华南理工大学

South China University of Technology

本科毕业设计（论文）

题目：基于 Android 平台的照片处理

学　院　　电子与信息学院

专　业　　信息工程

学生姓名　　黄彬

学生学号　　200930243373

指导教师　　王伟凝

提交日期　　2013 年 5 月 27 日

摘　要

本文研究了基于 Android 平台的数字照片美化算法，依据构图准则设计了照片的构图优化算法，实现了对照片构图的优化调整。针对人脸和照片的颜色、亮度等视觉特征，设计了一些图像特效，使照片具有更好的美感。

本文针对单主体构图和垂直构图的照片类型提出了一种构图优化算法。对于单主体构图照片，采用显著区域检测和 mean-shift 分割结合的方法提取出主体，利用基于样例的图像修复算法进行背景修复，根据三分法则原理实现了照片的三分法则优化。对于垂直构图照片，我们提出了一个水平线检测算法可以提取出照片中水平线所在的位置，然后根据视觉平衡法则，我们利用图像空间重构算法和图像修复算法实现了照片的视觉平衡优化。我们建立了一个由 93 张照片构成的照片数据库，在我们的实验中，主体区域提取的成功率是 81%，水平线检测算法的成功率是 91.2%，这两个算法直接影响了构图优化的结果。因此我们认为我们提出的方法是有效的。本文还针对人脸设计了瘦脸，肖像漫画等特效算法以及针对照片的视觉特征实现了怀旧，LOMO 等多种风格化滤镜。最后，我们介绍了如何在 Android 平台上进行 NDK 和 OpenCV 的开发，并且实现了本文提出的所有算法，设计出了一个基于 Android 平台的照片处理工具。

关键词：构图优化；照片特效；图像处理

Abstract

In this paper, we studies the digital photo beautification algorithm on Android platform, propose the composition optimization algorithm based on the Composition rule. Aiming at the face and the visual features of photo such as color and brightness,we designed some image effects to make the photo has a better sense of beauty.

Composition optimization algorithm is proposed in this paper, mainly is suitable for two types of composition,the single-subject composition and vertical composition. For the photos with single-subject composition, we use saliency detection and segmentation of mean-shift algorithm to extract the subject, fill region and remove object by Exemplar-Based image inpainting, and then optimize photo composition based on the rule of thirds. For the photo width vertical composition, we propose a horizontal line detection algorithm can extract the horizontal line, then according to the visual balance rule, we use image space reconstruction algorithm and image inpainting algorithm to optimize photo composition. We construct an experiment image database of 93 images. in our experiments,the success rate of The main region extraction is 81%, the success rate of the horizontal line detection algorithm is 91.2%, so we think that our method is effective. This paper also designs some image effects on human face,such as face-lift and whitening and realize some image filters based on the visual features of photos, such as LOMO and Nostalgic filter. Finally, we introduce how to develop NDK and OpenCV on the Android platform, and implements all algorithms proposed in this paper.

Keyword: composition optimization,image effects ,image process

目录

第一章　引言

1.1 本文背景及意义

摄影是人们发现美，记录美的途径。随着无线宽带技术和移动终端技术的飞速发展，移动互联网应运而生并迅猛发展。各种移动社交应用如微信，微博，陌陌等也迅速发展。而移动社交又以文字，图片，声音，视频的形式表现出来，其中尤以文字和图片为主。人们可以在自己的社交圈子里上传和分享自己的照片。因此，人们希望能有好的工具来修复和美化照片。但是拍摄出一张美感的照片对一个业余爱好者来说并不是一件容易的事，甚至常常有一些不受人喜欢的意外对象进入镜头。这时候我们就需要一些工具来修复照片，并且移除我们不喜欢的对象，使照片更具美感。因此，有必要设计出一个基于移动终端平台的，方便实用的照片处理工具。

照片的美感跟照片的构图、亮度、颜色等视觉特征有关。构图是照片中图形元素的组合，它对照片的视觉感知非常重要，在很大程度上决定了照片的美感。好的构图可以清楚的向观众展示照片的主题，并有效地表达摄影者的感受。但是，构图也是人们拍摄时很难掌握的技巧，这就限制了照片修复的空间。因此，基于构图美学，本文提出了一种构图优化方法。而对于照片的亮度、颜色等视觉特征，本文实现了一些图像特效算法。

移动终端是移动互联网的重要载体，它在移动互联网中扮演的角色越来越重要。目前的移动终端主要以 Android 系统，苹果的 IOS，以及微软的 Windows Phone 平台为主。Android 是一个以 Linux 为基础的开放源码操作系统，由 google 和其主导的开放手机联盟发展。Android 凭借其开放性和良好的用户体验迅速占领了市场。据统计，目前 Android 在全球市场的占有率已经达到 60%，而在中国市场的占有率更是已经超过了 70%。因此，本文研究了基于 Android 平台的照片美化技术，设计并实现了一个实用的照片处理工具。

1.2 照片处理的研究现状

随着数码相机的普及和互联网的快速发展，可以访问的照片数量爆炸式增长，图像美学成为计算机图像技术的一个热门研究领域。图像美学是研究如何评估一张照片是否具有美感以及如何制作出更具美感的照片。目前人们对于影响照片美感的视觉特征已经有一些共识。人们认为照片的构图、颜色、亮度等视觉特征是影响照片美感的重要因素。高品质的照片一般满足三个原则：一个明确的主题，集中最为关注的子对象，移除分散注意力的子对象。摄影师通过巧妙的控制照片的构图，亮度和子对象的聚焦来努力实现这些原则。但是大多数人缺少摄影的专业知识，这就需要有一个工具可以对照片的构图、颜色等视觉特征进行调整，使照片更具美感。

目前基于 Android 平台的照片处理工具已有不少，但是大多数工具都着重于从照片的色彩、颜色等特征方面对照片进行美化。而对影响照片视觉感知至关重要的构图却无能为力，这也是摄影者最难把握的技术，因此照片修复的效果相对有限。此外，移动终端的处理器性能相对较弱，而图片又是大数据的集合，用 Java 语言来进行图片处理效率通常不令人满意，这就要求 Android 平台的照片处理尽可能地使用 NDK 开发，用 C/C++语言来实现。

1.3 本文主要研究工作

本文的目标是设计出一个有效的构图优化方法，能对照片的构图进行优化，使照片更具美感，并根据照片的色彩、亮度等视觉特征实现一些图像特效算法，从多方面提升照片的美感，最终设计并实现一个基于 Android 平台的照片处理工具。

本文的主要研究工作有：

1) 探讨并实现了主体区域提取和图像修复算法，根据构图准则提出了三分法则构图优化方法。
2) 研究并实现了水平线检测和图像空间重构算法，根据构图准则提出了视觉平衡优化方法。
3) 针对人脸设计了一些特效算法，针对影响照片美感的颜色、亮度等视觉特征实现了一些滤镜。
4) 探讨了 Android 平台 NDK 和 OpenCV 的开发，实现了本文提出的所有算法，设计了一个基于 Android 平台的照片处理工具。

1.4 本文结构

本文结构如下：第二章介绍了常见的构图法则，主体区域检测和分割算法，研究和改进了现有的图像修复算法和图像空间重构算法，提出了一种有效的水平线检测方法，最后综合利用这些方法提出了两种有效的构图优化算法，并用实验验证了算法的可行性。第三章针对人脸以及照片的颜色、亮度等视觉特征实现了一些实用的图像特效算法。第四章探讨了 Android 平台 NDK 和 OpenCV 的开发以及照片处理软件的设计。

第二章　构图优化

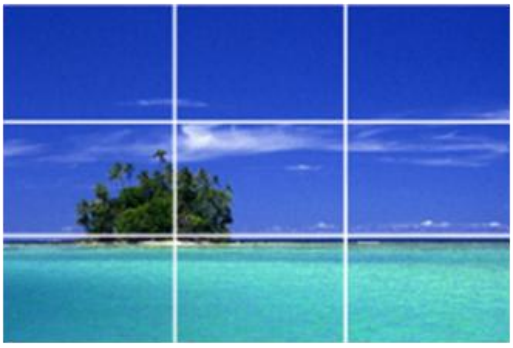
构图是照片内所有图形元素的组织，好的构图可以清楚的向观众展示照片的主题，并有效地表达摄影者的感受。但是大多数人拍摄出的照片通常对构图准则掌握不好，在本章，我们将会详细探讨如何对照片进行构图优化，使它更具美感。

2.1 构图法则

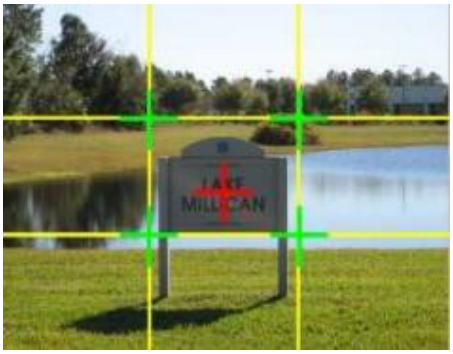
专业摄影师在拍摄照片的时候通常遵循一些构图法则，这使他们拍摄出来的照片比业余爱好者有更强的视觉效果。已经有很多研究总结了一些构图法则被证实能够使拍摄出来的照片质量更好。这些法则通常基于一些简单的规则，如三分法则，视觉平衡法则。

