出人脸所在的位置，把人脸选取出来。而人脸的识别就是把选取出来的人脸与数据库中已有的人脸进行比较，找出匹配的档案来。有的文献把人脸的定位和识别统称为人脸识别，但定位和识别则是两个主要的步骤。完整的人脸识别系统涉及到决定照片或视频中有无人脸，并计数，定位，定出大小，然后根据数据库识别出个人，可能的话还要识别表情，以及根据脸的图像做出描述（瓜子脸，丹凤眼等等就是日常生活中“描述”的例子），或者反过来根据描述挑选匹配的人脸图像。一个典型的流程如下图：



（图 1.1 人脸识别）

人脸识别系统虽然有诱人的应用前景，但是在现实中却还没有开始大规模的应用。其主要原因之一就是用计算机自动进行人脸的识别十分困难，目前的识别效果（检测率、错检率和性能）不如其他的生物识别技术，如指纹识别，虹膜识别等。人类在日常生活中就进行了大量的人脸识别工作，当然全部是由人的视觉系统和大脑“自动”进行的。目前还不清楚人的视觉系统和大脑的工作原理，因此这项人可以轻而易举完成的任务，对于目前还只会死板地执行程序指令的计算机来说却是极端困难。困难主要存在于两个方面：

1．人脸的图像数据具有高度的随机性。光照条件，脸的偏向，表情，发型，胡子，化妆，衣饰（眼镜，帽子）等等略有变化，就可以给识别系统带来巨大的困难。

2．人脸的图像数据量巨大。目前出于计算量的考虑，人脸定位和识别算法研究大多使用尺寸很小的灰度图像。一张 64*64 像素的 256 级灰度图像就有 4096

个数据，每个数据有 256 种可能的取值。定位和识别算法一般都很复杂，在人脸库较大的情况下，计算量十分大，很多情况下速度令人难以忍受。而灰度数据事实上是丧失了象色彩，运动等等的有用信息的。如果要使用全部的有用信息，计算量就更大了。

1.2 人脸检测

人脸检测是人脸识别的第一步。只有在图像中首先定位人脸，才能进一步进行模式识别。人脸检测作为人脸信息处理中的一项关键技术，近年来成为模式识别与计算机视觉领域内一个受到普遍重视、研究十分活跃的课题。

人脸检测之所以称为人脸识别的一个关键环节，是因为早期的人脸识别研究主要针对具有较强约束条件的人脸图像(如无背景的图像) ，往往假设人脸位置已知或很容易获得，因此人脸检测问题并未受到重视。但是，近几年随着电子商务等应用的发展，人脸识别成为最有潜力的生物身份验证手段，这种应用背景要求自动人脸识别系统能够对复杂背景中的人脸具有一定的适应能力。由此所面临的一系列问题使得人脸检测开始作为一个独立的课题受到研究者的重视.。今天,人脸检测的应用背景已经远远超出了人脸识别系统的范畴，在基于内容的检索、数字视频处理、人机交互、视频监测等方面有着重要的应用价值。

从本质上说，人脸检测是一个 2 类的分类问题。它的基本思路是先用统计或知识的方法对人脸建模，然后比较所有可能的待检测区域与已建立模型的相似度，从而得到人脸存在的可能区域。

相应人脸识别则是 n 类的分类问题（n 为图像数据库中人的个数而不是人脸的个数）。其目标是将检测出来的人脸在数据库中搜索，看是否在人脸数据库中，具体是那个人。

人脸识别和人脸检测面对的问题不一样，考虑问题的重点也不一样，解决的方法也不一样。例如：人脸识别强调尽量将 n 类之间的距离加大（如 Fisher 准则），而人脸检测由于只有 2 类，可以不必考虑类似的技术，关键是精确的划分 2 个类。再如：人脸识别必须考虑人脸图像数据库的人脸数据量巨大的问题，而人脸检测则偏重于人脸的建模。

在了解了人脸检测和人脸识别的关系后，再看看目前人脸检测算法的分类。还有的学者[14]将人脸检测从技术角度分为下面几类：

（1）基于知识的方法。

利用先验知识，将典型的人脸相关知识编码，通常是面部特征之间的的相互关系，建立若干规则，通过规则指导完成人脸检测，这里的规则包括人脸器官对称性和非对称性分布、轮廓形状、颜色明暗、纹理粗细和运动方向等。例如文献[13]。基于知识的方法通常都是自上而下的。

（2）利用特征不变性的方法。

算法的目标是寻找人脸的“结构性特征”，例如面部的边缘特征、纹理特征、肤色、以及上述多特征的综合。基于特征的方法通常都是自下而上的。

（3）模板匹配的方法。

保存几个标准的模式，来描述人脸或者面部特征。计算输入图像和保存的模板之间的相关度。例如预先存储的人脸模板和变形模板。

（4）基于外观（或象素）的方法

和模板匹配相反，通过一个训练集合的学习，基于统计的方法从二维人脸图像中提取出高维特征向量，得到一个模型，把人脸检测问题转化为高维空间信号的分布概率问题。再使用这些模型来检测人脸。例如特征脸[2]、神经网络[2]、支持向量机[15]、基于多高斯概率分布等方法。

1.3 几个术语

检测率（detection rate），就是检测器（detector）在一副图像中检测出的人脸的个数和通过“人”检测出的人脸的个数的比值。

检测器常犯的几个错误。一个是**漏检**（false negatives），也就是人脸被遗漏，这会降低检测率。另外一个是**错检**（false positives），也就是检测器检测为人脸，但实际上却不是。一个优秀的人脸检测算法，就是提高检测率、避免漏检和错检。而一般情况下提高了检测率、错检率也会提高。

对任何目标，都可以将之分为两类：一类是**正例**（Positive），属于人脸；一类是**反例**（Negative），不属于人脸。

1.4 相关工作

本论文的目标是快速人脸检测系统的设计和实现。由于人脸检测是人脸识别的第一步，只有快速的计算，才能为后续人脸识别留下更多的 CPU 计算资源。

为了实现实时性或快速人脸检测，有很多可行的策略和方法。如利用人脸肤色模型及运动模型减少搜索区域，提高检测速度[44]，利用粗检测和层次化局部搜索技术，并基于区域特征的快速人脸检测[45]，使用双眼模板搜索候选人脸的多关联模板匹配的人脸检测[46]，采用几何投影方法的人脸粗定位和快速检测[50]，线性和非线性层次型 SVM 方法[51]，利用瞳孔定位和人脸结构的可检测平面旋转人脸的方法[53]，利用人脸几何特征的人脸定位[60]。以上文献都有一个共同的规律，那就是要实现快速而准确的人脸检测，肤色模型、启发式搜索、分层次搜索（人脸粗定位）是很重要的策略。本文引入了这些被证明是成功的策略，利用肤色特征和几何特征来缩小搜索空间，具体实现方法与上述文献却不相同。

还有文献对运动中的人脸检测感兴趣，如基于差分图象的人脸检测算法[55]，利用人的头肩轮廓特点进行运动的人脸定位的方法[56]，图像序列中将包含人脸的前景与背景分开的方法[57]。和上述研究类似，本文也就图像序列中人脸的分割进行了探索， 提出了基于混合高斯背景模型[64][70]、阴影检测[67]和头肩几何特征的快速人脸粗定位技术，具体见第 8 章。

为了提高人脸检测的准确率，人脸检测算法中大都是有基于各种模式识别技术的对“人脸”和“非人脸”进行分类的步骤。如人工神经网络方法[32][49]，特征脸（主分量分析）方法[2]，基于 AdaBoost 学习的方法[6]，针对人脸检测方法中分类器训练困难和检测计算量大等问题的层次型支持向量机的人脸检测方法[51]，采用多视角人脸图像对隐马尔可夫模型训练得到状态图后进行人脸检测的方法[52]。本文的核心算法是 Viola 提出的基于 haar 特征的人脸分类和瀑布式识别算法[6]。在同一个测试数据库下，该算法在检测率和漏检率等方面与其他方法比较，基本保持不变，而运算复杂度极大降低，具体见第 5 章。

不过，当我们采集了一些实验样本后，我们发现 Haar 特征人脸检测算法虽然可以轻松的达到实时处理，但在复杂背景中的检测率、错检率和性能并不理想。所以，本文综合利用人脸的多种特征和多种方法，对 Viola 的算法进行了改进。试验证明这些改进是成功的，具体见第 7 章。

此外，本文还构建一个用于实时的快速的人脸检测的应用系统。利用 Windows 平台下最快速的视频处理技术－DirectX[45][46]，设计了一个从视频文件或者采集卡（USB 摄像头）读出数据，然后进行处理，最后将结果输出的机

器视觉开发和实验平台。系统的实现细节见第 7 章。

1.5 本文的工作

- 设计并实现了一个基于 DirectX 技术的机器视觉平台，可以处理视频文件和采集设备得到的视频流，并将结果输出到屏幕或者视频文件；
- 在 HSV 颜色空间中，利用肤色模型得到得到肤色概率图，并进行人脸粗定位；
- 利用形态学算子、最外轮廓、轮廓面积、长宽比等几何特征，排除非人脸区域，获得候选人脸区域；
- 利用 Canny 边缘检测图，进行启发式搜索，排除“假脸”；
- 提出基于背景模型和头肩几何特征的人脸粗定位技术；

本文围绕人脸检测，分章节进行了讨论。其中包括人脸检测各种常见算法（第 4 章），本论文构件的基于 DirectX 的试验平台（第 7 章），快速人脸检测的核心算法（第 5 章），启发式搜索策略（第 6 章），基于肤色模型的人脸粗定位算法（第 2、3 章），基于背景模型的人脸粗定位算法（第 8 章）。最后，通过试验证明，在启发式搜索和人脸粗定位策略下，人脸检测更加有效和快速。

第二章 肤色模型

每到春天，出外踏青，我们常常不由自主的赞叹大自然的色彩斑斓、五颜六色。“红花还需绿叶配”，我们在一片绿色中看到一点红，自然想到的，这一点红肯定是花。其实，人脸也有着特殊的颜色。快速人脸检测，就从颜色开始。

2.1 颜色理论

肤色是人脸的重要信息，不依赖于面部的细节特征，对于旋转、表情等变化情况都能适应，具有相对的稳定性并且和大多数背景物体的颜色相区别。这种天然而理想的特征对快速人脸检测非常重要。所以，先来认识颜色理论。

颜色是外界光刺激作用于人的视觉器官而产生的主观感觉。所以颜色特性既可以从客观刺激方面来衡量，也可以从观察者的主观感觉方面来描述。描述客观刺激的概念是心理物理学概念；描述观察者主观感觉的概念是心理学概念。确定光的心理物理量与心理量的关系是感觉心理学研究的重要任务。颜色视觉有三种特性，描述颜色的心理物理量是**亮度**、**主波长**和**纯度**，相应的心理量是**明度**，**色调**和**饱和度**。

颜色分两大类：非彩色和彩色。非彩色是指黑色、白色和介于这两者之间深浅不同的灰色。它们可以排成一个系列，由白色逐渐到浅灰、中灰、深灰直到黑色。这叫白黑系列或无色系列。白黑系列由白到黑的变化可以用一条直线代表，一端是纯白，另一端是纯黑。中间有着各种不同等级的灰色过渡。所谓灰色是相对的，比周围明亮的称为浅灰，比周围暗的称为深灰，灰色是最不饱和色之一。所谓纯白和纯黑也是相对而言的，并无绝对的标准，白雪接近纯白，黑绒接近纯黑，由白和黑按不同比例混合可得出各种灰色。白色和各种灰色是物体表面没有选择性的反射。白黑系列的非彩色的反射率代表物体的明度。反射率越高时接近白色，反射率低时接近黑色。一张洁白的纸的反射率可达 85%以上。用来测量颜色、定标用的标准白板的反射率可达 90%以上。一张黑纸的反射率可低至 5%以下，黑色天鹅绒的反射率甚至可低于 0.05%。

表示光的强度的心理物理学概念是**亮度**（Luminosity）。所有的光，不论是什么颜色都可以用亮度来测量。非彩色的白黑变化相应于白光的亮度变化。当白

光的亮度非常高时，人眼就感觉到是白色的；当光的亮度很低时，就感觉到发暗或发灰；无光时是黑色的。与亮度相应的心理学概念是**明度**（brightness）。明度是人眼对物体的明亮感觉，受视觉感受性和过去经验的影响。通常明度的变化相应于亮度的变化。通常，物体表面或光源的亮度越高，人感觉到的明度就越高。

彩色系列或有色系列是指除了白黑系列以外的各种颜色。我们通常所说的颜色即指彩色。彩色的第一个特性是用心理物理量亮度和心理量明度来表示的。所有的光，不论是什么颜色都可以用光的亮度来定量。与非彩色相似，彩色光的亮度越高，人眼就感觉明亮，或者说有较高的明度。彩色物体表面的反射率越高，它的明度就越高。

表示彩色的第二个特性的心理物理学概念是**主波长**（dominant wavelength）。与主波长相应的心理学概念是**色调**（hue）。光谱是由不同波长的光组成的。用三棱镜可以把日光分解成光谱上不同波长的光，不同波长所引起的不同感觉就是色调。例如，700 纳米波长光的色调是红色，579 纳米波长光的色调是黄色，500 纳米光的色调是绿色等。若将几种主波长不同的光按适当的比例加以混合，则能产生不具有任何色调的感觉，也就是白色。事实上，只选择两种主波长不同的光以适当的比例加以混合也能产生白色。这样的一对主波长的光叫做互补波长。例如，600 纳米的橙色和 492 纳米的蓝绿色是一对互补波长；575.5 纳米的黄色和 474.5 纳米的蓝色也是互补波长。一对互补波长的色调叫互补色。光源的色调决定于人眼对辐射光的光谱组成产生的感觉。物体的色调决定于光源的光谱组成和物体表面反射（透射）的各波长的比例对人眼产生的感觉。例如，在日光下，一个物体反射 480-560 纳米波段的辐射，而相对吸收其它波长的辐射，那么该物体表面为绿色。

表示彩色第三个特性的心理物理学概念是颜色**纯度**（purity），其相应的心理学概念是**饱和度**（saturation）。纯色是指没有混入白色的窄带单色光。在视觉上就是高饱和度的颜色。可见光谱的各种单色光是最饱和的彩色。当光谱色搀入白光成分越多时，就越不饱和。例如，主波长为 650 纳米的光是非常纯的红光。如果把一定数量的白光加到这个红光上，混合的结果便产生粉红色。加入的白光越多，混合后的颜色光就越不纯，看起来就越不饱和。

光刺激的心理物理特性可以按亮度，主波长和纯度来确定。这些特性又分别

同明度、色调、饱和度的主观感觉相联系。颜色可分为彩色和非彩色。光刺激如果没有主波长，这个光就是非彩色的白光，它没有纯度。然而所有视觉刺激都有亮度特性。亮度是彩色刺激和非彩色刺激的共同特性，而主波长和纯度表示刺激是彩色的。

早期的机器视觉系统，由于受采集设备和计算设备的限制，大多都使用灰度图像，也就是之处理只有黑白区分的图像。随着技术的发展，逐渐开始利用各种彩色信息。出现了在计算机上使用的各种颜色模型。

2.2 颜色模型

面对自然界丰富多彩的颜色，如何进行建模，一直是人类研究的课题。最早可以追溯到十七世纪后期牛顿的工作。他用棱镜把太阳光分散成光谱上的颜色光带。牛顿通过实验证明了：（1）白光是由很多不同颜色的光混合而成的结果；（2）作为白光成分的单色光具有不同的折射度；（3）光谱上的两种颜色相混合会出现一种新的颜色。

1807 年 Young 提出了红、绿、蓝三种原色以不同比例混合可以产生白色和其它各种颜色的假设。这个假设为以后的颜色混合实验所证实。在此基础上 1862 年 Helmholtz 提出了一个颜色视觉的生理学理论。他假设在人眼内有三种基本的颜色视觉感觉纤维，后来发现这些假设的纤维和视网膜的锥体细胞的作用相类似。所以近代的三色理论认为三种颜色感觉纤维实际上是视网膜的三种锥体细胞。每一种锥体细胞包含一种色素，三种锥体细胞色素的光吸收特性不同，所以在光照射下它们吸收和反射不同的光波。

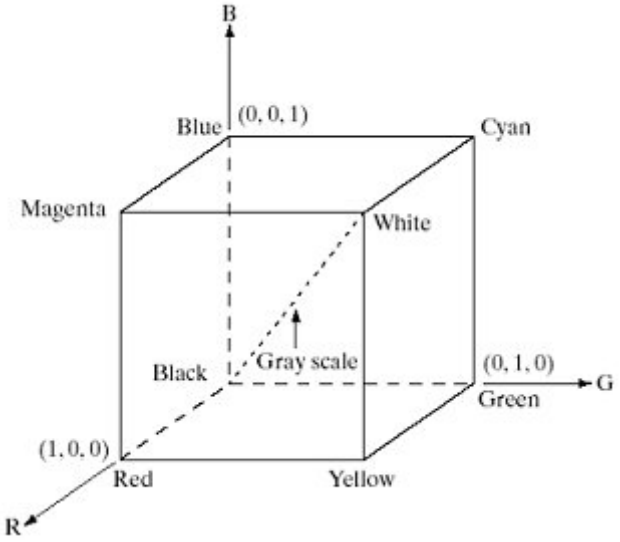
三色理论最后形成了一个被人类广泛使用的颜色模型：

2.2.1 RGB 颜色模型

国际照明委员会（CIE）所制定的 RGB 颜色表示系统选择红、绿、蓝三种单色光作为表色系统的三基色。任何颜色都可以由这三基色混配得到。规定三基色的波长为：红色（RED）：λ=700.00nm；绿色（GREEN）：λ=546.1nm；蓝色（BLUE）：λ=435.8nm。