1、三分法则

三分法则是最知名的构图法则之一。它用两条水平线和垂直线把一张照片划分成相等的九部分。如图 2-1（a），由这四条直线形成的四个相交点被认为是力量点，摄影师都尽力把主体对象放置在这些点的附近而不是放置在照片的中心，如图 2-1（b）。



a）三分法则放置主体的照片



b）不符合三分法则的照片

图 2-1 三分法则构图示例

2、视觉平衡法则

视觉平衡法则也是常见的一种构图法则，它认为，在垂直构图中，水平线把照片分成两个区域，这两个区域的面积比应该为黄金分割比例（0.618），即水平线是照片的黄金分割线。如图 2-1（a），用设天空区域的长度为Y_k（即图中绿色直线的长度），海洋区域的长度为Y_g（即图中黄色直线的长度）。按照视觉平衡法则满足以下公式：

$$\frac{Y_g}{Y_k}=\frac{Y_k}{Y_k+Y_g} \qquad (2\text{-}1)$$

a）不符合视觉平衡的照片　　b）符合视觉平衡的照片

图 2-2 视觉平衡构图示例

主体是照片中摄影师主观上想要突出表现的对象，比如照片中的人物或者动物，或者某个特殊的物件。主体是照片的核心，并且主体在照片中的位置很大程度上决定了照片的构图类型。因此，对照片进行构图优化，需要有适当的方法提取出主体区域。而我们在对照片进行构图优化的时候，主要是改变主体的大小和位置，这时我们就需要一个有效的图像修复算法来修复空缺的区域。水平线也是一种重要的构图元素，尤其在垂直/水平构图中。在对这类构图的照片进行优化的时候，我们希望可以调整水平线的位置，这就需要一个有效的水平线检测算法能够检测出水平线的位置，以及有效的图像空间重构算法可以对照片的特定区域进行拉伸。下面，我们将依次介绍这些算法。

2.2 主体区域检测和提取

在本节中，我们将详细介绍两种主体区域提取方法，一种是交互式的，一种是自动的。交互式的方法适用于大多数的情况，但需要手动标记前景和背景。而另一种方法比较适用于主体比较明确、背景较为简单的照片，但是它比较便捷。

2.2.1 交互提取主体区域

OpenCV 图像处理库实现了一种经典的显著区域提取算法，即 GrabCut 算法[1]。该算法利用用了图像中的颜色信息和边界信息，只要少量的用户交互操作即可得到比较好的分割结果。如果前景和背景之间的颜色反差不大，分割的效果不好；不过，这种情况下允许手工标记一些前景或背景区域，这样就能得到较好的结果。这个算法依赖于 Windows 窗口，我们在 Android 平台上改写了它，并且获得了很好的效果。GrabCut 算法的流程为：

1) 用矩形窗或掩码图像初始化 grabCut。

2) 标记照片中的前景和背景。执行分割。

3) 执行一次分割。

4) 如果对结果不满意，回到 2)；

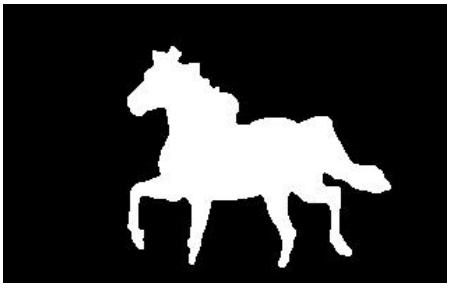
5) 得到主体掩码。

如图 2-3 所示。（b）中画出了矩形区域，用红色标记前景，蓝色标记背景。（c）是一次分割后的结果。对于大多数照片，一次分割即可获得很好的效果。（d）是提取出的主体掩码。

a）原图

b）画出矩形区域并标记前后景色

c）执行分割



d）得到的主体掩码

图 2-3 主体区域交互提取过程

2.2.2 自动提取主体区域

为了更方便用户，我们在本节将详细探讨如何智能地提取主体区域。提取主体区域必须实现两个关键点——显著区域检测和图像分割。

1、显著区域检测

显著区域检测得到显著图，一个显著图应当具备以下的一些特点：

a) 可以突出最大的主体对象。

b) 可以高亮整个显著区域。

c) 显著对象有明显的边界。

d) 可以消除来自文理，噪音等的高频影响。

我们采用了一种基于 LAB 颜色空间的显著区域检测算法来产生显著图[2]，它利用照片的颜色和亮度特征计算显著性。对于一张照片 I ，它的显著图 S 满足如下公式：

$$S(x,y)=\|I_u - I_w(x,y)\| \qquad (2\text{-}2)$$

其中，I_u 是照片所有像素的平均值，而 I_w 是原照片高斯模糊后的照片，这样可以消除纹理细节和噪音等的影响。I_w 和 I_u 都用 LAB 颜色空间表示，我们通过计算 I_w 和 I_u 的欧几里德距离来计算显著性。

如图 2-4 所示是一张照片和根据（2）得到的显著图。

a）原图　　b）得到的显著图

图 2-4 显著图示例

2、图像分割

得到显著图后还要进行图像分割才能提取出照片的主体区域，在这里我们利用了上一步中计算的显著图，结合成熟的 mean-shift 分割算法提取出了主体区域。具体流程如下：

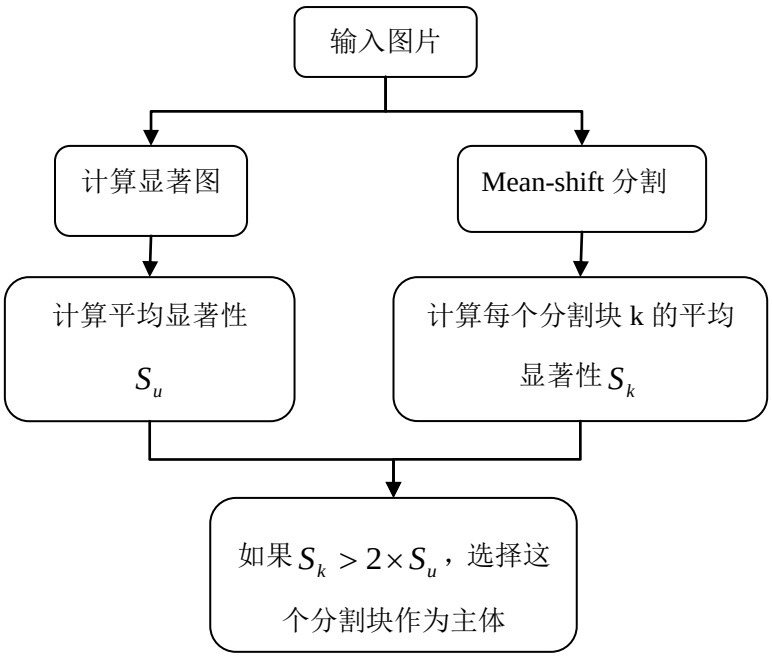


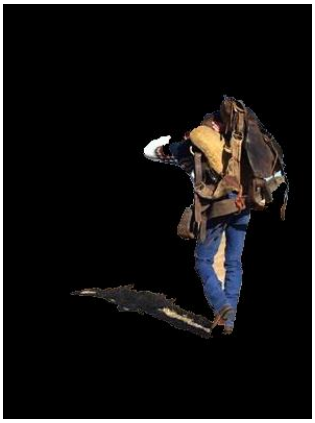
图 2-5 自动提取主体流程

结果如图 2-6 所示。（b）是用显著区域检测后得到的显著图，（c）是对原图 mean-shift 分割后的结果，（d）是按照图 2-5 的流程提取出的主体区域。

a）原图

b）得到显著图

c）mean-shift 分割



d）提取主体

图 2-6 自动提取主体示意图

2.3 图像修复

我们可以通过改变主体的大小和位置，对照片进行构图优化。这时我们就需要一个有效的图像修复算法来修复空缺的区域。

图像修复是图像处理中一个非常重要的组成部分。为了修复照片中损坏的部分，人们常常需要花费大量的手工工作，而且需要经验丰富的人来处理。这就促进了数字图像技术的发展，如今数字图像修复技术已不仅用来修复照片的损坏部分，还常常被人们用来移除照片中不喜欢的对象。目前数字图像修复技术主要集中在两个方面，基于像素点的方法和基于像素块的方法。基于像素点的方法比较适合修复小面积区域，常见的有 PDE 模型，CDD 模型等。基于像素块的算法比较适合修复面积较大的区域，常见的有基于纹理合成的修复算法和基于样例的修复算法。

在本节中，我们采用了一种基于样例的图像修复算法[3]，并且根据我们的需求做了一

些改进。

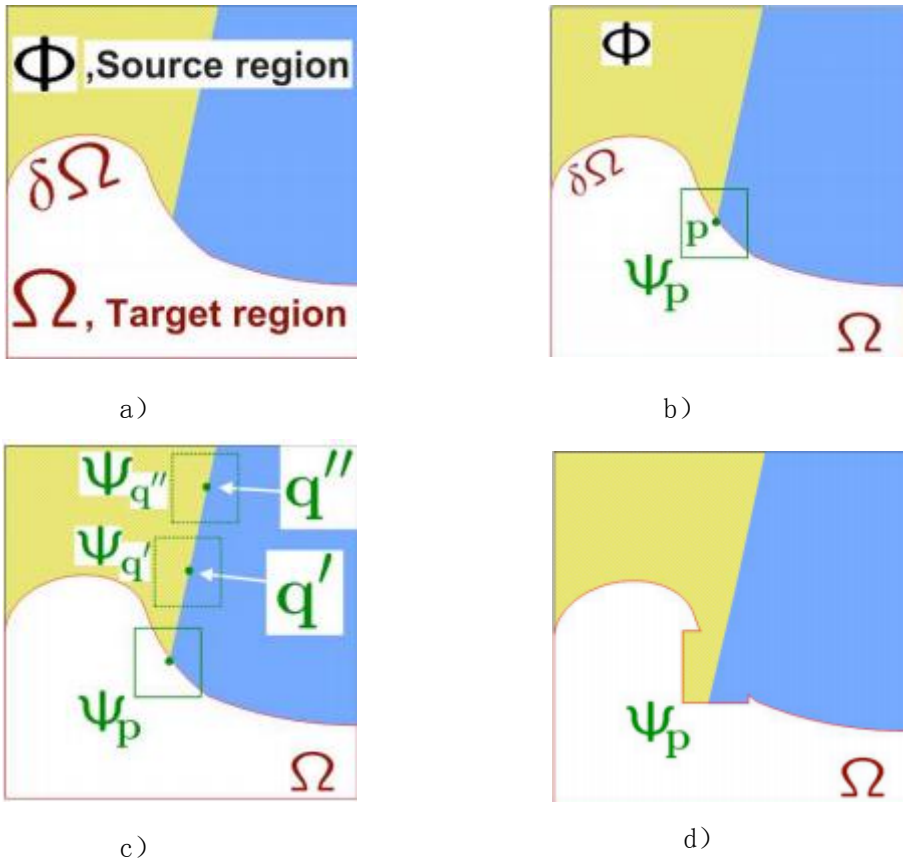


图 2-7 基于样例修复示意图

在图 2-7 中，Ω是修复的目标区域，$\delta\Omega$是分界线，也是填充的前端。ϕ是图像的原始区域，给填充过程提供样例。设像素块$\psi_p \in \Omega$，$\psi_{q'} \in \phi$。ψ_p是要被填充的像素块，它的最佳匹配块是在ϕ中与ψ_p中非填充区域最相似的像素块，即$\psi_{q'}$。我们用$\psi_{q'}$填充ψ_p，即完成一次修复过程。此外，填充的顺序也对结果有很大的影响，我们认为处于光照度线上并且具有高置信因子的像素块应该先被填充。如图 2-8 所示，修复的具体过程如下：

（1） 提取待修复区域Ω和原始区域ϕ的边界线$\delta\Omega$。

（2） 对$\forall p \in \delta\Omega$，计算他的优先权$P(p)$，优先权可由如下公式计算：

$$P(p) = C(p)D(p) \tag{2-3}$$

其中，$C(p)$是置信因子，$D(p)$是数据因子，他们可以用如下公式计算：

$$C(p)=\frac{\sum_{q\in\psi_p\cap(\tau-\Omega)}C(q)}{|\psi_p|},D(p)=\frac{|\nabla I_p^{\perp}\bullet n_p|}{\alpha} \qquad (2\text{-}4)$$

其中，$|\psi_p|$是ψ_p的面积，α是权重，一般取$\alpha=255$。n_p是$\delta\Omega$上的点p的法向量，$\nabla I_p^{\perp}$是点p的光照度线。在初始化的时候，$\forall p\in\Omega$，$C(p)=0$。$\forall p\in\tau-\Omega$，$C(p)=1$。

（3） 找到优先权最大的像素块ψ_p。

（4） 在ϕ中找到与ψ_p均方差最小的像素块ψ_q，即为最佳样例。

（5） 用ψ_q替换ψ_p中待修复的目标区域。

（6） 更新ψ_p中原来待修复区域的置信因子。

（7） 重复以上步骤，直到$\Omega=\phi$。

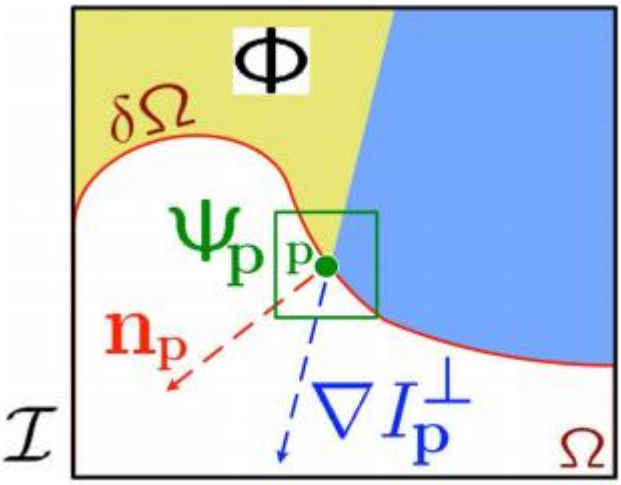


图 2-8 样例示意图

但是，以上算法仍然存在一些问题：

1) 算法太过耗时。由于需要在整个原始区域搜索最佳匹配块，计算复杂耗时。
2) 用搜索到的像素块直接替换目标像素块，但窗口过大时可能产生马赛克的效果，而窗口过小相似度的计算又会不准确。

针对以上的问题，我们提出了以下的解决方法：

1) 实际上，最佳匹配块一般都在目标区域的附近，我们可以设置一个搜索半径，只对半径内的像素块进行计算。
2) 在搜索到最佳样例时，我们不是直接替换而是以较小的窗口半径进行替换。这样在搜索的时候我们可以保证相似度的准确性，在替换的时候又可以避免出现马赛克。

在我们的实验中，我们设置搜索半径 $R=50$，像素块的窗口为 8×8，替换窗口为 4×4。

图像修复算法的应用很广，可以用在对象移除，背景修复，去水印等方面。图 2-9 是应用图像修复算法的一些例子。（a）中我们移出了照片中远处的几个我们不喜欢的人物，只保留了女孩和狗。（b）中我们去除了照片右下角的水印，并且用背景修复了这些区域。这些都让照片的主题突出，更具美感。

a）对象移除

b）去水印

图 2-9 修复示例

2.4 水平线检测

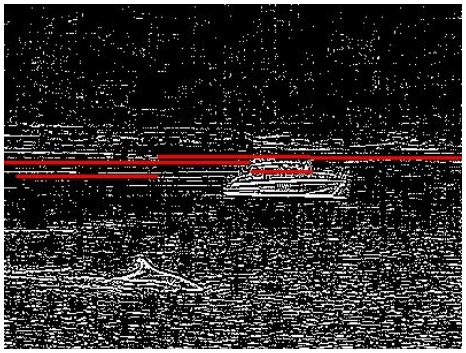
在垂直/水平构图中，水平线是一种重要的构图元素。在对这类构图的照片进行优化的时候，我们希望可以调整水平线的位置，这就要求我们能够检测出水平线的位置。在这里我们提出了一种水平线检测算法，它利用了水平线的长度特征和密度比特征。直线把一张图片分成两个区域，图片二值化后，我们把两个区域非零像素点的个数之比定义为这条直线的密度比。如果一条直线的密度比越大，长度越长，那么它便更可能是水平线。基于这样的思想，我们在前期尽可能地检测出所有的直线，在后期中通过合并相似直线，计算权

重等方式选出最佳直线作为水平线。具体方法如下：

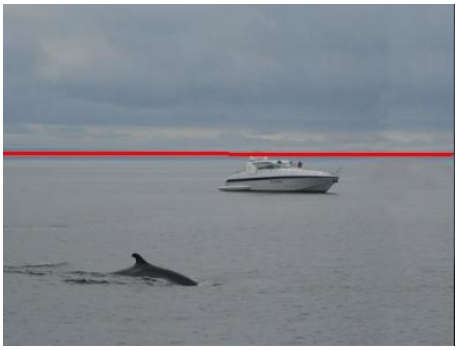
1) 对输入照片二值化，自适应阀值分割，得到边缘检测后的照片。
2) 利用霍夫变换检测出（1）中的所有直线。
3) 合并相似直线，把满足 dRho < DELTA_RHO && dTheta < DELTA_THETA 的所有直线合并成一条直线。dRho 和 dTheta 是两条直线 Rho 和 Theta 的差值。在我们的实验中，DELTA_RHO=1, DELTA_THETA=1/114.36。
4) 计算直线的密度比 densityRatio 和长度 length。
5) 归一化，给密度和长度加权，求得总的分数，并按从高到低排序。计算方法如下：
   Sum = densityRatio * WEIGHT_DENSITY + length * WEIGHT_LENGTH;
   在我们的实验中，WEIGHT_DENSITY=3/4, WEIGHT_LENGTH=1/4。
6) 选出分数最高的直线，并判断角度，大于 10 度放弃，否则我们认为是水平线。

如图 2-10 是对一张照片水平线检测的过程。

a）原图



b）阀值分割合并直线后



d）选出水平线

图 2-10 水平线检测示意图

2.5 图像空间重构

在上一小节，我们讨论了如何检测出水平/垂直构图中的水平线，要实现水平/垂直构图

调整，需要我们在调整水平线的同时不损坏图像的主体内容。在这一小节，我们采用了一种图像空间重构算法——插缝法[4]，并根据我们的需求进行了改进。

图像空间重构最初的目的是为了屏幕适配，手机，平板，电视，电脑等他们的屏幕尺寸和分辨率都不一样，为了在这些设备上显示相同的内容，需要对图片进行拉伸和缩小，又不能对原来的内容造成太大的破坏。于是一些基于内容重要性的图像空间重构算法应运而生，插缝法便是其中的一种。