颜色空间示意图如下，对角线代表的是由白色到黑色，3 个坐标分别代表

RGB 三基色。



（图 2.2.1 RGB 颜色模型）

计算机系统使用的大多数是 RGB 颜色系统，我们从摄像机得到的彩色图像，在显示器上看到的图像，默认就是使用 RGB 颜色模型的数据。

2.2.3 YIQ 颜色模型

YIQ 颜色模型的初衷是用于电视广播。早期的电视是黑白电视，在出现彩色电视后，如何保持兼容性，使黑白电视也能接受彩色电视信号。出现了 YIQ 颜色模型。RGB 到 YIQ 空间的变换可以按照下面三式进行：

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & -0.311 \end{bmatrix} \begin{bmatrix} \mathrm{Re}\,d \\ Green \\ Blue \end{bmatrix}$$

Y 分量表示亮度，它占用了相当大部分的带宽。I 和 Q 都是颜色分量，其中，I 表示橙~青颜色，Q 表示其它部分颜色。这样，在 YIQ 空间中，亮度和颜色分离开了。亮度和颜色的分离，使得在处理图像的亮度成份时不致影响颜色分量。黑白电视机就只利用了 Y 分量。再如，如果要将一幅彩色图像去彩色以得到灰度图，可以对每个象素点作如下操作：提取出该点的 red、green、blue 值，根据上式得到 Y 值，再令该点的 red、green、blue 分量都为 Y。这样，彩色图像就变成了灰度图像。

2.2.4 CMY 颜色模型

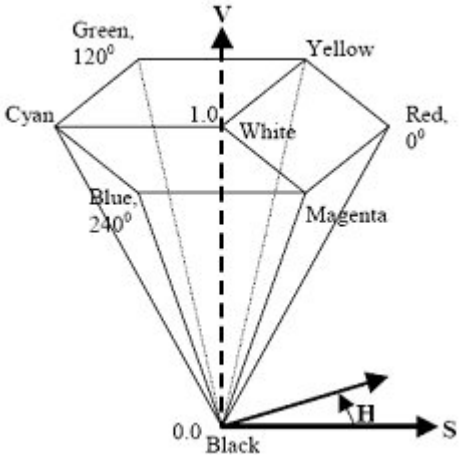
该模型用于打印输出，比如彩色打印机和复印机。RGB 到 CMY 空间的变换可按下面式子进行：

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

这里所有的颜色值都已被归一化到[0,1]范围内。

2.2.5 HSV 颜色模型

顾名思义，HSV 颜色模型有三个分量 H（hue，即色度）、S（saturation，即饱和度）和 V（intensity，即亮度），且这三个分量是正交的。因此，HSV 模型是在图像处理中最重要的模型。



（图 2.2.5.1 HSV 颜色空间）

RGB 向 HSV 的变换公式：

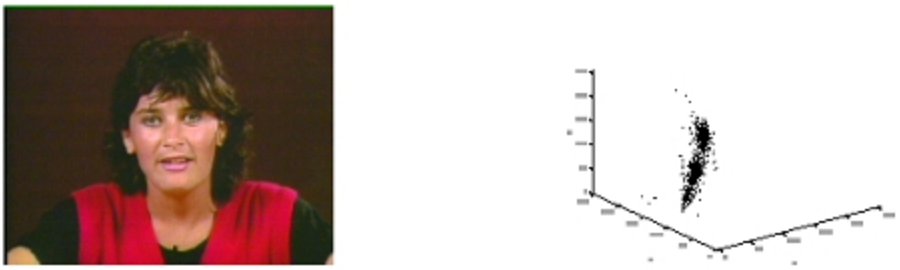
$$V = \frac{1}{3}(R+G+B)$$

$$S = 1 - 3 * \frac{Min\{R,G,B\}}{R+G+B}$$

$$TempH = \arccos\left\{ \frac{(R-G)+(R-B)}{2*\sqrt{(R-G)^2+(R-B)*(G-B)}} \right\}$$

如果 B≤G，H=TempH 否则，H=2π - TempH

回到我们的论文的主题。我们想在图像中检测出人脸，人脸的肤色是分割人脸和环境的一个非常重要的特征。如果我们可以首先大致的分割出人脸区域，将大大缩小后续算法的搜索区域，因此降低计算复杂度。我们同时发现，使用前面的 RGB 等模型，肤色的概率分布函数规律性不强，或者说可操作性不强。



（图 2.2.5.1 RGB 空间的肤色概率分布）

因此，我们的系统没有使用 RGB 模型，而是 HSV 模型。并且，只使用 HSV 颜色空间中的色调分量 H。检测肤色的时候只使用色调 H 可以吗？回答是肯定的。（1）亮度信息 V 对于寻找肤色没有帮助，抛弃亮度分量还可以消除环境光线强弱的影响。（2）饱和度 S 起的作用也不大。我们的试验证明，肤色较浅的人和肤色较深的人，在色度 H 这个分量上，几乎没有差别，主要差别是肤色较深的人的饱和度分量 S 比较高。（3）只使用色调 H 可以降低检测所需的计算量。所以，只使用 HSV 颜色空间中色度分量 H 的信息，不但可行，而且还有助于消除环境光线和肤色深浅的影响，并且会降低计算量。

在实验中，我们发现，采集到的 RGB 图像使用 HSV 空间时有一些问题。首先是，太低亮度和太高亮度的象素点（例如 V 和 S 分量接近于 0），会导致色度 H 分量计算异常，取值大幅度变化。所以，我们在系统中过滤掉这些象素点。使用的函数是 Color2HSV_8U1C_4img()，具体请见附录。

更加进一步的关于颜色模型对人脸检测的影响的讨论见文献[16]。

2.3 肤色模型及检测

肤色是人脸的重要信息，不依赖于面部的细节特征，对于旋转，表情等变化情况都能适用，具有相对的稳定性并且和大多数背景物体的颜色相区别。因此肤色特征在人脸检测中是最常用的一种特征，在人脸特征定位中也经常作为图像预

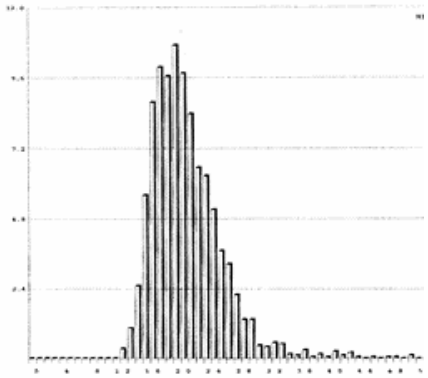
处理的方法。肤色特征主要由肤色模型描述[59]。

建立肤色模型，可以使用各种方法。常用的肤色模型有高斯模型[34]、混合高斯模型[35][36]和直方图模型[20]。除这三种肤色模型外，还有直接利用几何参数描述肤色区域分布范围的模型[37]、三维投影模型[38]、基于神经网络的肤色模型[39] [40]等。此外也有同时考虑“肤色”与“非肤色”象素分布的基于贝叶斯方法的模型[41]。

文献[20]比较了高斯模型和混合高斯模型在不同色度空间中的性能,发现除了少数情况外,一般需要使用混合高斯模型才能较好地描述肤色区域的分布. Terrillon 等同时指出,最终限制检测性能的因素是不同色度空间中“肤色”与“非肤色”区域的重叠程度。

Jones 等[21]研究了 RGB 空间中“肤色”与“非肤色”像素的分布，根据标定出肤色区域的近 2 万幅图片(包含约 20 亿个像素)建立了三维直方图，在此基础上比较了直方图模型和混合高斯模型，发现前者的性能略好于后者。因此，本文主要采用基于直方图的方法，下面是具体实现细节。

为了更加快速的检测出包含肤色的象素，在不损失正确率的前提下，我们先将采集到的 RGB 分量映射到 HSV 颜色空间。对其中的色调 H 分量。建立一维的直方图，类似下图所示：

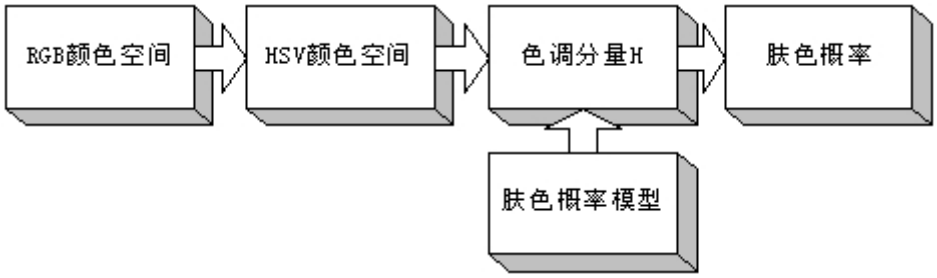


（图 2.3.1 H 分量的肤色分布）

对于含 N 个人脸的训练样本，假设任意训练样本中的人脸区域大小为 $m*n$，对每个样本，得到人脸样本色调分量H的 180 个bin的直方图$h_{i=(}h_{i1},h_{i2},\ldots,h_{i128),}$其中 $h_{ij}(1\le i\le N,1\le j\le 180)$ 表示第i个训练样本的第j个bin的 H 值。对于第 i 个训练人脸的第j个 bin，用如下的高斯分布表示其分布概率：

$$f_j(h_{ij}) = \frac{1}{\sqrt{2\pi}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(h_{ij}-\mu_j)'\Sigma_j^{-1}(h_{ij}-\mu_j)} \quad (1 \le i \le N, 1 \le j \le 180)$$

其中，Σ_j 和 μ_j 分别表示所有训练人脸图象第 j 个 *bin* 的方差和均值。对于从视频流中的每个采样点，计算 $sample = \sum_{j=1}^{128} f_j(h_{ij})$。*sample* 就是肤色出现的概率。肤色检测的流程如下图所示。



（图 2.3.2 肤色概率框图）

我们系统中肤色概率的例子如下图所示：

（图 2.3.3 肤色概率）

右图是原始的图像，左图是经过肤色检测之后得到的肤色概率图像。左图中的象素点亮度就是人脸肤色的概率。亮度越大，该象素符合肤色模型的概率越高；亮度越小，是肤色的概率越低。可以看到，大部分背景区域被过滤，和肤色模型符合的象素点被凸出。

我们期望得到的肤色区域，是能够尽可能多的分割出包含人脸的所有区域，和绝大多数的非人脸区域。但是，如上图所示，在实际应用中肤色模型得到的区

域连续性非常差，个别样本点的缺失严重影响了人脸和后续的人脸检测。显然，我们无法直接使用肤色概率图。我们必须进一步进行滤波。

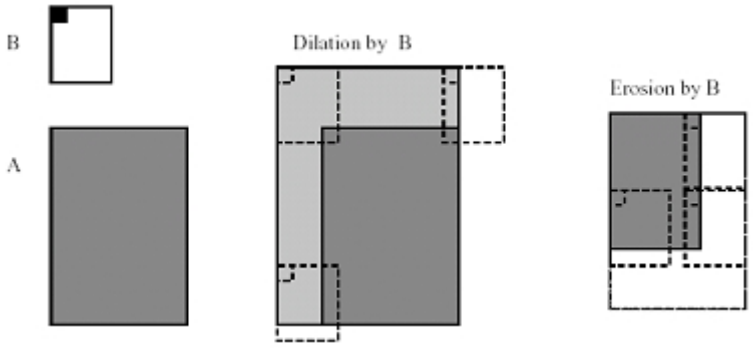
第三章 连通区域

要得到边缘平滑并且连通的区域，有很多方法。越是准确和鲁棒的方法，计算复杂度越高，为了达到人脸检测系统的实时性，经过多种方法对比研究，我们最后使用的方案是。首先，对人脸概率图进行二值化，然后，利用形态学算子，对肤色概率图进行形态学运算，最后，利用轮廓的包含关系以及大小等信息，进行轮廓过滤。

3.1 形态学算子

数学形态学是集合理论在图象分析上的一种应用，首先由 Matheron 和 Serra 于 1982 年提出来[17]。它具有非常严格的数学推导和证明。所有的操作都包含：图像 A，叫做**目标**,以及使用一个核元素 B，叫做**结构元素**。图像和结构元可以是任何维数，但是通常使用于二维的二值图像，或者三维的灰度图。

结构元素 B，常常是正方形或者圆形，以及其他任何形状。正如和图像卷积相同，B 是一个核或者模板，具有锚点。如下图所示，结构元 B 的锚点是黑色的小正方形。



（图 3.1.1 形态学算子）

上图中左边是目标和结构元素，中间和右边是两个基本的形态学操作：腐蚀 Erosioin(细化)，以及膨胀 Dilation(加粗)。A 经过 B 膨胀后扩大到了灰色区域，A 经过 B 腐蚀后缩小。形态学算子的数学定义如下：

腐蚀： $A\Theta B = \{t : B_t \cap A \neq 0\}$

膨胀： $A \oplus B = \{t : B_t \subseteq A\}$

继续扩展一下，$A \oplus nB$表示膨胀 n 次，$A \Theta nB$表示腐蚀 n 次。形态学上的开(Open)运算和闭(Close)运算，实质上是上面两个算子的组合：

开： $A \circ B = (A \Theta nB) \oplus nB$

闭： $A \bullet B = (A \oplus nB) \Theta nB$

在图像处理和分析中，由于处理的都是数字图像，设 A(x,y)，B(i,j)均为二值图像，于是公式等价为，其他公式以此类推：

腐蚀 $E(x,y) = (A \Theta B)(x,y) = \underset{i,j=0}{\overset{m}{AND}}[A(x+i, y+j) \,\&\, B(i,j)]$

膨胀 $D(x,y) = (A \oplus B)(x,y) = \underset{i,j=0}{\overset{m}{OR}}[A(x+i, y+j) \,\&\, B(i,j)]$

数字形态学算子常常用来边界检测、过滤噪声、分割或者合并图像区域。我们用它来消除大多数的小“气泡”，也就是面积较小的不连续区域，平滑轮廓的边界。下图是形态学运算前后的肤色填充图片。

（图 3.1.2 形态学运算前）

（图 3.1.3 形态学运算后）

图 3.1.3 左图是闭运算的结果，右图是开运算的结果。可以看到，原始图像中播音员的黄色毛衣部分和染发部分，严重干扰了肤色检测，出现了大量的“假

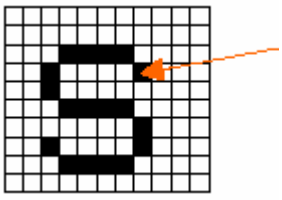
肤色”点。而闭运算虽然平滑了边界，但是使假肤色区域继续扩大，而显然只有开运算不但消除了小“气泡”区域，而且很好的覆盖了人脸的关键区域。

对比上图，可以看到，很小的不连续区域都被形态学算子“抚平”。我们得到了基本能够覆盖所有人脸区域的区域。并且边界平滑。但是对于某些较大的“气泡”区域，例如干扰噪声区域、人离摄像头很近时的眉毛区域，简单的形态学算子无法达到消除连通区域中的闭合非连通区域的目的，因此要使用轮廓技术。

3.2 获得最外轮廓

获得图像的轮廓是图像处理的一个重要技术。获得轮廓的方法一般是：首先，使用某种图像分割算法，将感兴趣的目标和背景分割出来，我们使用的基于颜色模型的肤色分割。然后，将这些目标赋值为 1，背景赋值为 0，也就是二值化。然后将得到的二值图进行轮廓搜索。搜索算法有很多种，我们使用的是 Suzuki 提出的轮廓追踪算法[18]。

搜索得到的轮廓使用链码 chain 表示，并且还有一个外接矩形 rect 表示轮廓的长和宽。如下图是一条线条的链码表示：



起始点
对应链码：34445670007654443

（图 3.2.1 轮廓链码表示）

在寻找轮廓的过程中，由于人脸的复杂、光照的多变、形态学算子的自身原因，不可避免有一些“气泡”区域。。如下图：

（图 3.2.2 轮廓分析例子）

区域 W2 没有内含轮廓，而 W1 包含 B2 和 B3，W3 包含 B4 等。由于肤色

已经作为前景（白色区域），非肤色点作为背景（黑色区域）。搜索所有的连通区域的轮廓，如果轮廓存在包含关系的，将内层轮廓从轮廓表中删除，最后保留最外层的轮廓。

（图 3.2.3 所有轮廓例子）

上图是通过轮廓搜索算法得到的所有轮廓图，女主播一小块黑头发被形态学算子漏掉，形成面部的漏洞。所以，我们必须进行轮廓的筛选。寻找具有包含关系的轮廓，只保留最外层的轮廓。