插缝法的基本思想是，保留图片中重要的像素点不变，删除或者插入不重要的点。给像素点设置一个能量函数，越重要的点能量越大，在删除或者插入像素点的时候尽可能地避开能量比较高的点。能量函数一般可用边缘检测实现，在我们的实验中，我们用一对 sobel 算子的梯度运算作为能量函数。因为拉伸和缩小的原理是相同的，所以在此我们只讨论插缝法缩小的过程：

1) 首先，获得图片新的尺寸大小。
2) 获得每个像素点的能量，得到能量图，可以通过边缘检测实现。
3) 每次删除一条缝，选取能量值之和最小的路径，把它删除，缝必须是连续的。
4) 删除缝后图片大小改变，对新的图片重复以上步骤，直到达到目标大小。

但是，这个方法仍然存在一些问题：

1) 每次只删除一条缝，要对新的图片重新获取能量，比较耗时。
2) 在整个图片中搜索最佳缝，但有时我们并不希望这样。因为我们在调整构图的时候只需要调整照片的一个区域。

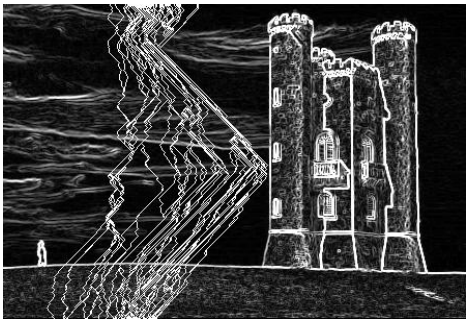
我们提出了一些简单的方法来解决上面的问题。

1) 一次删除多条缝，把找到的最佳缝隙暂存起来，同时提升他的能量，这样下一次就不会再找到他了。找到多条缝之后同时删除，注意可能存在相交点，同一个点不能删除两次（但可以插入两次），这时可删去他的临近点。
2) 设置一个插缝区间，通过限制入口点来实现在一定的区域内插缝。我们还可以传入一个区域掩码，把需要保护的区域全部置成白色，使这个区域的能量最高。这样我们在插缝的时候就可以避开这些区域，从而保护了这些区域。

图 2-11 是插缝法拉伸图片的一个例子，他保留了照片的主体没有遭到破坏。我们利用一对 sobel 算子对照片进行边缘检测得到了图 2-11（b）中的能量图，在图 2-11（c）中，我们把缝的入口点限制在人和城堡之间，使缝避开了主体区域。（d）是拉伸后的照片。而图 2-12 是原方法和我们改进后的方法的缩小效果对比图。（a）是原方法的效果，由于无法控制插缝的范围，照片的所有位置都可能被插缝，因此主体不可避免的遭到了一定程度的破坏，如红色标注的区域。（b）中我们限制了插缝区间，只在人和城堡之间进行插缝，因此只是缩短了人和城堡之间的距离，因此有效地保护好了主体。

a）原图

b）能量图



c）限制区间，并插入多条缝

d）原图拉伸后

图 2-11 图像空间重构示意图

a）原方法

b） 改进后

图 2-12 缩小效果对比

2.6 优化构图

在本章的前面几个小节，我们探讨了构图优化所必需实现的几个算法，在本节，我们将详细研究如何利用前面的算法来优化照片构图。

2.6.1 三分法则优化

三分法则是我们最熟知的构图法则，也是生活中大多数照片，尤其是单主体的照片一般采用的构图法则。图 2-13 是一些利用我们开发的图像工具三分法则优化构图的例子，我们可以看出，优化后的照片具有更好的美感。

a）原图

b）优化后

c）原图

d）优化后

图 2-13 三分法则优化示例

我们知道，当主体对象位于照片的四个力量点之一时，能够更加突出地表现主体，并且具有更好的美感。那么在优化照片构图的时候，我们希望能把主体对象移动到这些力量点的附近。此外，主体区域在照片中所占的比例也会影响我们的视觉感受，特别是当比例小于 0.1 时[3]，我们很难获得好的美感。这时候我们就需要改变主体的大小。

我们已经知道照片中四个力量点的位置，但是我们如何改变主体的大小？Ligang Liu 等人[3]对一个庞大的图片数据库做过统计，他们认为，当主体区域所占的比例在 0.1，0.56 和 0.8 时，能够获得更好的视觉美感。

于是我们提出了下面的一种利用三分法则优化照片构图的方法：

（1） 用 2.2 中提出的主体区域检测和分割方法提取出照片中的主体。

（2） 对主体掩码中值滤波并膨胀，消除砂眼和残留的边界痕迹。

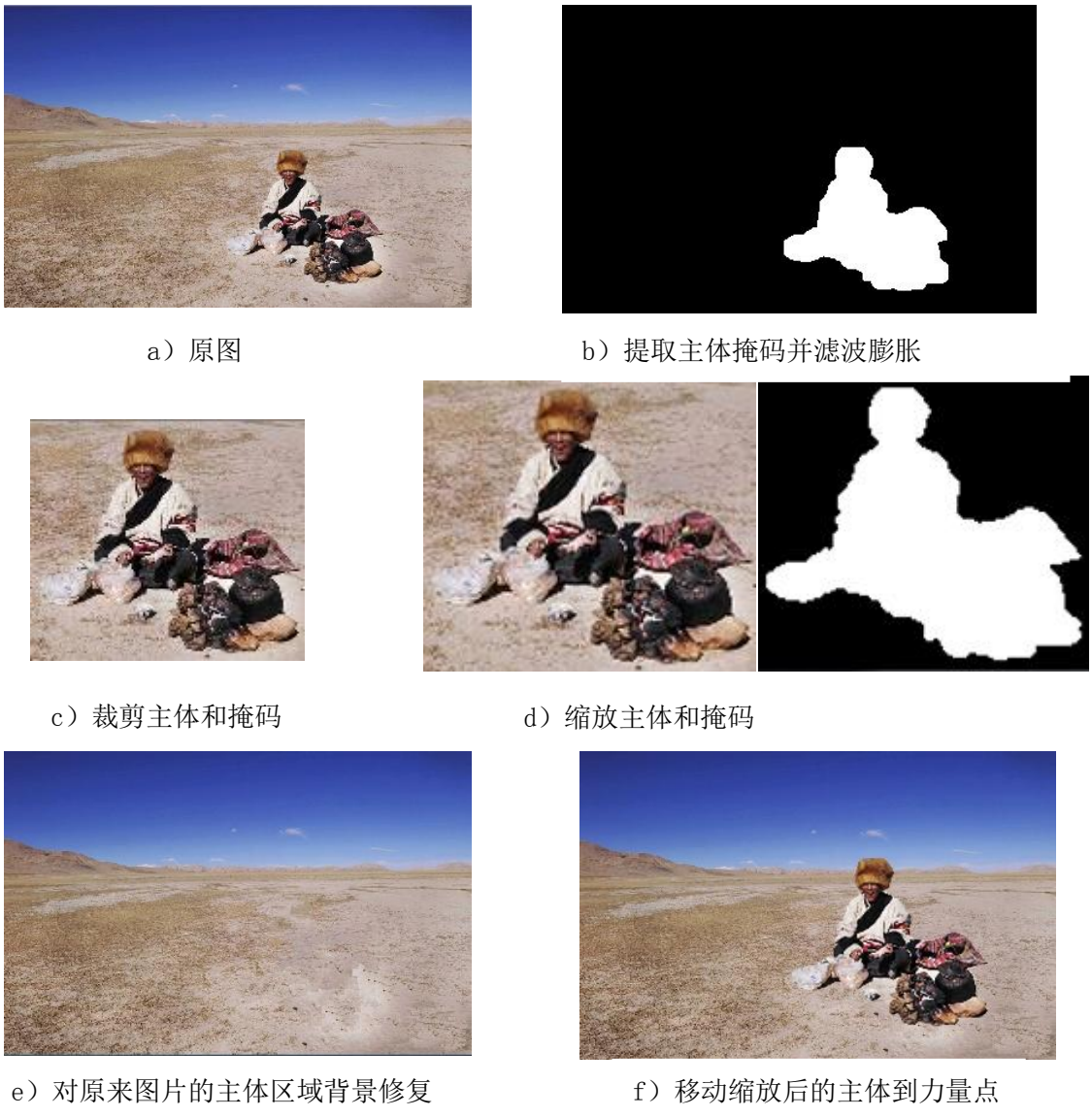
（3） 由主体掩码可以得到主体的尺寸，根据这个尺寸裁剪原图中的主体和主体掩码。

（4） 把主体和主体掩码缩放到合适的比例，即缩放到 0.1，0.56 或 0.8 中最接近的一个比例。

（5） 用 2.3 中提出的图像修复算法背景填充原来的主体区域。

（6） 把缩放后的主体质心移动到（5）中距离最近的力量点。

图 2-14 是利用三分法则优化照片构图的整个过程。



a）原图　　b）提取主体掩码并滤波膨胀

c）裁剪主体和掩码　　d）缩放主体和掩码

e）对原来图片的主体区域背景修复　　f）移动缩放后的主体到力量点

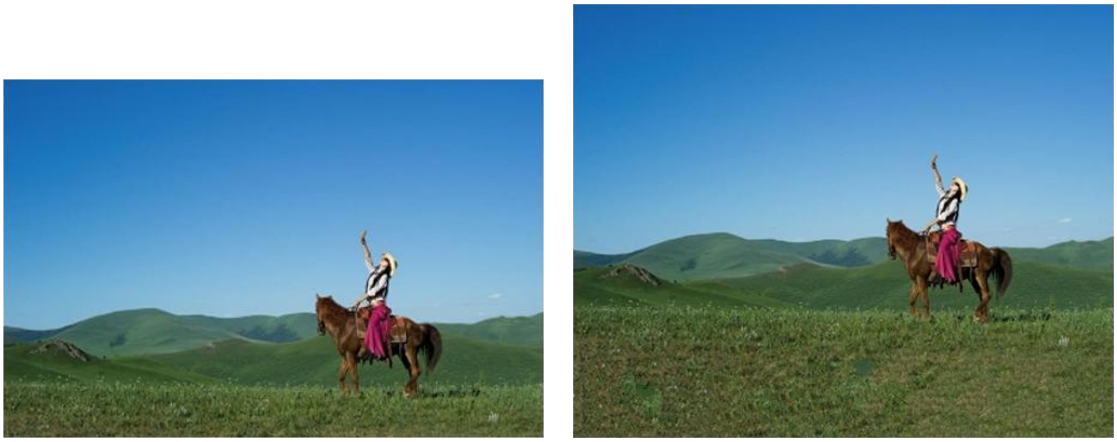
图 2-14 三分法则优化过程

2.6.2 视觉平衡优化

视觉平衡法则也是常用的构图法则之一，尤其在垂直/水平构图类型的照片中。黄金分割是常用的能够达到视觉平衡的方法之一，此外还有三角形构图，中心构图法。在本节中，我们将研究如何用黄金分割对照片进行视觉平衡优化。

在 2.1 中我们知道，水平线把一张照片分成两个区域，这两个区域的面积之比接近黄金比例 1：0.618 时便会给人视觉平衡的美感。如图 2-15 便是利用我们开发的图像工具对照片进行视觉平衡优化的例子。

a）原图　　b）优化后



c）原图　　d）优化后

图 2-15 视觉平衡优化示例

人们在拍照的时候通常不能把握好两个区域的比例，这时候我们就需要通过拉伸或者缩小照片中的其中一个区域来优化。由公式（2-1），我们可以增加或者减少 Y_k 或者 Y_g 的长度使 Y_k/Y_g 在合适的比例。我们选择了增加的方式，因为减少的方式会丢失很多的内容。

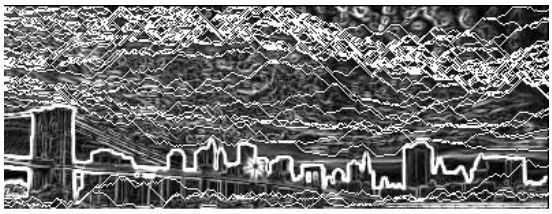
首先我们必须知道水平线所在的位置，在 2.4 中我们已经讨论了一种有效的水平线检测方法。得到水平线的位置之后，我们便能得到 Y_k 和 Y_g。设照片的高度为 H，根据公式（2-1），我们可以计算出 Y_k 或者 Y_g 应该被拉伸的长度 L。当 $Y_k > Y_g$ 且 $H - Y_k < 0.618 * Y_k$ 时，Y_g 应该被拉伸，拉伸的长度 $L = 1.618 * Y_k - H$。同理可以求得其他情况下的长度 L。

求得需要拉伸的长度之后我们就需要一个合适的拉伸算法，能够最大程度的保持图像的相似性。在 2.5 中我们讨论了一种图像空间重构算法插缝法，但是这种算法比较适用于对纹理简单的区域做大的改变。于是我们可以用它来拉伸天空、草原等纹理简单的区域，

这些区域通常处于上部区域，如图 2-15（a）。在 2.3 中我们还讨论了一种基于样例的图像修复算法，他比较适合于纹理较复杂的区域修复，但是对纹理简单的区域反而容易留下痕迹。于是我们用它来拉伸照片中纹理相对较复杂的区域，这些区域通常处于照片的下部区域,如图 2-15（c）。经过我们的实验验证，用这两种方法结合起来进行视觉平衡优化能够获得较好的效果。图 2-16 是用插缝法对照片进行视觉平衡优化的过程，用基于样例的图像修复算法进行优化的过程与之相似。

a）原图

b）检测水平线位置



c）传入区域掩码插缝

d）拉伸上部区域

图 2-16 视觉平衡优化过程

在图 2-16（c）中，我们传入了一个区域掩码，即把照片中水平线的下部区域全部置成白色，把他们的能量提升到最高，这样就可以避开这个区域，只拉伸上部的区域。

2.7 实验结果及分析

2.7.1 照片数据库的选择

我们的照片都来自互联网，但并不是所有的照片都适用于本文提出的构图优化算法。因为我们的算法是实现构图的调整，所以被用于实验的照片必须是构图类型明显的照片。

对于三分法则优化算法而言，单主体的照片比较适合，对于多主体的照片，只有紧凑的多主体照片才适合。对于中心构图的多主体照片，显然不属于三分法则构图了。对于视觉平衡法则优化算法而言，必须是垂直构图的照片才适合，因为我们的算法基于水平线的检测和调整，如果不存在水平线，自然就不符合了。图 2-17 展示了照片数据库中的部分照片。

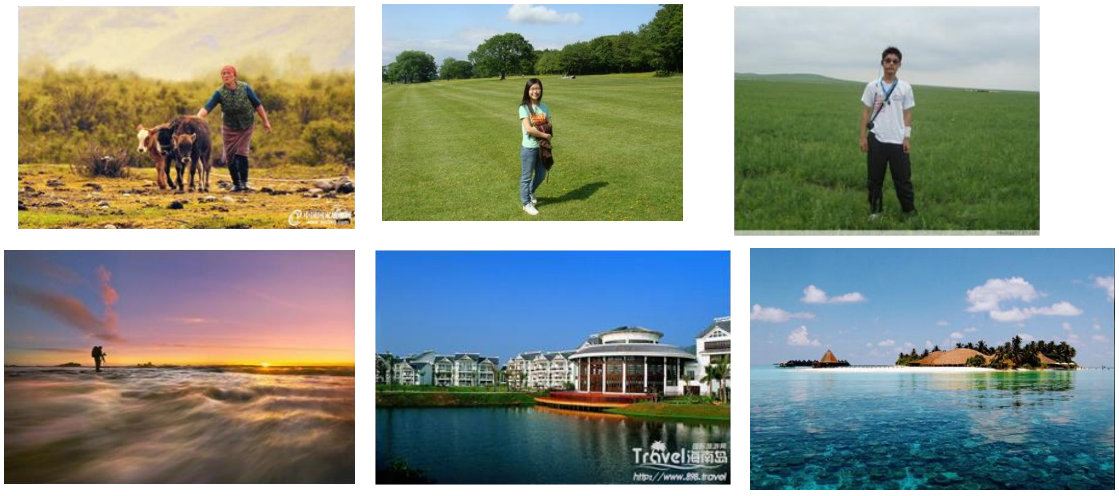


图 2-17 照片数据库部分照片

第一行中的照片均主体突出，适合于三分法则优化。第二行中的照片都是垂直构图，符合视觉平衡优化的要求。整个照片数据库由 93 张照片构成，其中垂直构图 51 张，主体突出照片 42 张。本实验对垂直构图照片进行视觉平衡优化实验，对主体突出照片进行三分法则照片优化实验。

2.7.2 主体提取实验结果及分析

三分法则优化关键是主体区域提取和背景修复，其中主体区域提取有交互方式和自动提取方式，交互方式由于是交互的，所以都可以提取出主体。因此本节对主体区域自动提取进行了实验，在实验中我们认为当提取出的区域达到主体区域的 90%以上并且没有其他无关区域相连则认为是成功的，否则都为失败。实验结果如表 2-1 所示。

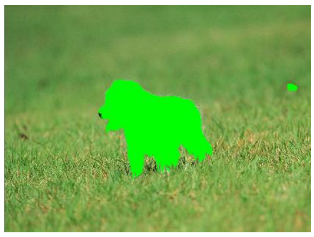
表 2-1 主体区域提取实验结果

成功数/张	失败数/张	成功率
34	8	81%

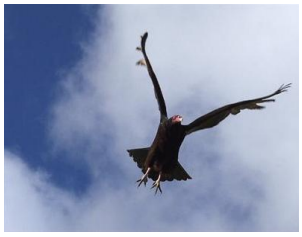
由表 2-1 中可以看出成功率达到 81%，图 2-18 给出了成功的一些例子，图 2-19 给出了失败的一些例子。

对结果深入分析可以发现，失败的情况有两种，一种是背景比较复杂的照片，成功率较低。一种是主体区域与周边区域色差不明显的照片，因为色差对 mean-shift 分割有很大的影响。

a）原图



b）提取主体



c）原图

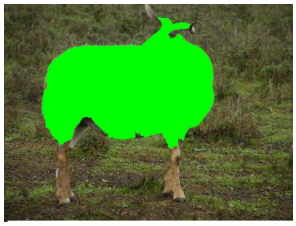
d）提取主体

图 2-18 主体提取成功的例子

a）原图

b）提取主体

c）原图



d）提取主体

图 2-19 主体提取失败的例子

2.7.3 水平线检测实验结果及分析

视觉平衡法则优化的关键是水平线位置的检测，只有准确检测出水平线的位置我们才能把它调整到黄金分割的比例。在水平线检测实验中，只有检测出的直线跟水平线完全重合或者几乎完全重合我们才认为是成功的。实验结果如表 2-2 所示。