3.3 进一步轮廓过滤

在排除相互存在包含关系的轮廓后，还必须进一步轮廓过滤。为了便于解释，下图是将轮廓填充成单一颜色的填充图：

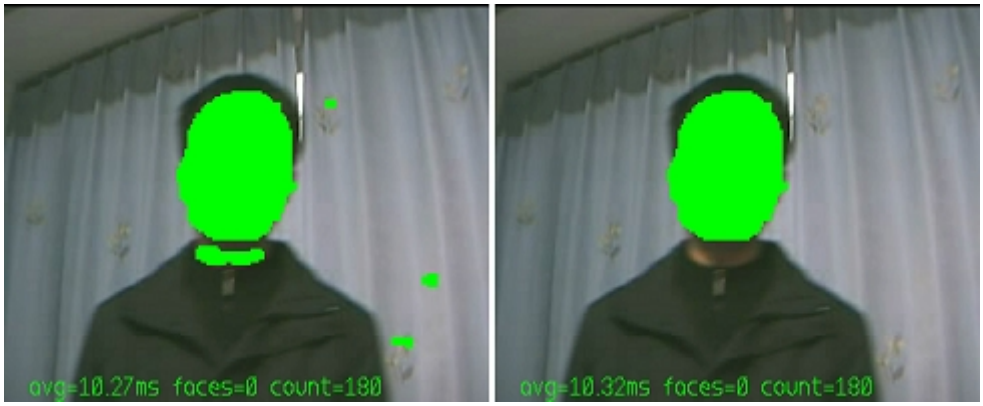
（图 3.3.1 轮廓过滤前后）

左图是没有进行轮廓过滤的，有大量的小面积轮廓。而右图经过了轮廓过滤，某些轮廓被过滤掉。例如图中的噪声，背景中的人脸肤色干扰，人手的干扰等。

轮廓过滤的第一条规则，是轮廓的外接矩形 rect 的面积必须小于 20*20。这样做的依据，是在图像中，人脸总是具有一定的面积，并且基于的 Haar 特征检

测算法的最小检测尺度是 20*20 的矩形区域，所以，我们可以将面积小于这个区域的所有轮廓都剔除。

轮廓过滤的第二条规则，是区域外接矩形的高度和宽度之比(w/h)应在某个合适范围内。太狭长和太扁平的区域（w/h 太小或太大）也不可能是人脸。如图 3.3.2 中，人物的下巴以下脖子部分的肤色被过滤：



（图 3.3.2 轮廓过滤例二）

经过轮廓过滤，我们可以将大多数虽然符合肤色模型，却非真正人脸的区域过滤掉。肤色过滤加上轮廓过滤，占用的 CPU 时钟并不高。可以从图中看到，在 P4 1.7G 机器上，运算需要占用 10 毫秒的 CPU 时间。

第四章 人脸检测算法

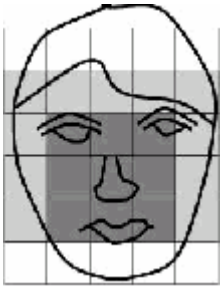
很多人脸检测系统、视频监视系统、图像（视频）检索系统、都利用肤色这个人类很重要的特征来进行模式匹配。例如，为了避免未成年人浏览成人网站，我们要在一大堆网上图片中，利用肤色搜索过滤包含成人网站内容的图片。但是，很显然，只使用基于肤色模型的方法来检测人脸是远远不够的。

很多人提出了很多人脸检测的方法。我们在第一章中对人脸检测算法做了分类，下面将回顾这些算法，特别是几类典型的方法。

4.1 基于知识的由上向下的方法

基于研究者对人脸的知识。很容易就通过人脸特征的关系得到一些简单的法则，例如，人脸有两个对称的眼睛，一个鼻子，一个嘴巴。它们之间的关系可以用它们之间的距离表示出来。对输入的图像，首先抽取面部特征，再得到候选的人脸，通过知识（规则）来检测，最后验证检测结果。

Guangzheng Yang 和 Huang[22]提出了一种基于镶嵌图(Mosaic Image)的人脸检测方法很典型。所谓镶嵌图就是将图像划分为一组大小相同的方格，每个方格的灰度为格中各个像素的平均值。该方法从人脸内部区域的灰度分布规律出发，构建了一个三层的基于知识的金字塔（hierarchical knowledge-based）结构的人脸检测系统。第一层先把图像转换成尺寸大小为 n*n 的镶嵌图，然后利用规则，如“人脸中心具有 4 个均一亮度值”，搜索可能的人脸区域。第二层则对第一级搜索出的可能人脸运用更加严格的规则，排除大部分的假脸。第三层则通过提取脸部的部件来做最后的验证。



（图 4.1 镶嵌图法）

他们的算法的一个很有趣的特征，就是由粗倒细和注意焦点的策略，这会降低计算量。虽然他们的方法检测率不高，但是对后来的工作起了很大的作用。

Kotropoulos 和 Pitas 的方法[23]。对水平和竖直方向进行投影，其中嘴唇，

鼻子和眼睛具有局部最小值。对于严格约束的背景，只有一个人脸的图像具有很好的检测效果。相反，对复杂背景和多个人脸效果很差。他们的投影的方法在某些场合很有用。

基于知识的方法在背景受约束的情况下可以检测出人脸，并且基于的规则都是人类很容易理解的。缺点是很多规则很难用计算机描述，复杂背景下效果很差。

4.2 自底向上的基于特征的方法

和自顶向下的方法相反，人们试图寻找人脸中不会变化的特征来进行检测。假设基于这样一个观察结果：人类自身可以检测不同光线和不同姿式的人脸的器官，因此，这些特征必然存在某些和上述变量无关的量。

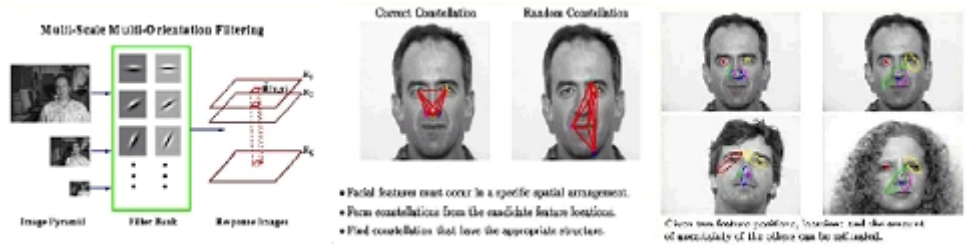
可以利用的人脸特征分为颜色特征和灰度特征两大类。其中肤色特征在第二章中已经论述，这里不再赘述。灰度特征包括亮度、边缘、纹理、轮廓等。

Sirohey 提出一种在均一背景下分割人脸的定位方法[24]。它使用 Canny 边缘检测，然后启发式的搜索，删除边缘以及将边缘归并成组，最后只有人脸轮廓被保留下来。然后一个椭圆形来匹配这个边缘。它的方法达到 80％的准确率。

Chetverikov 使用另外一种方法，就是使用“污点”（blob）和“条纹”streak[25]。它们的人脸模型由两个黑色的 “污点”，以及三个白色的“污点”组成，分别代表眼睛，颧骨和鼻子。然后使用“条纹”来表示人脸轮廓，眉毛和嘴唇。使用 2 个三角形关系来对这些污点的空间关系编码。使用低分辨率的 Laplacian 图像进行“污点”检测。然后扫描图像寻找三角形关系作为人脸候选，最后在候选人脸周围发现到“条纹”后，就是检测到人脸。

Yow 和 Cipolla 提出的基于特征的方法[26]。首先使用一阶高斯梯度滤波器，对滤波器的响应点进行检查和分类（例如检测边缘的长度、宽度、亮度等），将特征和数据库中的特征计算相关度，如果各个特征之间的 Mahalanobis 距离满足条件（使用 Bayes 神经网络），人脸检测成功。他的方法的特点是可以检测不同方向和不同姿式的人脸。

Leung 提出的随机图匹配方法[27]。它将人脸检测问题转换为“面部特征”的几何关系匹配问题。“面部特征”定义是多方向、多尺度高斯梯度算子的平均响应，一般是人眼、鼻子等。先对多个人脸特征之间的距离通过高斯分布进行学习，然后在被测试图像中，使用随机图匹配的方法寻找人脸。

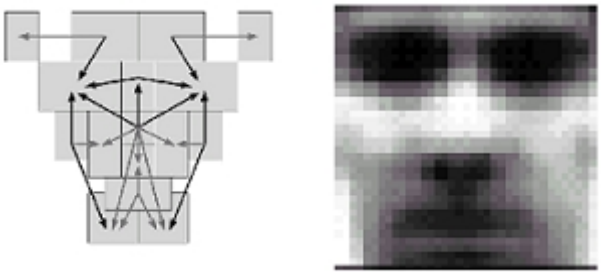


（图 4.2 面部特征）

基于特征的算法的优点是对人脸姿式和方向具有较好的鲁棒性，缺点是在光线、噪声等影响下会有非常大的误差。特征的边界可能非常微弱，阴影也可能造成很强的边缘。这些都会使得算法无效。而且，基于特征的算法无法在复杂背景下检测人脸，上述文献中大多要求均一的背景。

4.3 基于模板的方法

人们很早就开始使用模板匹配方法来尝试检测人脸。比较重要和典型的是 Sinha 提出的“比值(ratio)模板”的方法[28]。他的关键思想是，虽然面部特征的亮度变化很大，但是人脸特征之间亮度的“相对亮度”是基本不变的（例如眼睛总是是比周围的区域暗）。于是，他使用了如图中所示的“比值模板”，每块的面积是 14*16 象素，计算成对的亮度“比值”。



（图 4.3 人脸模板和例子）

Sinha 的这种思想后来被广泛的利用，有人也对这种思想进行了扩展，例如引入小波来检测汽车、人脸等。

基于模板的方法，最大的特点是简单好用，但是和基于知识的方法一样，缺点是对人脸姿式敏感，算法复杂度高。

4.4 基于外观的方法

其实，基于外观的方法，就是基于统计学的方法。一般流程是，首先在大量

的学习样本中，对人脸和非人脸样本进行学习，得出统计规律，也就得出了分类器。然后，对需要检测的图像进行某种预处理，再使用分类器进行搜索，最后将搜索结果返回。

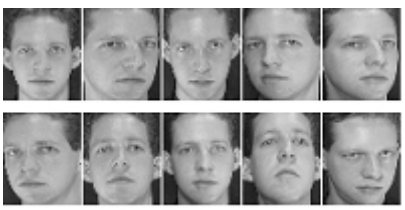
外观可以用亮度表示，也可以用边缘等其他方式来表示。分类器也可以选择很多种，例如混合高斯概率模型、神经网络、PCA（Principaple Component Analysis）、SVM（Support Vector Machine）、HMM（Hidden Markov Modal）等。

4.4.1 特征脸法

主分量分析本是统计学中用来分析数据的一种方法，它基于 KL 分解。最早将其用于人脸识别中的是 Turk 和 Pentland，并因为它的有效很快流行起来。简单地说，它的原理就是将一高维的向量，通过一个特殊的特征向量矩阵，投影到一个低维的向量空间中，表征为一个低维向量，并不会损失任何“有效”信息。也就是说，通过低维表征的向量和这个特征向量矩阵，可以完全重构出所对应的原来的高维向量。

一个应用特征向量的早期例子是 Kohonen[29]使用简单神经网络，并使用通过特征向量的近似人脸描述。后来，Kirby 和 Sirovich 证明人脸图像可以通过适度的“基图像”被线性编码[30]。

Turk 和 Pentland[2]第一个使用特征脸方法来对人脸进行识别和检测。它将特征图像叫做“特征脸”（Eigen Face），这些特征脸张开成为一个子空间。所有的人脸图像都投影到这个子空间，并且形成聚类。类似的，非人脸图像也投影到同一个子空间，也形成一个聚类。最后，被测试图像的每个位置的子图像都投影到特征子空间，使用最近邻法和神经网络来识别。

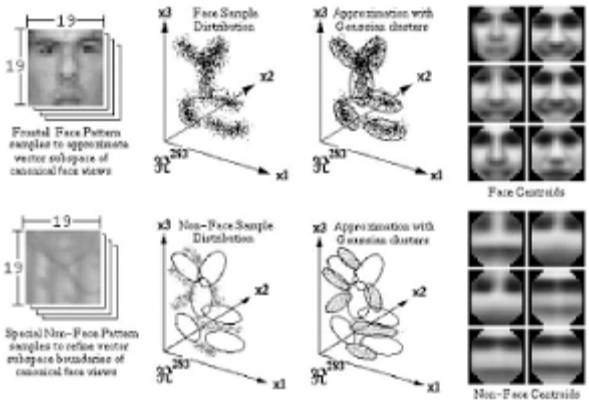


（图 4.4.1，人脸图库）

（图 4.4.2 平均脸和特征脸）

4.4.2 概率分布法

MIT 的 Sung 等[31]提出了基于事例学习的方法,，同时使用了 19×19 像素分辨率的“人脸”和“非人脸”样本。样本预处理后，采用 k－均值聚类方法建立 6 个“人脸”簇(Clusters)，同时建立包围“人脸”簇的 6 个“非人脸”簇，以使“人脸”与“非人脸”模式的边界更为清晰。Sung 等使用样本到各个簇中心的距离训练一个多层感知器进行分类。



（图 4.4.3 概率分布法）

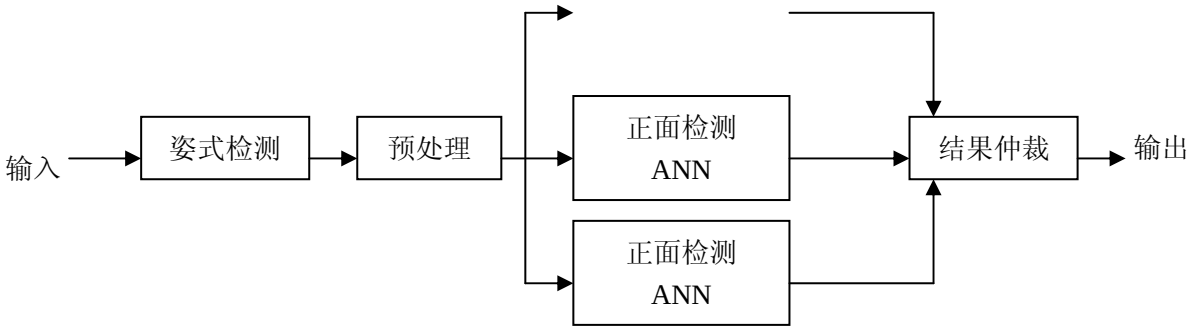
需要指出的是，人脸检测中“非人脸”样本的选取是一个较为困难的问题。Sung 等使用了“自举”(bootstrap)方法加以解决：首先建立一个仅使用“人脸”簇的初始分类器对一组图像进行检测，将所有的错误报警(不是人脸而被错检为“人脸”的结果) 加入“非人脸”样本库，构造新的使用“人脸”与“非人脸”簇的分类器重新检测。以上过程不断迭代，直到收集了足够的“非人脸”样本。

4.4.3 人工神经网络法

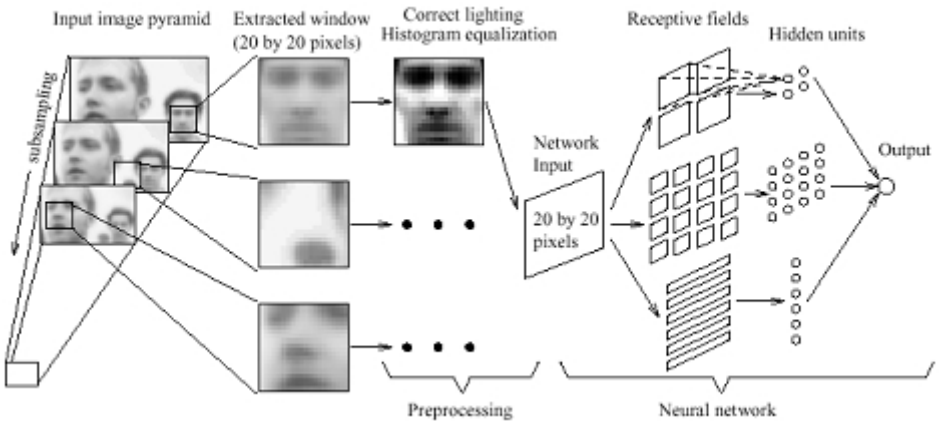
人工神经网(ANN)在模式识别的诸多领域获得了成功，例如字符识别、机器人自动驾驶等。它把模式的统计特性隐含在 ANN 的结构和参数之中，对于人脸这类复杂的、难以显式描述的模式，基于 ANN 的方法具有独特的优势。

最典型的是 CMU 的 Rowley[32]使用多个 ANN 来检测多姿态的人脸，算法

的框架如图 4 所示。图中显示了两类 ANN：1 个姿式检测器(poseestimator) 用于估计输入窗口中人脸的姿式、3 个检测器(detector)分别检测正面(frontal)、半侧面(halfprofile)和侧面(profile)的人脸。使用经过对准和预处理的“人脸”样本以及采用“自举”(bootstrap)方法收集分类器错分的样本作为“非人脸”样本训练各个 ANN，进一步修正分类器。检测时对输入图像中所有可能位置和尺度的区域首先使用位姿检测器估计人脸位姿，经校准和预处理后送入 3 个检测器中，最后对检测器的分类结果进行仲裁。



（图 4.4.4 框架）



（图 4.4.5 神经网络法）

4.5 小结

以上是上个世纪主要的人脸检测算法，虽然检测率等指标各有高低，但共同的缺点是计算复杂、性能低，很难做到实时检测。要想做到实时检测、最优秀的算法就是下一章将要介绍的基于 Haar 特征的人脸检测算法。

第五章 基于 Haar 特征的人脸检测

在浏览了几种经典的人脸检测算法后，回到本文使用的人脸检测算法上。2002 年 Viola 提出基于 Haar 特征的人脸检测算法[6]。它是最近几年被引用较多，较典型的人脸检测算法，具有检测速度快、鲁棒性好、实时检测等优点。本文在他的工作的基础上，进一步改进后，作为人脸检测系统的主要检测算法。简称 IHAAR（Improved Haar Feature Face Detection）。

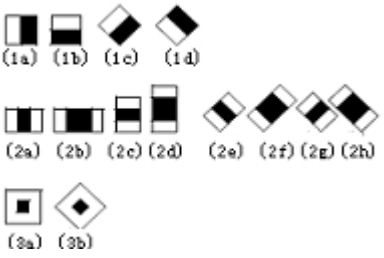
5.1 Haar 特征

IHAAR 算法是一种基于特征(feature)的算法，而不是基于象素的算法。由于算法面对的简单特征数量小于象素的数量。这种使用基于简单特征的的方法比基于象素的方法速度快。