表 2-2 水平线检测实验结果

成功数/张	失败数/张	成功率
46	5	91.2%

由表中可以看出成功率达到 91.2%，说明我们提出的水平线检测方法还是比较可靠的。

图 2-20 给出了一个成功的例子，图 2-21 给出了一个失败的例子。

对结果分析可以发现，失败的大多数原因是照片中存在多条水平直线，并且占据了有利位置，干扰了判断。

a）原图

b）检测直线

c）原图

d）检测直线

图 2-20 直线检测成功的例子

a）原图

b）检测直线

c）原图

d）检测直线

图 2-21 直线检测失败的例子

2.7.4 构图优化实验结果及分析

我们在对构图优化进行实验的时候发现，三分法则优化实验与自动提取主体区域实验的结果几乎是一致的，视觉平衡优化实验和水平线检测实验的结果几乎是一致的。即三分法则优化（自动提取主体区域）的成功率是 81%，视觉平衡优化实验的成功率是 91.2%。进一步验证了正确地提取出主体区域是三分法则优化的关键步骤，而准确地检测出水平线的位置是视觉平衡优化的关键步骤。

2.8 本章小结

本章阐述了主体区域检测和分割算法，图像修复算法，水平线检测算法和图像空间重

构算法，并最终利用这些算法实现了三分法则构图优化和视觉平衡构图优化。其中，我们介绍了两种主体提取算法，交互方式和智能方式。我们还探讨了一种基于样例的图像修复算法，并对它存在的问题进行了改进，提升了它的性能。接着我们研究了如何检测出水平线在照片中的位置以及利用图像空间重构算法改变照片的尺寸。最后，我们综合利用这些方法并根据构图理论实现了构图优化，它们使我们的照片获得了更好的美感。此外，我们还对构图优化算法进行了实验分析，验证了算法的可行性，也发现了存在的一些问题。

第三章　照片特效

3.1 人脸特效

在移动终端中，照片很多都是以人为主体，人脸相关的特效有很大的应用需求。在本节中，我们将研究如何在 Android 上实现人脸相关的特效。

3.1.1 瘦脸

很多女生都希望自己有一张精致 V 型脸，这里我们研究一种照片瘦脸算法。

很多时候，我们需要对一个图像的局部进行调整，这个调整必须是平滑的和可交互式的，我们希望这个调整看起来自然不突兀。

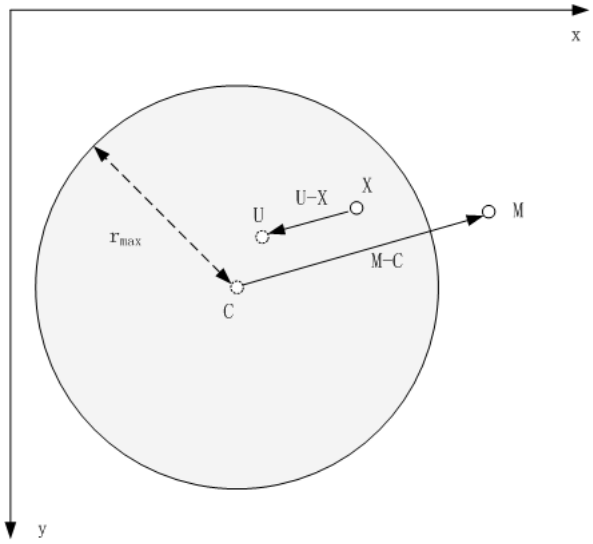


图 3-1 瘦脸示意图

图 3-1 中，阴影圆环代表一个半径为 r_{max} 的圆形选区。其中，C 点是鼠标点下时的点，也就是圆形选区的圆心。鼠标从 C 拖到 M，致使图像中的点 U 变换到点 X。所以，关键问题是找到上面这个变换的逆变换——给出点 X 时，可以求出它变换前的坐标 U，然后用变化前图像在 U 点附近的像素进行插值，求出 U 的像素值。如此对圆形选区内的每一个像素进行求值，便可得出变换后的图像。

Andreas Gustafsson 给出了这一逆变换公式[5]：

$$\vec{u} = \vec{x} - \left(\frac{r_{\max}^2 - \left|\vec{x} - \vec{c}\right|^2}{\left(r_{\max}^2 - \left|\vec{x} - \vec{c}\right|^2\right) + \left|\vec{m} - \vec{c}\right|^2} \right)^2 \left(\vec{m} - \vec{c}\right) \qquad (3\text{-}1)$$

为了控制变形的幅度，我们在上面的算法中引入了变形强度 S。于是逆变换公式变为：

$$\vec{u}=\vec{x}-\left(\frac{r_{\max}^2-\left|\vec{x}-\vec{c}\right|^2}{\left(r_{\max}^2-\left|\vec{x}-\vec{c}\right|^2\right)+S*\left|\vec{m}-\vec{c}\right|^2}\right)^2\left(\vec{m}-\vec{c}\right) \qquad (3\text{-}2)$$

这个变形算法的特点是：

1) 只有圆形选区内的图像才进行变形。
2) 越靠近圆心，变形越大，越靠近边缘的变形越小，边界处无变形。
3) 变形是平滑的。

具体实现步骤如下：

1. 对于圆形选区里的每一像素，取出其 R,G,B 各通道颜色值，存入 3 个 Buff（rBuff, gBuff, bBuff）中。
2. 对于圆形选区里的每一个像素 X，

a) 根据上面的公式，算出它变形前的位置坐标精确值 U。

b) 用插值方法，根据 U 的位置，和 rBuff, gBuff, bBuff 中的数值，计算 U 所在位置处的 R,G,B 等分量，在这里我们采用了双线性插值算法。

c) 将 R，G，B 等分量合成新的像素，作为 X 处的像素值。

图 3-2 是瘦脸的效果。

a) 原图　　b) 瘦脸后

图 3-2 瘦脸效果

3.1.2 鱼眼肖像

鱼眼肖像是利用鱼眼变形模拟出艺术画家画出的具有夸张效果的肖像漫画，通过选择人物头像需要夸张的部位和夸张的方式以获得不同艺术风格的图像。

首先选择影响整体五官，眼睛、鼻子和嘴巴的矩形区域，变形区域是矩形区域内的最大圆形区域，则鱼眼放大算法为：

1) 对最大圆形区域内的所有点(i, j)，把像素点坐标由直角坐标转换为极坐标。

2) 求得变形系数$S = R/\log(w*R+1)$。

3) 对 1 中的所有点，求得变换后的半径$r = (\exp(R/S)-1)/\mathrm{w}$。

4) 把 3 中的结果从极坐标转换到直角坐标，得到源像素点的浮点坐标$(srcx, srcy)$。

5) 用 4 中的结果双线性插值当前像素点(i, j)。

对鱼眼挤压算法，3 中的半径变换公式为$r = S*(1+w*R)$。

图 3-3 是由一张照片生成的一些肖像漫画。

a)原图

b)变形鼻子

c)凸起下巴

d)挤压下巴

图 3-3 肖像漫画示例

3.1.3 皮肤检测

皮肤肤检测是人脸识别的基础，也是很多人像识别技术的基础操作。在本节中，我们研究了几种基于像素统计学的皮肤检测技术，可以检测亚洲人种与白人的皮肤。

基于像素的皮肤检测主要是寻找正确的颜色空间几何，图像处理中，常见的颜色空间有如下几种：

1) RGB 色彩空间 – R 代表单色红，G 代表单色绿，B 代表单色蓝
2) HSV 色彩空间 – H 代表色彩， S 代表饱和度，V 代表强度值
3) YCbCr 色彩空间 – 是数字电视的色彩空间

一个简单的基于 RGB 颜色空间的皮肤算法如下：

如果点(R,G,B)同时满足$R>95,G>40,B>20$、$|R-G|>15$、$R>G\&\&R>B$且(R,G,B)三通道的最大值与最小值之差大于 15，则认为点(R,G,B)是皮肤。

一个简单的基于 HSV 颜色空间的皮肤算法如下：

如果点(H,S,V)满足$H>0\&\&H<50,H>0.23\&\&H<0.68$，则认为点$(H,S,V)$是皮肤。

一个简单的基于 YCbCr 颜色空间的皮肤算法如下：

如果点(Y,C_b,C_r)满足$Y>80\&\&85<C_b<135\&\&135<C_r<180,(Y,C_b,C_r)=[0,255]$则认为点$(Y,C_b,C_r)$是皮肤。

经过我们的测试，基于 RGB 颜色空间的皮肤检测算法能够获得更好的效果，并且满足实际的需求。在本文中，我们采用的都是这种皮肤检测算法。图 3-4 是皮肤检测的一个示例，（b）中黑色部分为非皮肤区域，其他区域为皮肤。

a）原图

b）皮肤检测后

图 3-4 皮肤检测示例

3.1.4 去斑去皱

由于斑点一般都比较小，所以去斑可以采用中值滤波来实现。中值滤波一般采用一个含有奇数个点的滑动窗口，将窗口中各点像素值的中值来替代窗口中心点的像素值。由于人像面部的皱纹起伏范围较大，可采用均值滤波的方法去皱。均值滤波是使用窗口模板法确定的邻域内像素的平均值代替窗口中心点的像素值。这种处理结果可降低图像中像素值”尖锐”变化的噪声，但也使图像整体变模糊。常用的3×3和5×5均值滤波模板如下：

$$\frac{1}{9}\begin{vmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{vmatrix} \quad , \quad \frac{1}{25}\begin{vmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{vmatrix}$$

我们用 3.1.3 中皮肤检测算法检测出皮肤，再用中值滤波对皮肤去斑，实现了智能去斑。同样，我们检测出皮肤后用均值滤波可以实现智能去皱。

图 3-5 是去皱的一个示例。图 3-5（b）中，我们利用3×3均值滤波模板对照片中的皮肤进行了去皱。它使皮肤更加平滑，有光泽。

a）原图　　b）去皱后

图 3-5 去皱效果

3.1.5 美白

美白就是增加皮肤区域中的亮度，使其看起来变白了，有光泽。对于 RGB 颜色空间表示的照片，增加亮度可以通过增加每个颜色通道的颜色值来实现。同样，我们用 3.1.3 中提出的皮肤检测算法检测出皮肤区域，然后增加皮肤区域的亮度，就实现了智能美白。

如图 3-6 是美白的一个效果图。

a）原图　　b）美白后

图 3-6 美白效果

3.2 图像滤镜

滤镜特效是图像处理中非常实用的功能，也是人们最常使用的功能。滤镜特效已经有很多很成熟的算法，在本节中，将主要研究几种比较特别的滤镜。

3.2.1 运动模糊

卷积模糊或者卷积平滑滤波，可以消除图像噪声，也可以产生一些常见的图像模糊特效，但是移动模糊特效也是基于卷积，相比于 Box Blur, Gaussian Blur 的算法，运动模糊只需要完成一次的一维卷积，所不同的是一维卷积的完成，要基于一定的角度，而不是只是在水平和垂直两个方向上。移动模糊的一维卷积要考虑一下三个因素：

1、操作数的多少，即距离(Distance)

2、一维操作数在像素数组中的移动方向，即角度(Angle)

3、一维操作数的拉影效应，即 Scale(放大和缩小程度)(Zoom/Scale)

距离和角度的关系可以用三角几何表示如图 3-7 所示：

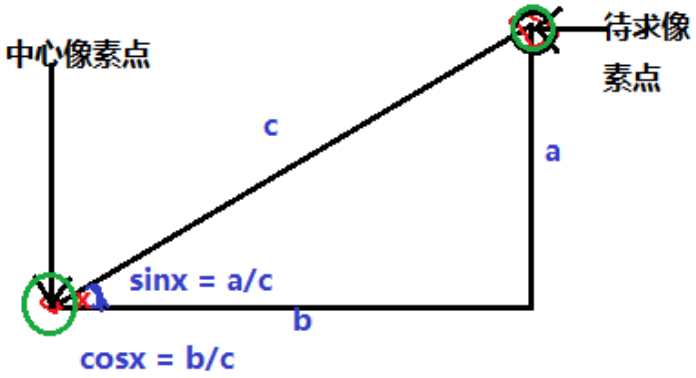


图 3-7 距离角度关系

在距离和角度已知的情况下，知道中心点(目标像素点)则可以求出每个操作数像素点坐标，假设中心点坐标为(x_0, y_0)，则操作数像素点(a,b)坐标公式可以表示如下：

$$\begin{cases} a = \sin x * c + y_0 \\ b = \cos x * c + x_0 \end{cases} \quad (3\text{-}3)$$

放缩功能其实是在 XY 两个方向对图像计算得到一个一维像素结合，再求这些像素的平均值即可，假设中心点像素坐标为(x_0, y_0)防缩比率为S_0则每个操作数像素点坐标可以表示为：

$$\begin{cases} a = x_0 - x_0 * S_0 + a \\ b = y_0 - y_0 * S_0 + b \end{cases} \quad (3\text{-}4)$$

对操作数像素点一维卷积后再填充目标像素点即可。

图 3-8 是运动模糊的一个示例

a）原图　　b）距离 0，角度 0，放缩-0.4 时的运动模糊效果

图 3-8 运动模糊效果

3.2.2 扭曲

使照片产生扭曲形变的方法有很多，常见的扭曲特效有水纹扭曲，哈哈镜，螺旋扭曲等，本节，我们研究了如何用三角几何实现扭曲特效。

1、水纹扭曲

主要是利用三角正弦函数与余弦函数的变换效果，完成对像素的位移变换，产生水纹效果，因为自然界中的水纹多少都是正弦波或者余弦波的叠加效果。设置两个参数调节，一个是波长 W，表示像素位移的多少，一个是周期 P，表示正弦或者余弦函数的变换周期。设目标坐标点为(x, y)，源坐标为$(srcx, srcy)$逆变换公式为

$$\begin{cases} srcx = x + W * \sin((2Pi * y)/P) \\ srcy = y + W * \cos((2Pi * x)/P) \end{cases} \quad (3\text{-}5)$$

用得到的源坐标$(srcx, srcy)$对目标坐标(x, y)双线性插值即可获得水纹扭曲后的照片。

2、哈哈镜和螺旋扭曲

在图 3-6 中的三角几何计算结果中，有两个可以改变其值再重新计算坐标。一个是角度，另外一个是半径距离，分别对角度与距离加以一定权重值计算，得到如下两种特效：

a)哈哈镜效果，主要是改变半径值，变换公式如下：

$$\begin{cases} newR = \sqrt{R} * factor \\ newX = centerX + (newR * \cos(theta)) \\ newY = centerY + (newR * \sin(theta)) \end{cases} \quad (3\text{-}6)$$

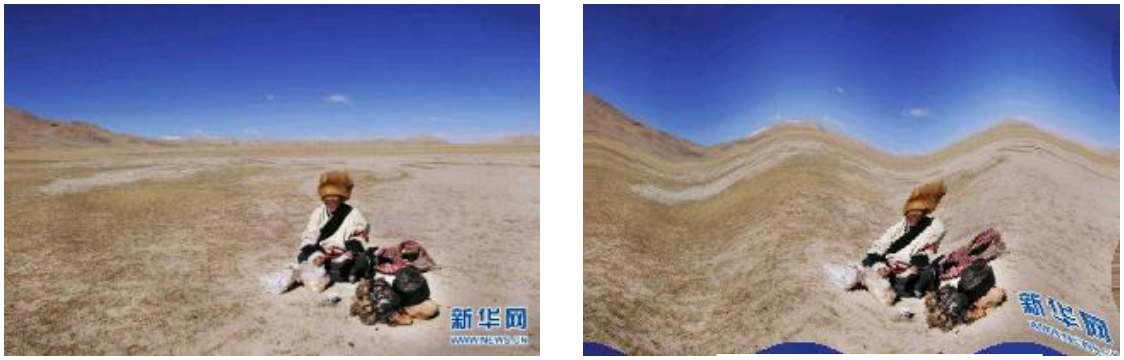
其中 *factor* 为输入参数，$(centerX, centerY)$是照片的中心坐标，*R* 是目标坐标与中心坐标的距离， *theta* 是偏转角度。

b)中心螺旋效果，主要是改变角度 *theta* 的值，变换公式如下：

$$\begin{cases} newX = centerX + (R * \cos(theta + degree * R)) \\ newY = centerY + (R * \sin(theta + degree * R)) \end{cases} \quad (3\text{-}7)$$

其中 degree 为输入参数。

如图 3-9 是对照片进行水纹扭曲的示例。



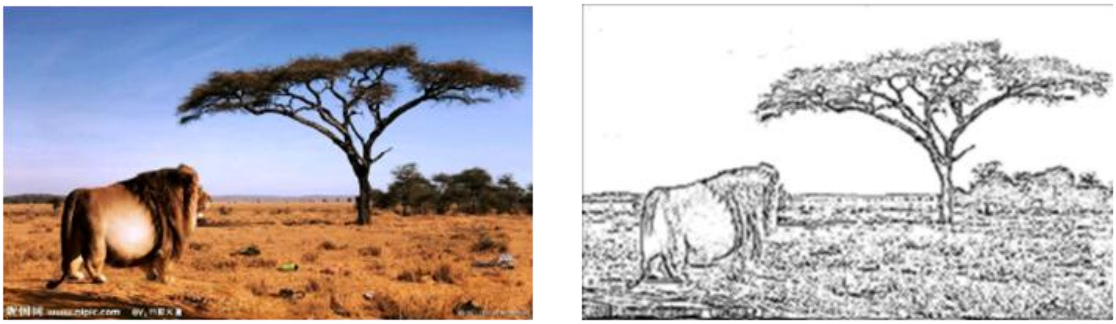
a）原图　　b）水纹扭曲后

图 3-9 水纹扭曲效果

3.2.3 素描

素描是美术技法中非常基础的一种方法，它是绘画者在一定面积或在物质的平面上，描绘出物体外在的形体及表现物体在空间中位置的一种技法。素描的特点是一种用线与面的表现方式表达物体的形象，用明暗层次表现一个物体在光照下的效果，因此素描可以帮助绘画者掌握物体的明暗层次和基本形象。实现素描的关键是准确地描绘图像的轮廓和反映物体的光照情况。我们可以根据拉普拉斯算子进行边缘检测生成边缘检测图，但其不足在于结果包含了过多的细节信息而易受各类杂点的影响。为了得到较理想的素描图像，我们先对图像进行降噪，然后在对素描图进行模糊处理，最终得到柔和的素描线条。图 3-10