正交的 Haar 特征族是 Papageorgiou 等人引入的[9] [10]. IHAAR 使用了其中三种类型的特征：（1）两矩形特征－两个矩形区域的象素总和的差值。（2）三矩形特征－两个外部矩形象素总和减去中间矩形象素总和。（3）四矩形特征－计算对角线矩形象素总和差值。

Rainer Lienhart 等人将 Haar 特征进行扩展[11]。引入了旋转了的 Haar 特征，扩展了目标识别的手段，并且扩展后的旋转 Haar 特征也可以利用积分图像在常数时间内进行计算。

在本论文的演示程序里可能使用的特征原型(prototype)如下图所示。图中的第一行是 2 矩形特征原型，代表图像中的边界信息，我们命名为 1a,1b,1c,1d，第二行是 3 矩形特征原型，我们命名为 2a,2b…，代表图像中的线条信息,第三行是 4 矩形特征原型，代表图像中的“包围”信息，我们命名为 3a,3b…。



（图 5.1.1 Haar 特征）

在文献[6]中使用的特征很少(1a)，(1b)，(2a)和一个 4 矩形特征。本论文的人脸检测系统中，使用了(1a)，(1b)，(2a)，(2c)，以及个别使用了(2e)和(2g)特征。虽然使用的特征数量很少，但是，每个特征由特征原型组合而成，可能组合非常多。可以使用下面的公式计算特征数量。

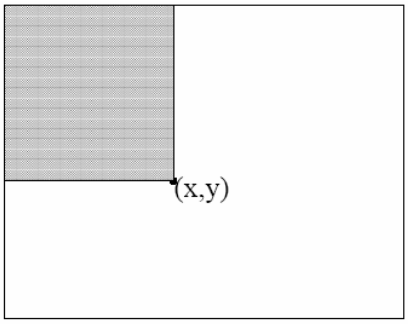
设 $X=\lfloor W/w \rfloor, Y=\lfloor H/h \rfloor$，W 是子图像的宽度，H 是子图像的高度，w 和 h 分别是 Haar 特征的宽度和高度。那么 X 和 Y 是在 x 和 y 两个方向上最大的放缩系数。特征原型的组合的特征数目是： $XY\cdot(W+1-w\frac{X+1}{2})\cdot(H+1-h\frac{Y+1}{2})$ 。对(1a)(1b)特征的组合就达到 43200 种。可能的特征数目是非常大的，如何迅速找到有效的特征将在 5.3 中叙述。

5.2 积分图像

Viola 提出的一个关键的思想，就是在图像处理初期就建立一个积分图像(Integral image)。积分图像是数字图像的一种表示方法，对(x,y)点处的象素值，代表所有左上角象素的总和。

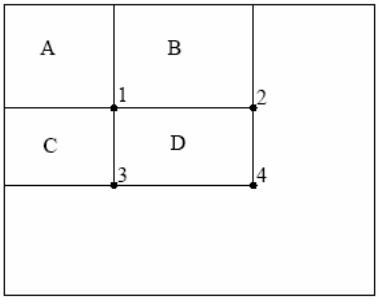
$$i(x,y)=\sum_{i<x,j<y} g(i,j)$$

其中 g(i,j)是原始的图像，i(x,y)是积分图像。如图，(x,y)点的数值等于左上角灰色区域的所有象素总和



（图 5.2.1 积分图像）

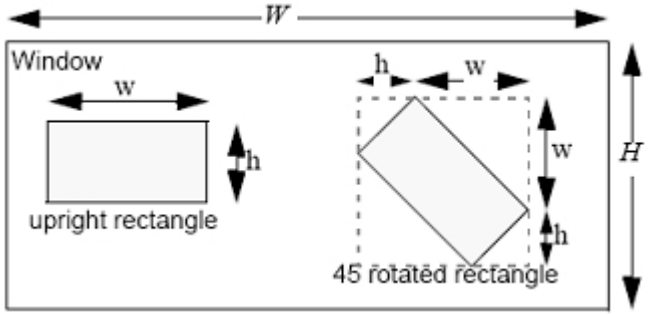
积分图像一旦建立，计算任意矩形的象素总和就可以在常数时间内完成。例如下图，要求的矩形区域 D 的象素和。



（图 5.2.2 计算矩形特征）

图 5.2.2 中第 1 点的数值是矩形 A 的象素和，2 点数值是 A+B，3 点数值是 A+C,4 点数值是 A+B+C+D。所以 D 区域的象素和＝4+1-（2+3）。对于我们的 Haar 特征，1 矩形的象素总和可以通过 4 个数值计算。2 矩形特征计算 6 个，3 矩形特征计算 8 次，4 矩形特征计算 9 次。

假设需要识别的子图像大小为W*H，在这个子图像中的一个矩形可以定义为： r=(x,y,w,h,α)，0<x,x+w<W, 0<y,y+h<H,x,y>0,w,h>0, α ∈(0°,45°)

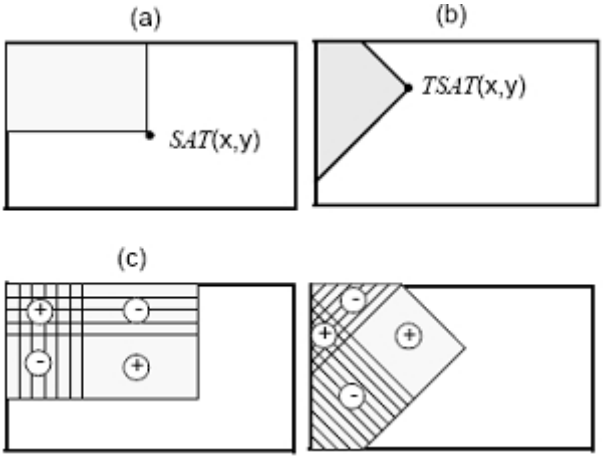


（图 5.2.3 矩形特征和旋转矩形特征定义）

那么，一个 Haar 特征就可以形式化定义为：

$feature = \sum_{i \in I=\{1,\ldots,N\}} \omega_i i(r_i)$，N 标识这是 N 矩形特征，$\omega_i$ 表示权重序列，r 为矩形

旋转的 Haar 特征计算和一般的 Haar 特征类似，如下图所示，左边是一般 Haar 特征，右边是旋转 Haar 特征。更具体计算方法参见文献[11]。



（图 5.2.4 计算 Haar 特征）

由于使用了积分图像以及基于简单矩形的 Haar 特征。所以 IHAAR 算法可以达到非常快速的检测速度，基本对人脸图像可以做到实时计算。P4 以上 CPU 基本可以做到很流畅的检测视频流数据。

使用积分图形的第二个好处，就是基于矩形的特征，可以在多尺度上进行计算，并且可在常数时间内完成。我们知道，一个数字化的图像中，随着目标离摄像机的距离不同。由于透镜成像的特点，导致目标所在区域的大小是不同的。离摄像机越远，目标越小；反之越大。要检测出图像中的任意大小的目标，必须进行多尺度扫描。

过去，都是使用“金字塔（pyramid）”计算。每一遍扫描图像的时候，都使用高斯函数，对上一次扫描的图像进行象素归并运算。这样就可以使用一个固定尺度的检测器在这些不同尺度的图像上扫描。金字塔方法运算量非常大。

IHAAR 检测器和其他的检测器一样，也需要在多个尺度扫描，一般图像扫描了 11 个尺度，每次搜索窗口增量 1.2 倍。由于利用了积分图像的任意矩形可以在常数时间内计算的特性，多尺度扫描基本不需要增加任何的计算量。

总之，积分图像使得人脸检测算法达到令人惊叹的快速，而且其多尺度特性给研究者带来很多便利，本后还有叙述。

5.3 学习算法

对于 Haar 特征而言，一个 24*24 的矩形区域，可以形成的 Haar 矩形特征达几万种，远远超过了 24*24 的象素的个数。即使每个特征都可以快速的计算，计算所有的集合都非常耗时。我们可以首先假设，一个很小的特征集合就可以组合成有效的分类器（后来的试验证实了这一个假设）。但是，最大的挑战就是如何选择这些特征。

在给定正例图像和反例图像作为训练图像集合后，再针对某个特定的特征集合，可以通过任何机器学习方法来训练。例如：混合高斯模型，神经系统，Winnow 学习等等。最近机器学习研究提出的支持向量机和放大（boost）方法，都可以在非常高维的空间中进行分类。我们使用后者，因为它可以在许多可能的特征中选择很少一部分。

AdaBoost 最早由 Freund 提出来。IHAAR 使用 AdaBoost 的一个改良后的方法[5]，它不仅用来选取特征，而且用来训练分类器。这个算法通过对简单学习算法的放大（boost），提升分类器的性能，（例如简单的感知器 perceptron）。通过组合多个弱分类函数来形成一个强分类器。它被叫做弱学习器。例如，一个感知器学习算法搜索所有感知器的集合，返回一个最小分类误差的感知器。

放大（boost）算法中，所谓的“弱学习器”（weak learner），就是指那些简单的学习算法。我们不期望最好的分类函数来对训练数据分类，例如最好的感知器也只能对训练集合达到 51%的分类。为了使得弱分类的增强放大，需要进行一系列的学习。对第一轮学习后，样本被重新计算权重，增强那些非正确分类部分。最后，一个“强分类器”就形成了，它是弱分类器在取某个阀值后的加权组合。例如，对于“简单感知器”而言，最终的“强分类器”就是“简单感知器”的加权组合，等价于一个 2 层的神经网络。

AdaBoost 算法在理论上的论证很严密。Freund 和 Schapire 证明[7]：强分类器的训练误差可以收敛于 0。也就是说，一个简单的、数量很少的弱分类器可以进行组合成为强分类器。

AdaBoost 学习算法的学习过程，可以理解为“贪婪的特征选择过程”。对一个问题，通过加权投票机制，用大量的分类函数的加权组合来判断。算法的关键就是，将那些分类效果好的分类函数赋予大的权重，分类效果差的赋予较小的权重。AdaBoost 是一个寻找那些可以对目标很好进行分类的少数特征的有效方法。

实际应用中，使用 AdaBoost 的方法选择特征，就是将“弱学习器”加上一个限定，一个“弱学习器”对应一个矩形特征，在进行放大(boost)的过程中，每一次放大选择一个学习器，就是选择一个特征。这个学习器对正例和反例的区分度达到最优。对每个特征，“弱学习器”使得每个分类函数的阀值达到最优。

使用 AdaBoost 算法还可以应用于图像检索，参见文献[12]。该文先构造高选择性特征（Selective Features），然后进行放大学习，得到的少量特征，最后应用于图像检索。该文献对本文具有很大的参考意义。

一个“弱分类器”$h_j(x)$包含特征f_j，阀值 θ_j和符号p_j，符号p_j表示进行阀值比较的方向（正或者负），x是 24*24 的子图像。使用的分类模型是高斯模型。

$$h_j(x)=\begin{cases}1 & p_j f_j(x) < p_j\theta_j \\ 0 & \text{其他}\end{cases}$$

单一的特征无法保证分类达到很低的误差。在早一轮的选择特征的错误率可以在 0.1 到 0.3 之间，而随着后一轮的放大，误差会增大，在 0.4 到 0.5 之间。

下面是学习算法：（T 为特征个数）

（1）、于样本(x1,y1),……,(xn,yn)， yi＝1 或者 0，表示图像是正例或者是反例

（2）、初始化权重值$w_{1,i}$

$$W_{1,i}=\begin{cases}\frac{1}{m} & y_i=0\\ \frac{1}{n} & y_i=1\end{cases}$$

（3）、t=1

（4）、将权重归一化。

$$W_{t,i}=\frac{W_{t,i}}{\sum_{j=1}^{n} wt,j}$$

（5）、对每个特征 j，训练一个只使用某一个单一特征的分类器 hj。然后得到本次分类的误差e_j。

$$e_j=\sum_i w_i\left|h_j(x_i)-y_i\right|$$

（6）、选择误差e_j最小的分类器h_t，更新权重：

$$w_{t+1,i}=w_{t,i}\beta_t^{1-v_i}$$

6.1 当x_i分类正确时：$v_i=0$。

6.2 当x_i分类错误时：$v_i=1$，$\beta_t=\frac{e_t}{1-e_t}$

（7）、t=t+1

（8）、t 小于 T，转向（4）

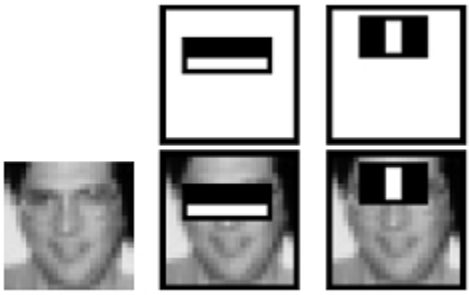
（9）、得到最后的分类器：

$$h_j(x)=\begin{cases}1 & \sum_{t=1}^{T}\alpha_t h_t(x)\geq\frac{1}{2}\sum_{t=1}^{T}\alpha_t\\ 0 & \text{其他}\end{cases}$$

这种选择特征的方法，还可以参照文献[10]，它成功的从 1734 个特征中选出 37 个。

AdaBoost 算法构造的特征最后大概 200 个。对于人脸检测，AdaBoost 选择的权重最高的几个矩形特征很容易理解。第一个特征关注的，就是包含眼睛的区域总是比鼻子和面颊的矩形区域亮度上更暗（亮度值小于某个阀值）。这个特征是和人脸大小以及位置无关。第二个特征表示的，是眼睛所在的区域总是比鼻梁

所在的区域亮度更暗。所以，通过选择和学习后的包含 200 个特征的分类器，对于目标检测是非常有效的。



（图 5.3.1 易理解的 Haar 特征）

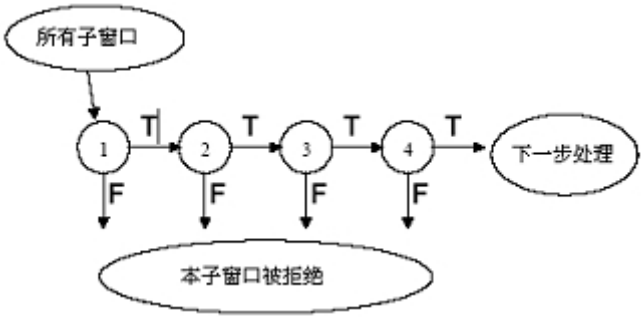
5.4 检测算法

人脸检测算法是一个类似瀑布的结构，简单的 Haar 特征分类器放在前面，复杂的放在后面。为什么要使用瀑布结构呢？一个关键原因就是它可以使检测算法更加高效。由于我们图像中绝大多数区域都是非人脸的，只使用那些简单而高效的分类器，就可以很快的“拒绝”反例。并且，将简单的分类器放在复杂的前面，有利于达到更高的检测率和降低错检率。

在 AdaBoost 算法中，较低的阀值可以达到更高的检测率和更高的错检率。例如图 5.3.1 中的 2 矩形特征的单个检测器，调节阀值后，达到 100％的人脸检测率和 40％左右的人脸错检率。要想它达到更高的水平不可能，但是，这样简单的 2 特征分类器，需要的 CPU 计算周期非常少（6 到 9 个对积分图像的取值操作，1 个比较操作），它可以大大减少后继分类器需要分类的子窗口数量。

从算法复杂度来比较。其他的算法，无论是模板匹配，还是单层感知器，在每个子窗口的计算量都至少是简单 Haar 特征方法的 20 倍。

所以，整个人脸检测的算法结构，就是一个退化的决策树，有的文献把它叫做“层叠器（Cascade）” [8]，结构见下图。

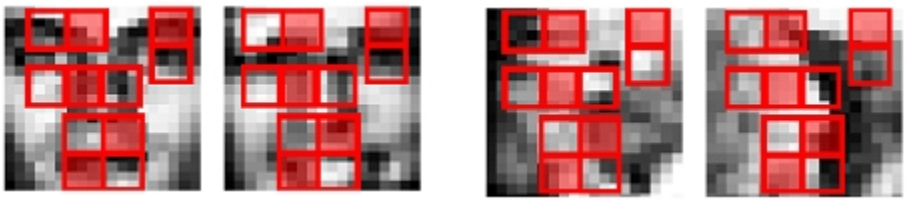


（图 5.4.1 层叠器结构）

前面一层的分类器对子窗口的图像进行分类后，如果分类结果是“T”，那么正例就传递到下一个分类器，并且触发下一个分类器进行处理，如此下去直到最后一个分类器。通过学习算法，调整好这个瀑布型结构的参数后，可以将检测器的检测率提高很多。反之，一个反例（非人脸）将不会到达“终点”就会被前面的分类器拒绝。我们的算法一次大约拒绝 50%左右的非人脸模式，一共 13 层。

总之，层叠式的结构反映了这样的事实，就是在任何一个的图像中，层叠器总是试图在早期就拒绝大多数的反例，而正例总是触发所有的分类器。

如何设计和构造一个优秀的层叠器呢？需要设计的包括层叠器的层数(stages)、每一层的特征数和每一层的阀值。IHAAR 系统中的层叠器包含 32 层。第一层使用 2 个简单的 2 矩形 Haar 特征分类器，如图 5.3.1，可以拒绝 60%的反例和达到 100%检测率；第二层使用 5 个特征分类器，如图 5.4.2，可以拒绝 80%的反例和达到 100%的检测率；第三、四、五层使用 20 个特征分类器。



（图 5.4.2，5 特征分类器）

和决策树方法类似，层叠器这样一个序列的分类器的训练方法，总是使用在前面几遍通过了的样本。后一个分类器总是面对比前一个更加困难的分类任务，所以，后一个分类器比前面一个具有更高的漏检率。

人脸检测的算法，来自于 OpenCv，具体见 www.opencv.org.cn.

第六章 利用边缘能量的启发式搜索

研究了基于肤色和基于 Haar 特征的人脸检测算法，我们自然会想到肤色特征失效的问题。最坏的情况：一个人站在和肤色完全一模一样的背景前面，我们的肤色模型会将所有的背景都看作是肤色，这时基于肤色的人脸检测将完全失效。IHAAR 算法退化成纯粹的 Haar 特征人脸检测。为了解决这个问题，我们引入了基于边缘能量的启发式搜索。

6.1 边缘检测

通过第五章，我们可以看到：使用了层叠器的 Haar 特征检测，其内部机制就包含了启发式搜索的思想。为什么还要再谈启发式搜索呢？原因是，Haar 特征检测只利用了图像亮度信息。而“边缘”(edge)这个图像中更加最重要的信息，却始终没有被利用到。

作为机器视觉最早研究的对象的，边缘检测一直是图像处理和机器视觉的热门研究课题，边缘检测技术的越来越成熟。

所谓边缘，是图像局部亮度变化最显著的部分，或者说是连续图象 f(x,y)，其方向导数在边缘（法线）方向上有局部最大值的点。

（1）常用的一阶边缘检测算子：

算子名	H_1	H_2	特点
Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	• 边缘定位准 • 对噪声敏感
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	• 平均、微分 • 对噪声有抑制作用
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	• 加权平均 • 边宽≥2 象素

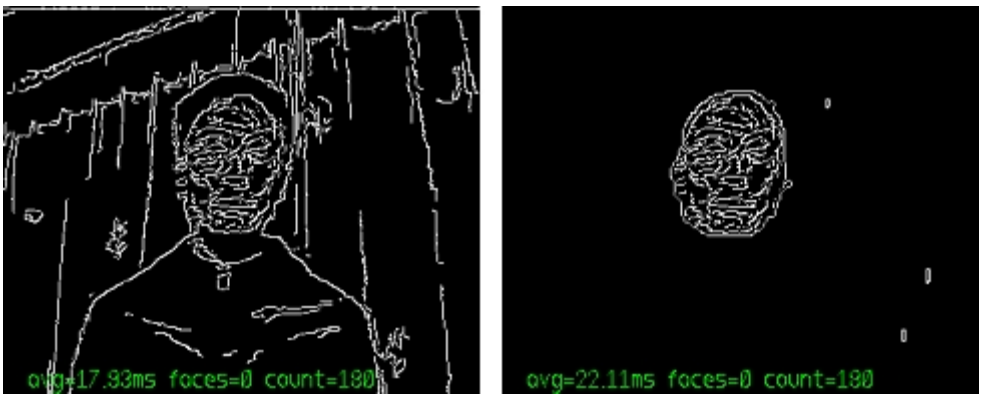
（表 6.1.1 边缘检测算子）

（2）二阶边缘检测算子如 Laplacian 算子，$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$，具有平滑功能的 LoG(Laplacian of Gaussian)算子。

（3）Canny 边缘检测器。它是高斯函数的一阶导数，是对信躁比与定位之乘积的最优化逼进算子[32]。利用了高斯函数的一阶导数，对噪声的抑制非常好，同时具有可以精确定位的特点。我们选择 Canny 检测作为系统边缘检测的主要方法。

6.2 利用边缘能量的启发式搜索

人脸由于存在五官特征、面部表情，所以有着丰富的边缘信息（如图 6.2.1），“边缘能量”（边缘图中子窗口象素的亮度和）较高。反之，如果背景不很复杂，那么边缘信息少，边缘能量低。抓住边缘能量这个特征，可以作为启发式搜索的启发函数。



（图 6.2.1 Canny 边缘检测和人脸）

在 5.4 中的层叠器以不同尺度搜索子图像之前，利用边缘信息来启发式搜索。同时，为了提高运算速度，我们再次利用了积分图像的思想。假设待测试图像为 Img(x,y)：

它的 Canny 边缘检测图像为 Canny(x,y)，边缘积分图像为 Integ(x,y)，当前搜索的子窗口为 rect(x,y,x2,y2)。该子窗口的边缘能量 e：

e= Integ(x2,y2)- Integ(x2,y)- Integ(x,y2)+ Integ(x,y)

如果 e 小于某个阀值 K(0<K<255)，表示这个子窗口的边缘信息不够丰富，一定不是人脸。反之，e 大于阀值 K，表示子窗口有足够的边缘信息，将之送入层叠器开始人脸检测。阀值 K 通过反复实验取得。

第七章 软件系统设计与实现

将上述几章的各个分散的功能综合起来，再利用微软的最新 DirectX 视频处理技术，就形成了基于 Haar 特征的完整的人脸检测实时系统。

7.1 实时系统的关键技术－DirectX

为了构建 Windows 系统的机器视觉平台，必须使用 DirectX 技术。微软的 DirectX 软件开发工具包（SDK）提供了一套优秀的应用程序编程接口（APIs），这个编程接口可以提供开发高质量、实时的应用程序所需要的各种资源。

使用 DirectX 的主要有两个好处：1、为软件开发者提供硬件无关性；2、为硬件开发提供策略。前者使得我们的系统不再倚赖于任何特定的视频采集设备。

DirectX SDK 为基于 Windows 平台的应用程序提供了以下几个组件。

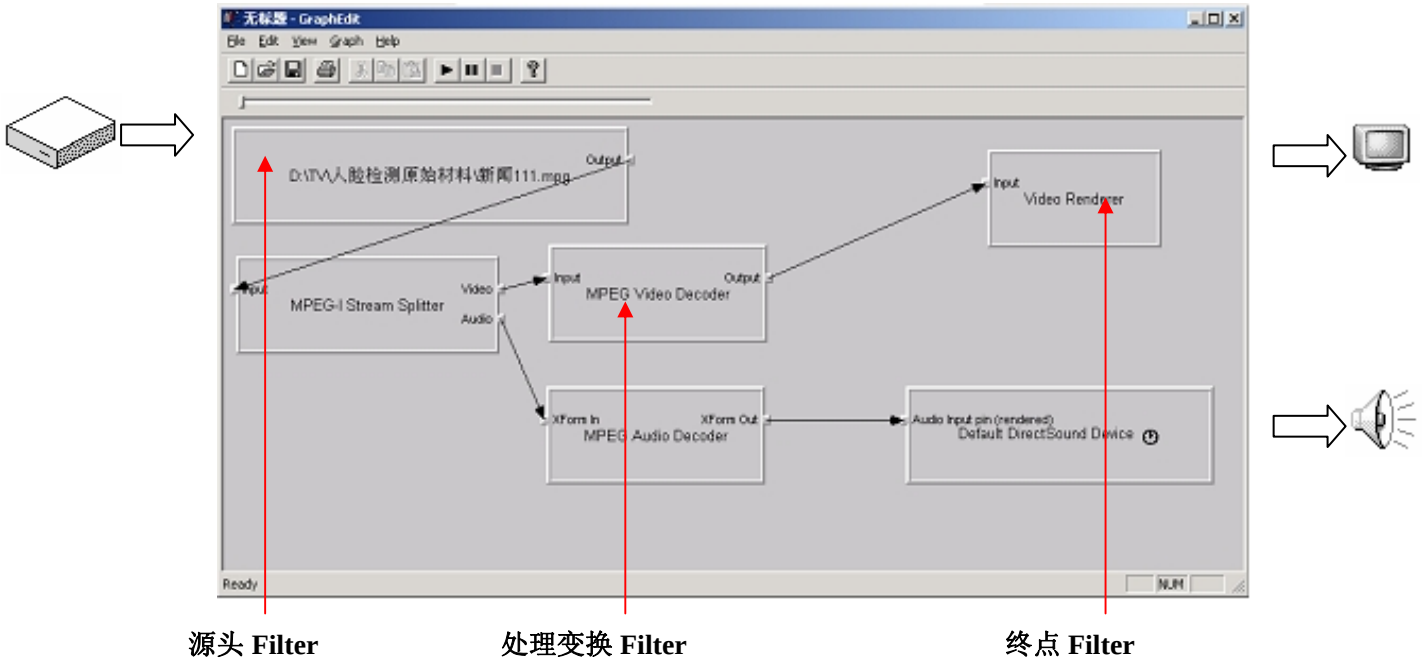
- DirectDraw ： 通过直接访问显示硬件来提供高级的图象处理能力。
- DirectSound ： 它提供了软硬件的低延迟声音混频和回放，硬件加速，以及直接访问音频设备的能力。
- DirectPlay ： 简化应用程序之间的通讯服务。
- Direct3D ：它为主流的 PC 和 Internet 用户提供实时的、交互的 3D 技术。
- DirectInput ：它简化你的应用程序访问鼠标、键盘和操纵杆设备的能力。
- DirectShow：它是早期 Video For Windows (VFW)的继承者，保持兼容 VFW 并且提供了更加优秀的接口。

我们的人脸检测系统基于 DirectX 技术，并主要应用 DirectShow 组件。DirectShow 技术是构建“实时”机器视觉系统的关键，也是建立机器视觉平台的主要的技术难点。

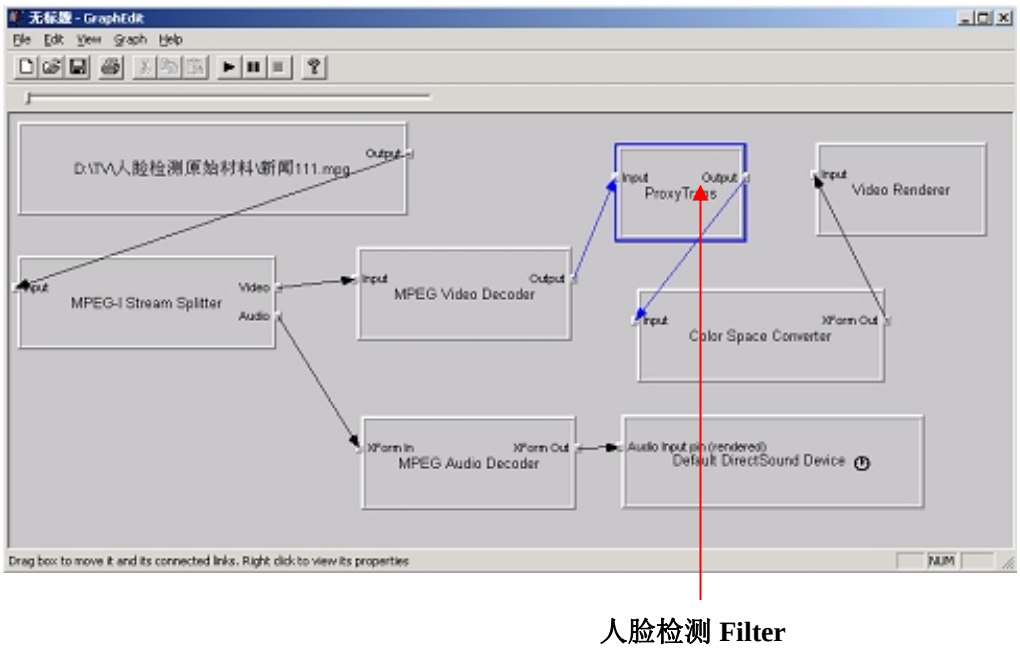
DirectShow 是基于 **COM**（Component Object Model）接口的，所以，我们可以将 DirectX 程序中各个程序组件看作对象的实例。在 DirectShow 中叫做 **Filter**，各个软硬件厂商可以开发自己的 Filter，例如可以播放 MPEG-4 格式的 Filter、可以访问 DVD 光驱的 Filter 等等，它们都遵循同一个软件接口。Windows 系统存在非常多不同类型的 Filter，这些 Filter 象盖房子的砖瓦一样搭建起一个系统。

利用工具 GraphEdit 可以可视化的编辑和查看 DirectX 的对象。下图是播放一

个 MPEG 文件的 Filter 图，MPEG 文件首先被“MPEG-1 Strem Splitter”分割成为视频流和音频流。然后，视频流由“MPEG Video Decoder”解码，最后送“Video Renderer”，由其显示在监视器的屏幕上；音频流解码后送声卡。



（图 7.1.1 播放 MPEG 文件的 Filter 图）



（图 7.1.2 人脸检测系统）

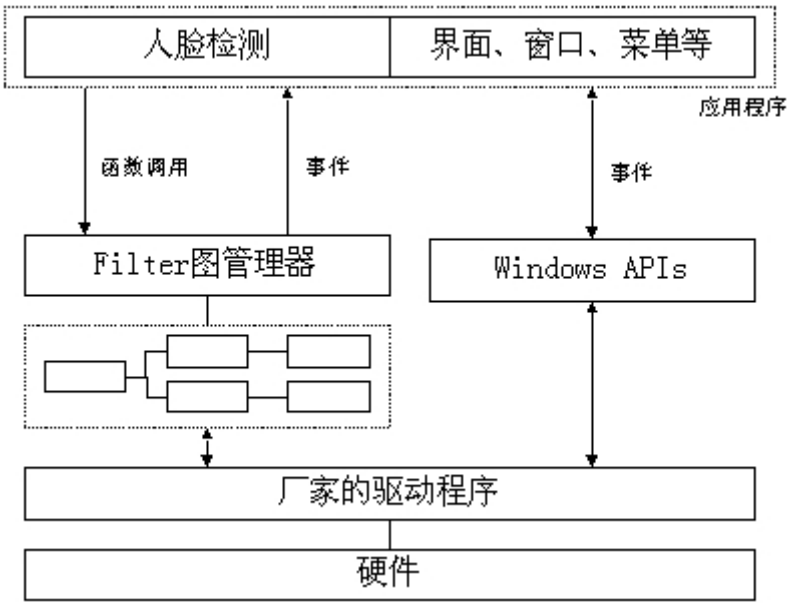
对比图 7.1.1 和图 7.1.2 可以看出，我们自己制作了一个 Filter，名字为“ProxyTrans”，利用程序设计语言或者手工将这个 Filter 插入到“MPEG Video Decoder”之后，“Video Render”之前。它的功能是：接受视频流的每一帧图像，

调用我们的图像处理算法进行处理，将结果送往下一个 Filter。关于 Filter 的更加详细的信息请查阅 MSDN（Microsoft Software Development Network）。

对于不同格式的视频文件和摄像机等不同的视频采集设备，只需要将源头 Filter 替换成对应的 Filter 即可，我们的应用程序改动非常少。基于 COM 接口的面向对象的层次性 Filter 结构的灵活性和兼容性，立刻体现出来。

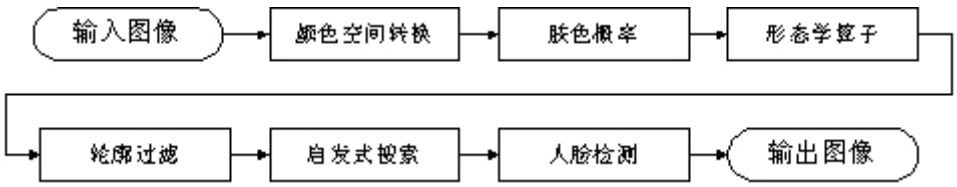
7.2 系统结构

基于 DirectX 的应用程序、Windows 日常 API 和 DirectX 的相互关系如下图：



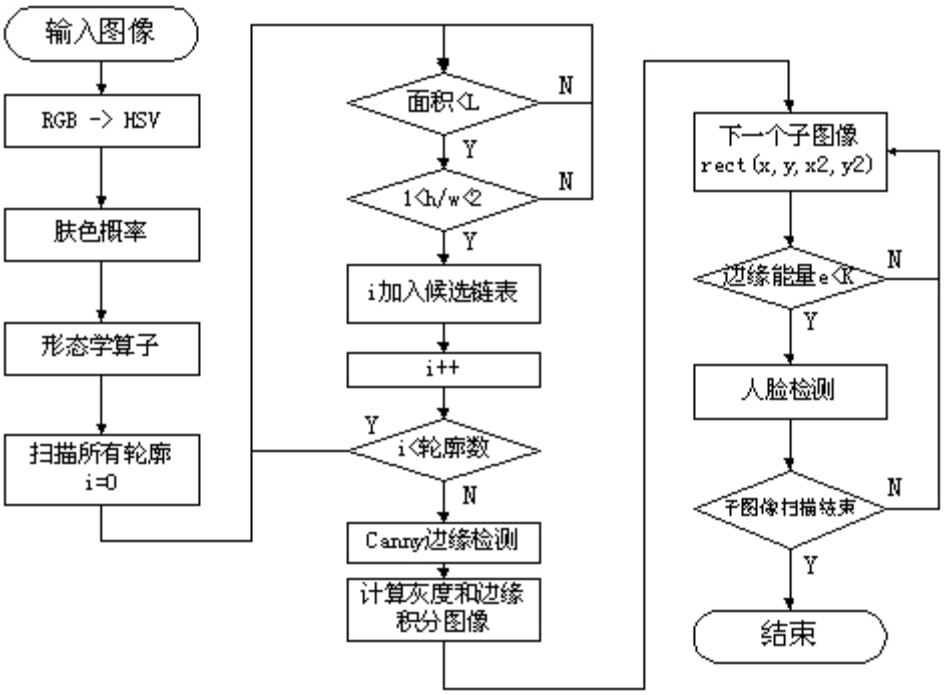
（图 7.2.1 DirectX 和应用程序）

Windows 日常 API 处理和用户的交互，包括菜单、窗口、对话框等；“Filter 图管理器”管理如图 7.1.2 的 Filter 层次结构，处理视频解压（采集），压缩数据解码、人脸检测、屏幕显示等；其中，最关键的人脸检测 Filter 的流程图如下：



（图 7.2.2 人脸检测 Filter 流程图）

将图 7.2.2 的流程图进一步细化，形成了人脸检测的算法。



（图 7.1.3 人脸检测算法框图）

7.3 实验环境

人脸检测系统采用 Visual C++ 6.0 开发，软件开发采用 MFC 的单窗口框架结构，以及微软 DirectX 8.0 的 SDK，硬件平台是 p4 赛扬 1.7G，内存 128M，采集卡（或摄像头）。

试验数据包括通过摄像机得到的视频材料，和用电视卡采集的视频片断。视频图像大小为 352×288 象素，包含 BGR 共 3 个颜色通道。全部是未经修饰和处理的原始素材，包含复杂背景。

我们选择了几个典型材料来分析：材料一为摄像机采集的人物片断、材料二为中央电视台新闻片断、材料三为中央教育电视台百家讲坛片断。

（图 7.3.2 实验材料）

评价算法性能时，使用的是 Windows 平台下最精确的时钟 API：QueryPerformanceCounter()，可以精确到毫秒。

7.4 实验结果与分析

（1）计算复杂度性能分析

项目	材料一（毫秒）	材料二（毫秒）	材料三（毫秒）
颜色空间转换	5.83	5.82	5.90
肤色概率	0.81（6.64）	0.76（6.58）	0.73（6.63）
形态学算子和轮廓过滤	10.9（17.54）	11.27（17.85）	11.16（17.79）
人脸检测	37.33（54.87）	44.33（62.18）	42.85（60.64）

（表 7.4.1 改进后的人脸检测）

对比实验：

项目	材料一（毫秒）	材料二（毫秒）	材料三（毫秒）
Viola 算法	70.26	107.90	89.75

（表 7.4.2 Viola 原始算法性能）

通过两组对比试验数据可以看出，算法性能提高如下表：

项目	材料一	材料二	材料三
性能提高百分比	21.9%	42.37%	32.43%

对比三则材料颜色空间转换、肤色概率和获得连通区域这三个模块的变化基本不大。而人脸检测模块的变化比较大。主要和图像中的人脸区域有关。

还可以看出，在不同的场景下，算法效率的提升是不同的。材料一的背景最简单，性能提升幅度最小，材料二的背景最复杂，性能提升幅度最大。

（2）检测率

项目	材料一		材料二		材料三	
	个数	检测率	个数	检测率	个数	检测率
Viola 算法	3	1.81%	203	100%	138	30.39%
IHAAR 算法	162	98.18%	196	96.55%	156	34.36%
实际人脸数	165		203		454	

（表 7.4.3 检测率分析）

我发现材料一的 Viola 算法检测率低得让人不可思议，原因是材料一的背景对人脸检测的影响较大。材料二，Viola 算法检测率略高于我们的算法。材料三，

检测率都不高，原因是材料中的人说话“声情并茂”，常常眨眼睛，闭眼时黑色瞳孔这个重要特征从画面消失。闭眼情况虽然和学习样本很大不符（学习样本全是睁眼的），不应该看作正例，但是我们仍把此时的人脸当作正例。这种计算方法导致检测率不高，同时也说明人脸表情的变化对人脸检测的影响巨大。

（3）错检率

项目	材料一		材料二		材料三	
	个数	错检率	个数	错检率	个数	错检率
Viola 算法	0	0%	16	7.88%	38	8.37%
IHAAR 算法	0	0%	0	0%	0	0%

（表 7.4.4 错检率分析）

材料二和三的 Viola 算法错检率都比较高，而 IHAAR 算法错检率极低。

本论文构造了一个在 Windows 平台下的实时人脸检测系统、改进了基于 Haar 特征的人脸检测算法。通过上述的实验和结果分析，可看到：使用肤色模型、形态学算子、轮廓过滤、边缘启发式搜索等综合措施，改进后的 HAAR 人脸检测算法，和 Viola 等人在 2002 年提出的算法比较，提高了算法的性能，提高了检测率和降低了错检率。

第八章 背景模型和人脸定位

8.1 概述

背景分割作为计算机视觉的重要研究课题之一，在视频监控、智能化交通系统、人机交互等领域有着广泛的应用。摄像机静止时，背景分割的主流方法是背景减除法，即用当前帧与背景参考模型进行比对，再通过阈值法来判断各个像素是否属于前景目标。视频背景中干扰运动是大量存在的(尤其是在室外场景中)，这使得构建合适的背景模型成为应用背景减除法成功的关键。

研究者已经提出了多种背景模型，如时间平均模型(Temporal average model, TAM)[62]和单高斯背景模型(Single Gaussian model, SGM)[63]计算量小，但是它无法处理场景中光线的缓慢变化带来的问题，且当背景像素亮度呈多峰分布时检测效果不佳。高斯混合模型(Gaus-sian mixture model, GMM)[64]将像素的亮度分布用几个高斯函数的加权和来近似，在一定程度上能够适应多峰分布的背景。与 TAM 或 SGM 相比，GMM 检测性能的提高是用大的计算代价换取的，但是，随着计算机性能的提高，基于 GMM 算法的应用越来越成熟[57][60]，因此本文利用 GMM 作为背景模型。

但是，研究 GMM 后就会发现，GMM 学习速度较慢且机械，特别是前几祯就是前景时，需要很长时间才能纠正过来。参数估计较机械化。于是，出现了很多对 GMM 的改进方法。例如文通过改进高斯混合模型的参数更新策略来提高效率[65]； ZivkovicZ[66]在混合高斯模型中引入了高斯函数个数的自动选择机制。同时，最大的问题就是误将移动目标的背景判断为前景，给后续的目标识别带来极大的困难，因此，本文引入了阴影检测[67]。

将背景分割后，第二步就是获得人体的轮廓（剪影）。轮廓被作为自动视频监控系统的重要特征，区分公路上的汽车和人体[68]。本文为了提高检测速度，采用了基于积分图像[6]和人体解剖特征[69]的快速检测，然后初步定位人脸的位置。

8.2 GMM 背景模型

多模态背景的需要用多个分布来共同描述一个图像点上的颜色分布。Stauffer 等[64]提出了一种自适应混合高斯模型，对每个图像点采用了多个高斯模型的混合表示。设用来描述每个点颜色分布的高斯分布共有 K 个（K＝3…5），N 时刻某个象素具有观测值 x_N，它可以用用这 K 个高斯分布的混合表示为：

$$p(x_N)=\sum_{j=1}^{K} w_j \eta(x_N,\mu,\Sigma)$$

其中 w_j 是第 j 个高斯分量的权重，$\eta(x,\mu_k,\Sigma_k)$ 是第 K 个高斯分布，μ_k 是均值，$\Sigma_k=\sigma_k^2 I$ 是方差。

$$\eta(x\ ,\mu_k,\Sigma_k)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_k|^{\frac{1}{2}}}e^{\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

各高斯分布分别具有不同的权值和优先级 w_k/σ_k，它们总是按照优先级从高到低的次序排序。按照优先级次序将像素与各高斯分布逐一匹配，若没有表示背景分布的高斯分布与匹配，则判定该点为前景点，否则第为背景点 B。匹配公式如下，其中阈值 T 决定了背景在整个混合模型中占的百分比。

$$B=\arg\min_b\left(\sum_{j=1}^{b} w_j > T\right)$$

若像素没有和任何一个高斯分布匹配，则将优先级最小的一个高斯分布去除，并根据引入一个新的高斯分布，并赋予较小的权值和较大的方差，然后对所有高斯分布重新进行权值归一化处理。

若像素与某个已知的高斯分布匹配，则对第 k 个高斯分布的权值更新公式：

$$w_k^{N+1}=(1-\alpha)w_k^N+\alpha p(\omega_k|x_{N+1})$$

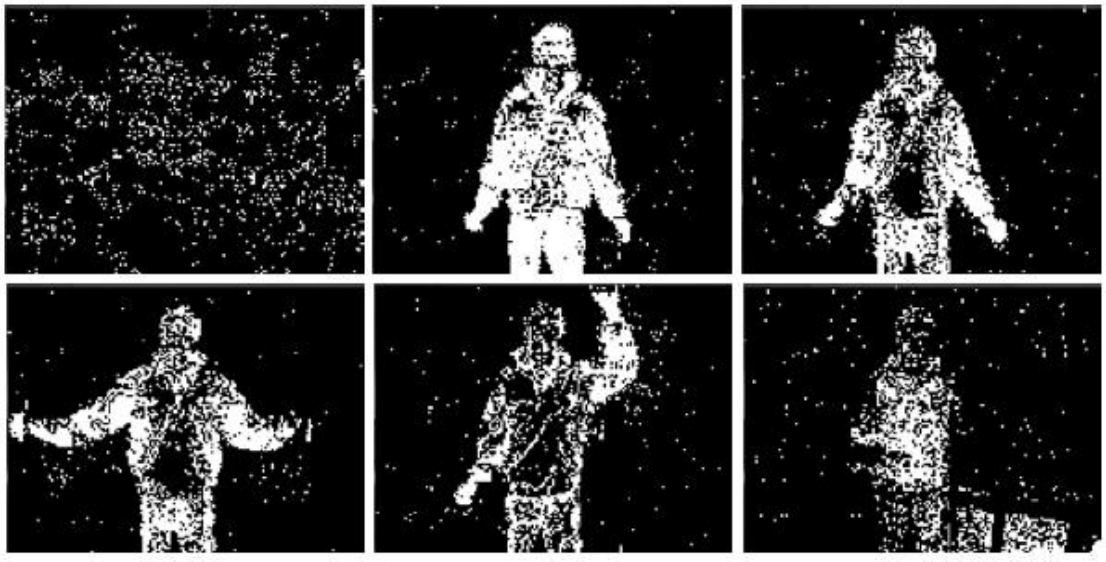
$$\mu_k^{N+1}=(1-\rho)\mu_k^N+\rho x_{N+1}$$

$$\Sigma_k^{N+1}=(1-\rho)\mu_k^N+\rho(x_{N+1}-\mu_k^{N+1})^T(x_{N+1}-\mu_k^{N+1})$$

$$\rho=\alpha\eta(x_{N+1},\mu_k^N,\Sigma_k^N)$$

$$p(\omega_k|x_{N+1})=\begin{cases}1; \textit{if } \omega_k \textit{ is the first match Gaussian component}\\ 0; \qquad\qquad \textit{otherwise}\end{cases}$$

背景模型的更新策略是背景模型设计中最关键的技术。其中 α 是表示背景更新快慢的常数，实验表明，很多时候无法找到一个合适的更新常数。如下图，背景更新系数如果保持不变且较高时，走到场景中的前景会慢慢变为背景，好像人体慢慢被腐蚀。



（图 8.2.1 更新系数高时效果）

因此不少针对更新策略的研究，如有研究使用 EM 算法更新模型参数[65]，其收敛速度高于上述算法；也有定义遗忘因子和学习率因子的更新算法[71]；排序的变化[73]。而文本的背景模型更新，采用随着时间动态线性递减的方式。递减系数 L(L=100)，也就是前 L 帧的更新系数是 α，第 n*L 帧的更新系数是：

$\alpha^{N+1}=\alpha^{N}*(1-*0.01*\text{int}(n-L)/L)$

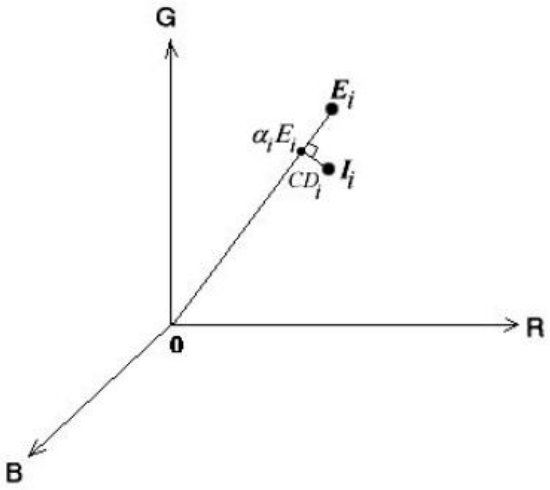
这样的设计，实际上提高了前 L 帧中像素是背景像素的先验概率。所以，我们在采集视频的时，前 L 帧尽量保证不出现人体，以取得最佳效果。

8.3 阴影检测

基于 GMM 模型的背景分割技术中，最困难的问题之一就是运动目标的阴影。若背景匹配阈值 T 太大，容易将阴影被误判为前景，如下图。相反，若 T 值太小，前景容易丢失。

（图 8.3.1 阴影误判）

为了解决这个矛盾，不少研究者采用阴影检测的方法[74][75]。阴影是亮度变化较大，且比背景像素低，色度变化较小的像素。有研究在 HSV 色彩空间进行阴影检测[76]，本文采用 Horprasert 的方法[67]，在 RGB 空间直接计算色度线。如图 8.3.2，RGB 色彩空间中，I 表示像素，E 表示该像素进行背景检测后的高斯分量的均值，OE 表示色度线。若 I 的落在 OE 直线上，即新像素的亮度变化了，但是色彩没有变化；若 I 落在 OE 直线外，I 投影到 OE 上的点为 α E，I 到 OE 的垂直距离为 CD(Color Distortion)，CD 表示色彩的差异性。



（图 8.3.2 色彩模型）

求 α 和 CD 的公式[67]如下：

$$\alpha = \arg\min_{z}(I - zE)^2$$

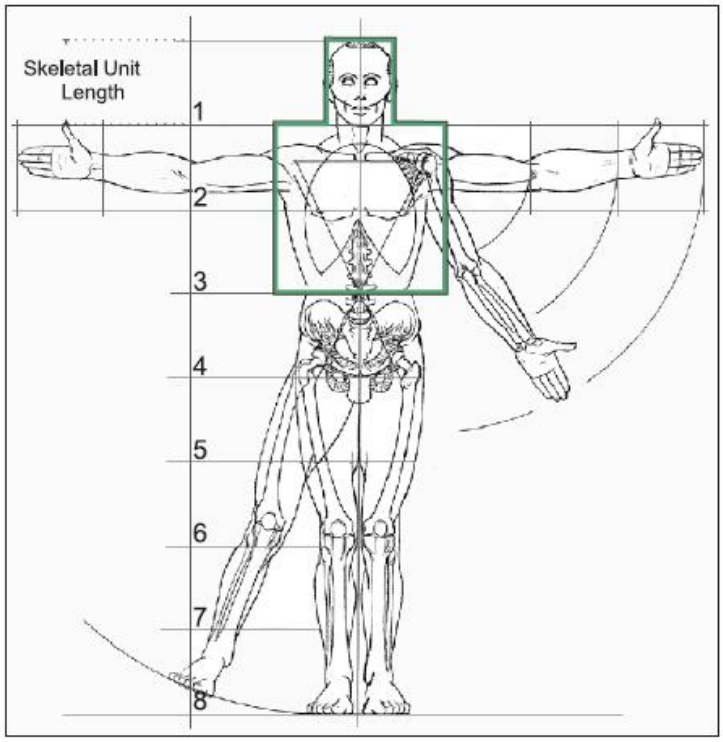
$$CD = \left\| I - \alpha E \right\|$$

阴影检测效果如下图，图中黑色为背景，白色为前景，灰色为阴影。

（图 8.3.3 阴影检测）

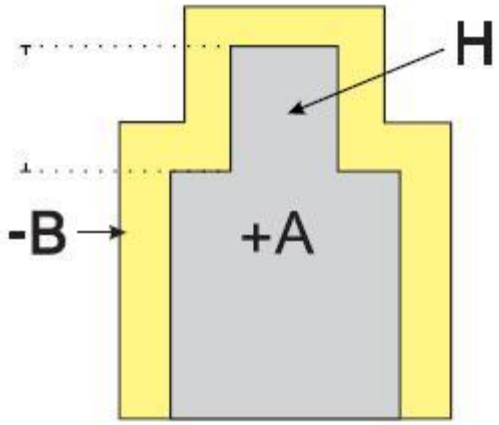
8.4 人脸粗定位

划分出前景和背景后，得到了人体轮廓，接着就是人脸的粗定位。其中人体轮廓的最大特点就是其头肩的比例。如下图，人体的头和躯干的长宽比例基本是固定的，且人体四肢运动时，该结构基本维持不变。所以，本文人体检测就利用了这一特征。将人体分成如图所示的“由”字形的矩形区域。



（图 8.4.1 人体解剖结构）

若头肩轮廓搜索过程中进行全画面搜索，计算量非常大。受到 Viola 的论文的启发，本文使用了积分图像来进行计算（具体见 5.2 节），实现了实时计算。如下图，本文使用的模板将头肩轮廓分为若干矩形模板，然后求模板各区域的像素总和。A 是人体区，B 是轮廓外围区，H 是人脸位置，其中 A 的面积等于 B，H 的长宽比 1:0.8。

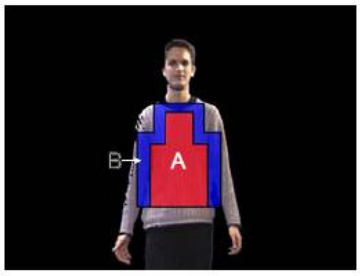


（图 8.3.2 人体检测模板）

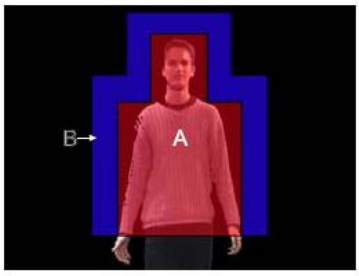
之所以要将 A 的面积等于 B，是为了计算相似度的公式：

$$\gamma = \begin{cases} (\Sigma A - \Sigma B)/area(A) & if\ \ \Sigma A > \Sigma B \\ 0 & otherwise \end{cases}$$

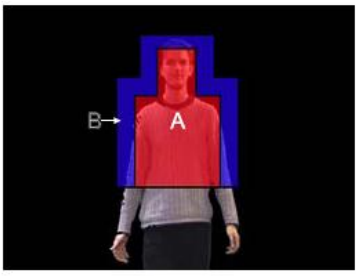
该公式的优势在于，模板刚好和人体轮廓匹配。如下图，B 包含的全是背景，A 的区域全是前景的时候，γ 的值最大，相反，若 B 包含前景，或者 A 包含太多背景，γ 的值低。



（A，ɣ 低）



（B，ɣ 低）



（C，ɣ 高）

（图 8.3.3 ɣ 的三种可能）

人脸粗定位后，就获得人脸的大概位置，再进行更加精确的人脸检测或者识别，参见第 5 章。

参 考 文 献

[1]H.A.Rowley,S.Baluja,andT.Kanade. Neural Network-Based Face Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1): 23-38, Jan. 1998.

[2]M.Turk,and A.Pentland,Face Recognition Using Eigenfaces, In Proceedings of International Conference on Pattern Recognition, pp. 586-591, 1991.

[3]R.Chellappa,C.Wilson,and S.Sirohey,Human and Machine Recognition of Faces: A Survey, Proceedings of IEEE, vol. 83, May. 1995

[4]D.Comaniciu,and P.Meer,Robust Analysis of Feature Spaces: Color Image Segmentation, CVPR'97, pp. 750-755, 1997.

[5]Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. In Computational Learning Theory:Eurocolt '95, pages 23–37. Springer-Verlag, 1995.

[6]P.Viola and M.Jones.Robust real-time object detection. Technical Report 2001/01, Compaq CRL, February 2001. 8

[7]Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[8] J. Quinlan. Induction of decision trees. Machine Learning, 1:81–106, 1986.

[9]A.Mohan,C.Papageorgiou,T.Poggio. Example-based object detection in images by components. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 4, pp. 349 -361, April 2001.

[10]C.Papageorgiou,M.Oren,and T.Poggio.A general framework for Object Detection. In International Conference on Computer Vision, 1998.

[11] Rainer Lienhart and Jochen Maydt ,An Extended Set of Haar-like Features for Rapid Object Detection,ICIP2002.

[12]K. Tieu and P. Viola. Boosting image retrieval. In International Conference on Computer Vision, 2000.

[13]G. Yang and T. S. Huang, "Human Face Detection in Complex Background," Pattern

Recognition, vol. 27, no. 1, pp. 53-63, 1994.

[14]Ming-Hsuan Yang, David J. Kriegman and Narendra Ahuja,Detecting Faces in Images: A Survey,IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 1, january 2002,

[15] E. Osuna, R. Freund, and F. Girosi, “Training Support Vector Machines: An Application to Face Detection,” Proc. IEEE Conf.Computer Vision and Pattern Recognition, pp. 130-136, 1997.

[16] Yang, J., Lu, W., and Waibel, A. (1997), "Skin-color modeling and adaptation," Proceedings of ACCV'98 (Technical Report CMU-CS-97-146, CS department, CMU, 1997).

[17] J. Serra. Image Analysis and Mathematical Morphology. Academic Press, 1982.

[18]S. Suzuki, K. Abe. Topological Structural Analysis of Digital Binary Images by Border Following. CVGIP, v.30, n.1. 1985, pp. 32-46.

[19]周 杰,卢春雨,张长水等.人脸自动识别方法综述.电子学报, 2000, 28 (4) : 102- 106

[20]Terrillon J C, Sh irazi M N , Fukamach i H et al.Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in co lorimages. Conference on Automatic Face and Gesture Recognition, Grenoble, France, 2000.

[21]Jones M J , Rehg J M. Statistical color models w ith application to skin detection. IEEE Conference on Computer Vision and Pattern Recognition, FortCollins, Colorado, 1999.

[22]G.Yang and T.S.Huang, “Human Face Detection in Complex Background,” Pattern Recognition, vol.27, no.1, pp. 53-63, 1994.

[23]C. Kotropoulos and I. Pitas, “Rule-Based Face Detection in Frontal Views,” Proc. Int’l Conf. Acoustics, Speech and Signal Processing,vol. 4, pp. 2537-2540, 1997.

[24]S.A. Sirohey, “Human Face Segmentation and Identification,”Technical Report CS-TR-3176, Univ. of Maryland, 1993.

[25]D. Chetverikov and A. Lerch, “Multiresolution Face Detection,”Theoretical Foundations of Computer Vision, vol. 69, pp. 131-140,1993.

[26]K.C. Yow and R. Cipolla, “A Probabilistic Framework for Perceptual Grouping of Features for Human Face Detection,”Proc. Second Int’l Conf. Automatic Face and Gesture Recognition,pp. 16-21, 1996.

[27]T.K. Leung, M.C. Burl, and P. Perona, “Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching,” Proc. Fifth IEEE Int’l Conf. Computer Vision, pp. 637-644, 1995.

[28]P. Sinha, “Object Recognition via Image Invariants: A Case Study,” Investigative Ophthalmology and Visual Science, vol. 35,no. 4, pp. 1735-1740, 1994.

[29]T. Kohonen, Self-Organization and Associative Memory. Springer 1989.

[30]M. Kirby and L. Sirovich, “Application of the Karhunen-Loe`ve Procedure for the Characterization of Human Faces,” IEEE Trans.Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 103-108,Jan. 1990

[31]H. Rowley, S. Baluja, and T. Kanade, “Rotation Invariant Neural K.-K. Sung and T. Poggio, “Example-Based Learning for View-Based Human Face Detection,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 39-51, Jan. 1998.

[32]H. Rowley, S. Baluja, and T. Kanade, “Neural Network-Based Face Detection,” IEEE Trans. Pattern Analysis and Machine Intelligence,vol. 20, no. 1, pp. 23-38, Jan. 1998.

[33]J.F.Canny,A computational approach to edge detection,IEEE Trans,Pattern Analysis and Machine Intelligence,8(6),679-698,November 1986.

[34] Menser, B., Wien, M., Segmentation and tracking of facial regions in color image sequences. Proc. SPIE Visual Communications and Image Processing 2000, 731–740.

[35]Terrillon, J.-C., Shirazi, M. N., Fukamachi, H., Akamatsu S., Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. Proc. of the International Conference on Face and Gesture Recognition, 54–61.

[36]Yang, M., AHUJA, N. Gaussian mixture model for human skin color and its application in image and video databases. In Proc. of the SPIE: Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99), vol. 3656, 458–466, 1999.

[37]Garcia C, Zikos G, Tziritas G, Face detection in color images using wavelet packet analysis, In: Proc. Multimedia Computing and Systems, Centro Affari,Florence, Italy, 1999, Vol.1: 703-708

[38]Abdel-Mottaleb M, Elgammal A. Face detection in complex environments from color images. In: Proc. IEEE Conf. on Image Processing, Kobe, Japan, 1999, Vol.3(11-13): 622-626

[39]Karlekar J, Desai U B. Finding faces in color images using wavelet transform. In:Proc. IEEE Conf. on Image Analysis and Processing, Venice, Italy, 1999, 1085-1088

[40] Ishii H, Fukumi M, Akamatsu N. Face detection based on skin color information in visual scenes by neural networks. In Proc. Systems, Man, and Cybernetics, Tokyo,Japan, 1999 Vol.5:

350-355

[41]Zarit B D, Super B J, Quek F., Comparison of five color models in skin pixel classification. In: Proc. Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time systems, Corfu, Greece, 1999, 58-63

[42]周 杰,卢春雨,张长水等.人脸自动识别方法综述[J].电子学报, 2000, 28 (4) : 102- 106

[43]梁路宏,艾海舟等.人脸检测研究综述[J].计算机学报,2002,05:

[44]刘明宝,姚鸿勋,高文. 彩色图像的实时人脸跟踪方法[J]计算机学报, 1998,(06) .

[45]陆其明. DirectShow 开发指南[M]. 北京:清华大学出版社,2003

[46]谢亚光,章琦,刘济林. 基于 Microsoft DirectShow 的多媒体应用程序开发[J]. 计算机应用研究, 2003,(04) .

[47]卢春雨,张长水等. 基于区域特征的快速人脸检测法[J]. 清华大学学报(自然科学版),1999,(1)

[48]梁路宏,艾海舟等. 基于多关联模板匹配的人脸检测[J]. 软件学报,2001,(1)

[49]梁路宏,艾海舟. 基于模板匹配与人工神经网确认的人脸检测[J]. 电子学报,2001,(6)

[50]姜军,张桂林. 一种基于知识的快速人脸检测方法[J].中国图象图形学报,2002,(1)

[51]马勇,丁晓青. 基于层次型支持向量机的人脸检测[J]. 清华大学学报(自然科学版) 2003,(1)

[52]陈茂林,戚飞虎. 自组织隐马尔可夫模型的人脸检测研究[J].计算机学报 , 2002,(11) .

[53]张洪明,赵德斌,高文. 基于肤色模型、神经网络和人脸结构模型的平面旋转人脸检测[J]. 计算机学报 , 2002,(11) .

[54]邢昕,汪孔桥,沈兰荪. 基于器官跟踪的人脸实时跟踪方法[J].电子学报 , 2000,(06) .

[55]艾海舟,王栓,何克忠. 基于差分图象的人脸检测[J]. 中国图象图形学报,1998,(12).

[56]郭迎春,于明,林晓静,刘妍春. 基于运动和彩色信息的人脸定位方法[J].河北工业大学学报, 2002,(06) .

[57]施可为,傅锡天等. 含人脸的前景在活动图像序列中的分割[J]北京邮电大学学报 , 2000,(01) .

[58]艾海舟 ,肖习攀 ,徐光祐. 人脸检测与检索[J]计算机学报 , 2003,(07) .

[59]陈锻生,刘政凯. 肤色检测技术综述[J]计算机学报, 2006,(02) .

[60]张朝阳,潘保昌,郑胜林,彭绍湖. 基于消除背景的人脸定位方法[J]广东工业大学学报 , 2007,(02) .

[61]陈启泉,邱文宇,陈维斌. 标准正面人脸图象的特征提取[J].华侨大学学报(自然科学版), 2000,(4).

[62]Friedman N, Russell S. Image segmentation in video sequences: a probabilistic approach. In: Proceedings of Thirteenth Conference on Uncertainty in Artifcial Intelligence.Providence, Rhode Island, USA, Morgan Kaufmann Publishers, 1997. 175-181

[63]Wren C R, Azarbayejani A, Darrell T, Pentland A P.Pfnder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence,1997, 19(7): 780-785

[64] Stauffer C, Grimson W. Adaptive background mixture models for real-time tracking. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. FortCollins, Colorado, USA, IEEE, 1999. 246-252

[65]KaewTraKulPong P, Bowden R. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01. Kingston, UK, 2001.

[66]Zivkovic Z. Improved adaptive gaussian mixture model forbackground subtraction. In: Proceedings of the 17th International Conference on Pattern Recognition. Cambridge,United Kingdom, IEEE, 2004. 2: 28-31

[67] Horprasert T., Harwood D., Davis L.S. a statistical approach for real-time robust background subtraction and shadow detection.in IEEE ICCV'99 FRAME-RATE WORKSHOP. 1999.

[68]Biswajit Bose and Eric Grimson. Learning to Use Scene Context for Object Classification in Surveillance. Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), Nice, France, 2003.

[69] C. Palmer. The Fundamentals of Drawing No. 2. Vinciana, 1992

[70]刘亚,艾海舟,徐光佑. 一种基于背景模型的运动目标检测与跟踪算法[J]信息与控制 , 2002,(04) .

[71]朱孝政. 一种新的混合高斯模型的学习算法[J]航空计算技术, 2006,(04) .

[72]曾祥塑,葛元,王林泉.基于自适应阈值的动态手势分割和识别[J]电脑开发与应用 , 2005,(11) .

[73]王亮生,程荫杭. 一种改进的基于混合高斯分布模型的自适应背景消除算法[J]. 北方交通大学学报,,2003,(6)

[74]明英,边馥苓,蒋晶珏,王钢. 对光照、阴影和反光具有鲁棒性的变化检测算法及实现[J].计

算机工程与应用 , 2003,(24) .

[75]张懿慧,徐晓夏,陈泉林. 基于阴影抑制和自适应背景更新的车辆检测系统[J].上海大学学报(自然科学版) , 2005,(05) .

[76]肖梅,韩崇昭,张雷. 交通监控系统中基于多源信息融合的运动阴影检测[J]. 西安交通大学学报,2005,(10).

附录：部分源代码

```
#define HSV_V_MIN 30
#undef hsv_shift
#define hsv_shift 12
static const int div_table[] = {
      （略）
};

inline void CMyTracker::Color2HSV_8U1C_4img(IplImage * src,IplImage * dst_h,IplImage * dst_s,IplImage *
dst_v,IplImage * gray,IplImage * mask)
{
      unsigned char * pSrcData,*pDst_h_Data,/**pDst_v_Data,*pDst_s_Data,*/*pGrayData;
      int src_step,dst_h_step,/*dst_s_step,dst_v_step,*/gray_step;
      CvSize size;
      int x,y;

      if(!src)
            return ;
      if(gray)
            cvGetRawData( gray, (uchar**)&pGrayData, &gray_step, &size );
      else
      {
            pGrayData = 0;
            gray_step = 0;
      }
      if(dst_h)
            cvGetRawData( dst_h, (uchar**)&pDst_h_Data, &dst_h_step, &size );
      else
      {
            pDst_h_Data = 0;
            dst_h_step = 0;
      }
      cvGetRawData( src, (uchar**)&pSrcData, &src_step, &size );
      if (!pSrcData)
            return;
      for(y =0;y<size.height;y++,   pSrcData +=src_step,
                                          pDst_h_Data +=dst_h_step,
                                          pGrayData +=gray_step)
      {
            unsigned char * pSrc=pSrcData;
            unsigned char * ph=pDst_h_Data;
```

```
            unsigned char * pgray=pGrayData;
            for(x=0;x<size.width;x++,     pSrc+=3,
                                                ph++,
                                                pgray++)
            {
                int b, g, r;
                int h, /*s,*/ v;
                int vmin, diff;
                int vr, vg;
                b = pSrc[0], g = pSrc[1], r = pSrc[2];

                v = CV_IMAX( r, g );
                v = CV_IMAX( v, b );
                vmin = CV_IMIN( r, g );
                vmin = CV_IMIN( vmin, b );
                diff = v - vmin;
                if (v < HSV_V_MIN /*|| v > 250*/)
                {     h = 0;
                      v = 0;      }
                else
                {
                      vr = v == r ? -1 : 0;
                      vg = v == g ? -1 : 0;
                      h = (vr & (g - b)) + (~vr & ((vg & (b - r + 2 * diff)) + ((~vg) & (r - g + 4 * diff))));
                      h = ((h * div_table[diff] * 15 + (1 << (hsv_shift + 6))) >> (7 + hsv_shift))
                            + (h < 0 ? 30*6 : 0);
                }

                *ph = (unsigned char)h;
                v =b*sB + g*sG + r*sR;
                v = descale(v,shift);
                *pgray=CV_FAST_CAST_8U(v);
            }
      }
}

void CMyTracker::ReAllocImgMemory(IplImage *src)
{     （略）}

CMyTracker::CMyTracker()
{     (略)    }

CMyTracker::~CMyTracker()
{
```

```
    if(m_pHist)
        cvReleaseHist( &m_pHist );
    ReAllocImgMemory(NULL);
}

BOOL CMyTracker::TrainSkinModel(const char *pStrFileName)
{
    char FilePath[MAX_PATH+1];
    CString strSkinFile,strMaskFile;
    CStringList list,mask_list;
    char * pStrSetting = "Setting",* pStrPicDate = "PicData";
    int iSkinPicCount,i=0,j=0,result=0,error_flag=0,repeate;
    unsigned char mr,mg,mb;

    iSkinPicCount = GetPrivateProfileInt(pStrSetting,"SkinPicCount",0,pStrFileName);
    if (iSkinPicCount <= 0)
        return FALSE;
    mr = (unsigned char)GetPrivateProfileInt(pStrSetting,"Mask_R",0,pStrFileName);
    mg = (unsigned char)GetPrivateProfileInt(pStrSetting,"Mask_G",0,pStrFileName);
    mb = (unsigned char)GetPrivateProfileInt(pStrSetting,"Mask_B",0,pStrFileName);
    repeate = (unsigned char)GetPrivateProfileInt(pStrSetting,"Repeate",0,pStrFileName);
    for (i=1;i<=iSkinPicCount;i++)
    {
        strSkinFile.Format("Pic%d",i);
        result = GetPrivateProfileString(pStrPicDate,(LPCTSTR)strSkinFile,"",
            FilePath,MAX_PATH,pStrFileName);
        if (result==0)
        {
            error_flag = 1;
            break;
        }
        list.AddTail(FilePath);
        strMaskFile.Format("Mask%d",i);
        result = GetPrivateProfileString(pStrPicDate,(LPCTSTR)strMaskFile,"",
            FilePath,MAX_PATH,pStrFileName);
        if (result==0)
        {
            error_flag = 1;
            break;
        }
        mask_list.AddTail(FilePath);
    }

    if (error_flag)
```

```
{
	CString str;
	list.RemoveAll();
	mask_list.RemoveAll();
	str.Format("读取参数出错:Pic%d",i);
	AfxMessageBox(str,MB_OK);
	return FALSE;
}
if (list.GetCount() != mask_list.GetCount())
{
	CString str;
	list.RemoveAll();
	mask_list.RemoveAll();
	str.Format("图片个数不相等:%d",list.GetCount());
	AfxMessageBox(str,MB_OK);
	return FALSE;
}
cvClearHist( m_pHist);
POSITION pos,pos2;
strSkinFile = "";
strMaskFile = "";
pos = list.GetHeadPosition();
pos2 = mask_list.GetHeadPosition();
for( ; pos != NULL && pos2 != NULL; )
{
	strSkinFile = list.GetNext( pos);
	strMaskFile = mask_list.GetNext( pos2);
	IplImage* src = cvLoadImage((LPCTSTR)strSkinFile,1);
	IplImage* mask_src = cvLoadImage((LPCTSTR)strMaskFile,1);
	if (!src || !mask_src)
	{
		error_flag = 1;
		break;
	}
	IplImage* h_plane = cvCreateImage( cvGetSize(src), 8, 1 );
	IplImage* s_plane = cvCreateImage( cvGetSize(src), 8, 1 );
	IplImage* v_plane = cvCreateImage( cvGetSize(src), 8, 1 );
	IplImage* mask = cvCreateImage( cvGetSize(src), 8, 1 );
	IplImage* planes[] = { h_plane, s_plane, v_plane};

	unsigned char * pMaskData,* pSrcData,* pMask1CData;
	int mask_step,src_step,mask1c_step;
	CvSize size;
	int x,y;
```

```
        cvGetRawData( src, (uchar**)&pSrcData, &src_step, &size );
        cvGetRawData( mask, (uchar**)&pMask1CData, &mask1c_step, &size );
        cvGetRawData( mask_src, (uchar**)&pMaskData, &mask_step, &size );

        if (!pSrcData || !pMaskData)
        {
            error_flag = 1;
            break;
        }
        for(y =0;y<size.height;y++,    pMaskData +=mask_step,pSrcData +=src_step,
                                        pMask1CData +=mask1c_step)
        {
            for(x=0;x<size.width;x++)
            {
                if(    pMaskData[x*3] != mb ||
                    pMaskData[x*3+1] != mg ||
                    pMaskData[x*3+2] != mr)
                {
                    pSrcData[x*3] = 0;
                    pSrcData[x*3+1] = 0;
                    pSrcData[x*3+2] = 0;
                    pMask1CData[x] = 0;
                }
                else
                    pMask1CData[x] = 1;
            }
        }

        strMaskFile.Replace(".","f.");
        cvSaveImage((LPCTSTR)strMaskFile,src);
        cvCvtColor( src, src, CV_BGR2HSV );
        cvCvtPixToPlane( src, h_plane, s_plane, v_plane, 0 );
        for (j=0;j<repeate;j++)
            cvCalcHist( planes, m_pHist, 1, mask);

        if (src) cvReleaseImage(&src);
        if (mask_src) cvReleaseImage(&mask_src);
        if (h_plane) cvReleaseImage(&h_plane);
        if (s_plane) cvReleaseImage(&s_plane);
        if (v_plane) cvReleaseImage(&v_plane);
        if (mask) cvReleaseImage(&v_plane);
    }
    if (error_flag)
```

```
    {
        list.RemoveAll();
        mask_list.RemoveAll();
        AfxMessageBox("使用模板过滤象素出错",MB_OK);
        return FALSE;
    }
    float max_val = 0;
    cvGetMinMaxHistValue( m_pHist, 0, &max_val );
    cvScale( m_pHist->bins, m_pHist->bins, max_val ? 255. / max_val : 0. );
    return TRUE;
}

void CMyTracker::Color_Transorm()
{
Color2HSV_8U1C_4img(m_ImgSrc,m_pImgPlane_H,m_pImgPlane_S,m_pImgPlane_V,m_pImgSrcGray,m_pI
mgMask);
}

void CMyTracker::GetSkinProbImg()
{
    IplImage* planes[] = { m_pImgPlane_H};

    cvCalcBackProject(planes, m_pImgBackProject, m_pHist );
    if (m_Disp_ZeroSrc)
        cvZero( m_ImgSrc );
    if (m_DispCode == Tracker_Disp_Skin_Prob_Pixel)
    {
        cvCvtColor( m_pImgBackProject, m_ImgSrc, CV_GRAY2BGR );
    cvThreshold(m_pImgBackProject,m_pImgBackProject,m_skin_color_bin_threshold,255,CV_THRESH_BI
NARY);
    }
    else if (m_DispCode == Tracker_Disp_Skin_Bin_Pixel)
    {
    cvThreshold(m_pImgBackProject,m_pImgBackProject,m_skin_color_bin_threshold,255,CV_THRESH_BI
NARY);
        cvCvtColor( m_pImgBackProject, m_ImgSrc, CV_GRAY2BGR );
    }
    else
    {
  cvThreshold(m_pImgBackProject,m_pImgMask,m_skin_color_bin_threshold,255,CV_THRESH_BINARY);
    }
}

void CMyTracker::GetConnectComponet()
```

```
{
    int i;
    CvSeq* contour = 0 ,*pContour = 0;
    if(m_Storage)
        cvReleaseMemStorage( &m_Storage );
    if(m_ImgSrc)
        m_Storage = cvCreateMemStorage(0);

    if (m_do_morphological)
    {
        cvErode( m_pImgMask,m_pImgMask,0,2);
        cvDilate( m_pImgMask,m_pImgMask,0,2);
    }
    cvFindContours( m_pImgMask, m_Storage, &contour, sizeof(CvContour),
        CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE );
    m_SkinColorContour = contour;
    static int Color[]={CV_RGB(255,0,0),CV_RGB(0,255,0),CV_RGB(0,0,255),
                CV_RGB(255,255,0),CV_RGB(255,0,255),
                CV_RGB(0,255,255),CV_RGB(255,255,255)};
    for(i = 0,pContour = contour; pContour != 0; pContour = contour= contour->h_next,i++)
    {
        int color = Color[i%7];
        CvContour * p = (CvContour *)pContour;
        if(m_contour_filter)
        {
        if ((p->rect.width * p->rect.height > 40*40)
            ||(((p->rect.height/p->rect.width) > 1.0)&&((p->rect.height/p->rect.width) < 2.0 )))
            {
            if (m_DispCode == Tracker_Disp_Contour)
            {
                cvDrawContours( m_ImgSrc, pContour, color, color, -1, 1, 8 );
            }
            else if (m_DispCode == Tracker_Disp_Contour_Fill)
                cvDrawContours( m_ImgSrc, pContour, CV_RGB(0,255,0), CV_RGB(0,255,0), -1,
CV_FILLED, 8 );
            else
            {
                cvDrawContours( m_pImgMask, pContour, CV_RGB(255,255,255) ,
CV_RGB(255,255,255), -1, CV_FILLED, 8 );
            }
            }
        }
        else
        {
```

```
                if (m_DispCode == Tracker_Disp_Contour)
                {
                    cvDrawContours( m_ImgSrc, pContour, color, color, -1, 1, 8 );
                }
                else if (m_DispCode == Tracker_Disp_Contour_Fill)
                    cvDrawContours( m_ImgSrc, pContour, CV_RGB(0,255,0), CV_RGB(0,255,0) , -1,
CV_FILLED, 8 );
                else
                {
                    cvDrawContours( m_pImgMask, pContour, CV_RGB(255,255,255), CV_RGB(255,255,255),
-1, CV_FILLED, 8 );
                }
            }
        }
        if (m_skin_filter)
            cvAnd(m_pImgSrcGray,m_pImgMask,m_pImgSrcGray);
}

static int is_equal( const void* _r1, const void* _r2, void* )
{
        const CvRect* r1 = (const CvRect*)_r1;
        const CvRect* r2 = (const CvRect*)_r2;
        int distance = cvRound(r1->width*0.2);
        return r2->x <= r1->x + distance &&
                r2->x >= r1->x - distance &&
                r2->y <= r1->y + distance &&
                r2->y >= r1->y - distance &&
                r2->width <= cvRound( r1->width * 1.2 ) &&
                cvRound( r2->width * 1.2 ) >= r1->width;
}

void CMyTracker::DetectFace()
{
        IplImage* detect_image = m_pImgSrcGray;
        IplImage *temp = m_pImgCannyGray, *sum =m_ImgSum, *tilted = 0, *sqsum = m_ImgSqsum;
        IplImage *sumcanny = m_ImgSumCanny;
        CvSeq* face = 0;
        int i=0, scale = 1;
        if (!m_detect_face)
            return;
        if( m_do_pyramids )
        {
            cvPyrDown( m_pImgSrcGray, m_ImgGraySmall, CV_GAUSSIAN_5x5 );
            scale = 2;
```

```
        detect_image = m_ImgGraySmall;
        sum = m_ImgSumSmall;
        sqsum = m_ImgSqsumSmall;
        sumcanny = m_ImgSumCannySmall;
        temp = m_pImgCannyGraySmall;
    }

    int split_stage = 2;
    int npass = 2;
    int hid_cascade_count0;
    double factor;
    CvSize detect_image_size = cvGetSize(detect_image);
    CvSeq* seq = 0;
    CvSeq* seq2 = 0;
    CvSeq* idx_seq = 0;
    CvAvgComp* comps = 0;
    int ncomp;
    seq = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvRect), m_Storage );
    seq2 = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvAvgComp), m_Storage );
    face = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvAvgComp), m_Storage );

    if( m_FaceDetect_Cascade->has_tilted_features)
    tilted = cvCreateImage( cvSize(detect_image->width+1,detect_image->height+1), IPL_DEPTH_32S, 1 );
    cvIntegral( detect_image, sum, sqsum, tilted);
    cvCanny( detect_image, temp, 10, 50, 3 );
    cvIntegral( temp, sumcanny );

    hid_cascade_count0 = m_FaceDetect_Cascade->count;
    if( split_stage >= hid_cascade_count0 || split_stage == 0 )
    {
        split_stage = hid_cascade_count0;
        npass = 1;
    }

    for( factor = 1; factor*m_FaceDetect_Cascade->origWindowSize.width < MIN( detect_image_size.width/2,
detect_image_size.height/2); factor *= m_scale_factor)
    {
        double const ystep = MAX( 2, factor );
        CvRect equ_rect = { 0, 0, 0, 0 };
        int *p0 = 0, *p1 = 0, *p2 = 0, *p3 = 0;
        int *pq0 = 0, *pq1 = 0, *pq2 = 0, *pq3 = 0;
        int pass, stage_offset = 0;

        CvSize winSize = cvSize( cvRound( m_FaceDetect_Cascade->origWindowSize.width * factor ),
```

```
                                        cvRound( m_FaceDetect_Cascade->origWindowSize.height * factor ));
        int stopHeight = cvRound((detect_image_size.height - winSize.height - ystep) / ystep);
        cvSetImagesForHaarClassifierCascade( m_FaceDetect_Cascade, sum, sqsum, tilted, factor);
        cvZero(temp);

        if (m_do_canny_pruning)
        {
            equ_rect.x = cvRound(winSize.width*0.3);
            equ_rect.y = cvRound(winSize.height*0.3);
            equ_rect.width = cvRound(winSize.width*0.7);
            equ_rect.height = cvRound(winSize.height*0.7);

            p0 = (int*)(sumcanny->imageData + equ_rect.y*sumcanny->widthStep) + equ_rect.x;
            p1 = (int*)(sumcanny->imageData + equ_rect.y*sumcanny->widthStep)
                    + equ_rect.x + equ_rect.width;
            p2 = (int*)(sumcanny->imageData + (equ_rect.y + equ_rect.height)*sumcanny->widthStep) +
equ_rect.x;
            p3 = (int*)(sumcanny->imageData + (equ_rect.y + equ_rect.height)*sumcanny->widthStep)
                    + equ_rect.x + equ_rect.width;
            pq0 = (int*)(sum->imageData + equ_rect.y*sum->widthStep) + equ_rect.x;
            pq1 = (int*)(sum->imageData + equ_rect.y*sum->widthStep)
                    + equ_rect.x + equ_rect.width;
            pq2 = (int*)(sum->imageData + (equ_rect.y + equ_rect.height)*sum->widthStep) + equ_rect.x;
            pq3 = (int*)(sum->imageData + (equ_rect.y + equ_rect.height)*sum->widthStep)
                    + equ_rect.x + equ_rect.width;
        }
        m_FaceDetect_Cascade->count = split_stage;

        for( pass = 0; pass < npass; pass++ )
        {
            for( int yInt = 0; yInt < stopHeight; yInt++ )
            {
                int iy = cvRound(yInt*ystep);
                int xIntStep = 1;
                int xInt, stopWidth = cvRound((detect_image_size.width - winSize.width - 2*ystep) / ystep);
                char* mask_row = temp->imageData + temp->widthStep * iy;

                for( xInt = 0; xInt < stopWidth; xInt += xIntStep )
                {
                    int ix = cvRound(xInt*ystep);
                    if( pass == 0 )
                    {
                        int result;
                        xIntStep = 2;
```

```
                    if( m_do_canny_pruning )
                    {
                        int offset;
                        int s, sq;

                        offset = iy*(sumcanny->widthStep/sizeof(p0[0])) + ix;
                        s = p0[offset] - p1[offset] - p2[offset] + p3[offset];
                        sq = pq0[offset] - pq1[offset] - pq2[offset] + pq3[offset];
                        if( s < 100 || sq < 30 )
                            continue;
                    }
                    result = cvRunHaarClassifierCascade( m_FaceDetect_Cascade, cvPoint(ix,iy),0);
                    if( result > 0 )
                    {
                        if( pass < npass - 1 )
                        {
                            mask_row[ix] = 1;
                        }
                        else
                        {
                            CvRect rect = cvRect(ix,iy,winSize.width,winSize.height);
                            cvSeqPush( seq, &rect );
                        }
                    }
                    if( result < 0 )
                        xIntStep = 1;
                }
                else if( mask_row[ix] )
                {
                    int result = cvRunHaarClassifierCascade( m_FaceDetect_Cascade,
cvPoint(ix,iy),stage_offset );

                    if( result > 0 )
                    {
                        if( pass == npass - 1 )
                        {
                            CvRect rect = cvRect(ix,iy,winSize.width,winSize.height);
                            cvSeqPush( seq, &rect );
                        }
                    }
                    else
                        mask_row[ix] = 0;
                }
            }
```

```
            }
            stage_offset = m_FaceDetect_Cascade->count;
            m_FaceDetect_Cascade->count = hid_cascade_count0;
        } // end pass
    } // end factor
    if(m_min_neighbors)
    {
        ncomp = cvPartitionSeq( seq, 0, &idx_seq, is_equal, 0, 0 );
        comps = (CvAvgComp*)cvAlloc( (ncomp+1)*sizeof(comps[0]));
        memset( comps, 0, (ncomp+1)*sizeof(comps[0]));

        for( i = 0; i < seq->total; i++ )
        {
            CvRect r1 = *(CvRect*)cvGetSeqElem( seq, i );
            int idx = *(int*)cvGetSeqElem( idx_seq, i );
            assert( (unsigned)idx < (unsigned)ncomp );

            comps[idx].neighbors++;

            comps[idx].rect.x += r1.x;
            comps[idx].rect.y += r1.y;
            comps[idx].rect.width += r1.width;
            comps[idx].rect.height += r1.height;
        }

        for( i = 0; i < ncomp; i++ )
        {
            int n = comps[i].neighbors;
            if( n >= m_min_neighbors )
            {
                CvAvgComp comp;
                comp.rect.x = (comps[i].rect.x*2 + n)/(2*n);
                comp.rect.y = (comps[i].rect.y*2 + n)/(2*n);
                comp.rect.width = (comps[i].rect.width*2 + n)/(2*n);
                comp.rect.height = (comps[i].rect.height*2 + n)/(2*n);
                comp.neighbors = comps[i].neighbors;
                cvSeqPush( seq2, &comp );
            }
        }

        for( i = 0; i < seq2->total; i++ )
        {
            CvAvgComp r1 = *(CvAvgComp*)cvGetSeqElem( seq2, i );
            int j, flag = 1;
```

```
                for( j = 0; j < seq2->total; j++ )
                {
                    CvAvgComp r2 = *(CvAvgComp*)cvGetSeqElem( seq2, j );
                    int distance = cvRound( r2.rect.width * 0.2 );

                    if( i != j &&
                        r1.rect.x >= r2.rect.x - distance &&
                        r1.rect.y >= r2.rect.y - distance &&
                        r1.rect.x + r1.rect.width <= r2.rect.x + r2.rect.width + distance &&
                        r1.rect.y + r1.rect.height <= r2.rect.y + r2.rect.height + distance &&
                        (r2.neighbors > MAX( 3, r1.neighbors ) || r1.neighbors < 3) )
                    {
                        flag = 0;
                        break;
                    }
                }
                if( flag )
                {
                    cvSeqPush( face, &r1 );
                }
            }
        }
        else
            face = seq;

        for( i = 0; i < face->total; i++ )
        {
            if (m_DispCode == Tracker_Disp_FaceDetect_Rectangle)
            {
                CvRect face_rect = *(CvRect*)cvGetSeqElem( face, i, 0 );
                cvRectangle( m_ImgSrc, cvPoint(face_rect.x*scale,face_rect.y*scale),
                                cvPoint((face_rect.x+face_rect.width)*scale,
                                        (face_rect.y+face_rect.height)*scale),
                                CV_RGB(255,0,0), 1);
            }
        }
        m_FaceRectTotalNum += i;
}
```