是素描特效的一个示例。



a）原图　　b）素描效果

图 3-10 素描效果

3.2.4 颜色转换

颜色转换是指将一张照片中的某种颜色转换成另一种颜色，可以将红色的花转换成紫色，把白色的衣服转换成红色等。

我们可以用多种不同的颜色空间来表示一张照片，常用的颜色空间有 RGB 模型，HSL 模型和 HSV 模型。大多数电子设备和计算机都采用 RGB 模型来表示图像，是应用最广泛的颜色模型。但是最适合人的视觉感知的颜色模型是 HSV 模型，HSV 模型用色调、饱和度、亮度来表示颜色。设 $M=\max(R,G,B)$，$m=\min(R,G,B)$，$C=M-m$，则 RGB 模型到 HSV 模型的转换公式为：

$$H=\begin{cases} 60*\left(\dfrac{R-G}{C}+4\right) & M=B \\ 60*\left(\dfrac{B-R}{C}+2\right) & M=G \\ 60*\left(\dfrac{G-B}{C}\bmod 6\right) & M=R \\ 0 & C=0 \end{cases} \qquad (3\text{-}9)$$

$$S=\begin{cases} C/V & C\neq 0 \\ 0 & C=0 \end{cases}$$

$$V=M$$

设 $h_i=\left\lfloor\dfrac{h}{60}\right\rfloor$，$f=\dfrac{h}{60}-h_i$，$p=V\times(1-S)$，$q=V\times(1-f\times S)$，$t=V\times(1-(1-f)\times S)$，则从 HSV 到 RGB 的转换公式为：

$$(R,G,B)=\begin{cases}(V,t,p) & h_i=0\\ (q,V,p) & h_i=1\\ (p,V,t) & h_i=2\\ (p,q,V) & h_i=3\\ (t,p,V) & h_i=4\\ (V,p,q) & h_i=5\end{cases} \tag{3-10}$$

HSV 颜色模型非常符合人的视觉感受，容易判断颜色类型。常用的颜色区域如表 3-1 所示。

表 3-1 常用颜色 HSV 颜色区间

红色 H=0	(H>-1)&&(H<21)‖(H>322)&&(H<361);S>0.2;V>0.3
黄色 H=60	(H>21)&&(H<78);S>0.25;V>0.3
紫色 H=300	(H>269)&&(H<320);S>0.3;V>0.3
蓝色 H=240	(H>210)&&(H<265);S>0.25;V>0.26
绿色 H=120	(H>78)&&(H<163);S>0.2;V>0.15

我们可以根据这个颜色表判断是什么颜色，同样可以根据这个颜色表转换成另一种颜色。图 3-11 是颜色转换的一个示例。

a）原图　　b）红色转换成紫色

图 3-11 红紫颜色转换

3.2.5 怀旧

怀旧是人们非常喜欢的一种滤镜效果，它能使照片褪色，老化，使照片具有老照片的效果。人们习惯了鲜艳色彩的照片，很希望看到一些充满艺术性的，经过岁月沉淀的令人怀念的照片。

怀旧效果可以通过像素点的运算来实现，算法如下：

$$\begin{cases}R=0.393r+0.769g+0.189b\\ G=0.349r+0.686g+0.168b\\ B=0.272r+0.534g+0.131b\end{cases} \tag{3-11}$$

写成矩阵形式是

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix}$$

Android 中可以通过颜色矩阵（ColorMatrix 类）方便的操作颜色，表 3-2 是通过颜色矩阵实现怀旧效果的方法。

表 3-2 通过颜色矩阵操作颜色

```
Canvas canvas=new Canvas(bmp);
Paint    paint =new Paint();
ColorMatrix cm=new ColorMatrix();
float[] lomoMatrix=new float[]
{0.393f,0.769f,0.189f,0,0,0.349f,0.686f,0.168f,0,0,0.272f,0.534f,0.131f,0,0,0,0,0,1.0f,0};
cm.set(lomoMatrix);
paint.setColorFilter(new ColorMatrixColorFilter(cm));
canvas.drawBitmap(bitmap, 0, 0, paint);
```

图 3-12 是怀旧效果的一个示例。

a）原图　　b）怀旧处理

图 3-12 怀旧效果

3.2.6 LOMO

LOMO 也是人们非常喜爱的一种图像滤镜，它能使照片清新，唯美。LOMO 滤镜也可通过像素点的运算来实现，算法如下：

$$\begin{cases} R = 1.7r + 0.1g + 0.1b \\ G = 0 + 1.7g + 0.1b \\ B = 0 + 0.1g + 1.6b \end{cases} \quad (3\text{-}12)$$

同样可以通过 Android 的颜色矩阵类快速的实现。图 3-13 是 LOMO 效果的一个示例。

a）原图

b）LOMO 处理

图 3-13 LOMO 效果

3.2.7 调色

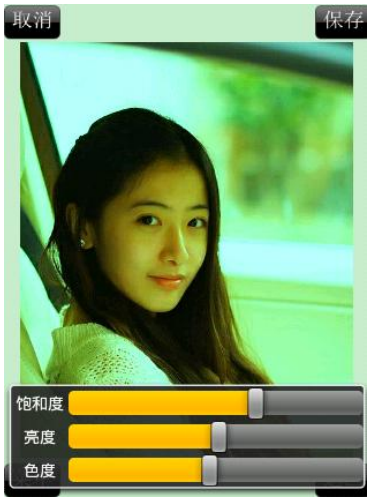
调色是图像处理要实现的一个基础功能，将特定的色调加以改变，形成不同感觉的另一色调图片。Android 的颜色矩阵 ColorMatrix 类对颜色操作提高了很好的支持，调色同样可以用它来快速对实现。表 3-3 是 Android 对调色操作的 API 支持。

表 3-3 Android 对调色操作的 API 支持

```
mHueMatrix.setScale(mLumValue,mLumValue,mLumValue,1);//调节亮度
mLightnessMatrix.setRotate(0,mHueValue);//调节色调
mSaturationMatrix.setSaturation(mSaturationValue);//调节饱和度
```

如图 3-14 是调色的一个示例。

a）原图



b）调色

图 3-14 调色处理

3.2.8 叠加

照片处理中，我们通常需要对多个图层叠加来得到叠加后的图像，图层叠加的方式有很多种，如 Multiply 叠加模式、Disslove 叠加模式、Overlay 叠加模式、Screen 叠加模式等。不同的叠加方式，效果有很大不同。

Multiply 模式：它产生的效果好像把两张幻灯片叠放在一起一样，所产生的混合色彩比原来各自的色彩都要暗淡。计算公式为：

$$v = v_1 * v_2 / 255 \qquad (3\text{-}13)$$

v_1 和 v_2 是叠加前各个图层的像素值，v 是叠加后的像素值。

Disslove 模式：他根据两个图层各自所占的比重进行叠加，比重越大的图片在叠加后的可见性就越强。计算公式为：

$$v = v_1 * t + v_2 * (1 - t) \qquad (3\text{-}14)$$

其中 $0 \le t \le 1$，是比重系数。

Screen 模式：它与 Multiply 模式相反，叠加时保留图层的强光，消弱图层的弱光。混合后的色彩比之前的两个图层都要亮。计算公式为：

$$v = 255 - [(255 - v_1)(255 - v_2)] / 255 \qquad (3\text{-}15)$$

如图 3-15 是用 Disslove 模式实现的相框效果。

图 3-15 Disslove 模式实现相框效果

3.3 照片编辑

照片编辑应该是最常见的图像操作了，如放大，缩小，旋转，反转等。Android 的空间矩阵类（Matrix）对照片的编辑提供了很好的支持，利用空间矩阵可以很方便的实现照片的各种空间变换。Android 同时提供了画布类（Canvas）可以很容易实现添加文字，涂鸦，

裁剪等编辑功能。

表 3-4 Android 实现照片编辑的方法

```
Matrix mMatrix=new Matrix();
mMatrix.postScale(scale,scale);//缩放，scale 为缩放系数
mMatrix.postRotate(degree);//旋转，degree 为旋转角度
Canvas c = new Canvas(bmp);
Rect cliprect=new Rect((int)window[0], (int)window[1], (int)window[2], (int)window[3]);
Rect dstrect=new Rect(0, 0, cliprect.width(), cliprect.height());
Paint paint=new Paint();
c.drawBitmap(bitmap, cliprect, dstrect, paint);//画布类实现裁剪
c.drawText ( text,  x,  y, paint);//添加文字
```

3.4 本章小节

在本章，我们实现了一些特别、实用的图像特效算法，如针对人脸的瘦脸和鱼眼肖像算法，可以调节照片的光线、色彩特征的怀旧、素描、模糊等滤镜效果以及可以使照片形变的扭曲效果。

第四章　照片处理软件的设计与实现

我们设计了一个基于 Android 平台的照片处理工具，实现了本文提出的所有算法。在本章，我们将详细介绍它的框架设计和具体实现。

4.1 Android 平台 NDK 开发

Android NDK 是一个原生态开发包，允许开发者用原生代码（C 或 C++）实现应用的一部分。这将给某些应用带来好处，这种方式可重用代码，而且在某些情况下可加快运行速度。对于图像处理，由于照片含有的数据量比较大，用 Java 实现效率会比较低，于是我们用 NDK 来实现我们的图像处理操作。Android NDK 开发其实就是使用 JNI 来调用本地的方法或者库来将 Java 程序和 Native 程序结合起来。因此 Android NDK 开发一般有以下步骤：

（1） JNI 接口设计。

（2） 使用 C/C++实现本地方法。

（3） 生成动态连接库。

（4） 将动态链接库复制到 Java 工程，运行 Java 程序。

1.JNI 接口设计

我们创建了 ImageUtilsJni.java 文件为 JNI 接口文件。在这个文件中定义我们的 JNI 接口函数，代码片段如表 4-1 所示：

表 4-1 ImageUtilsJni.java

```
public class ImageUtilsJni {
    static{
        System.loadLibrary("ImageUtilsJni");
        }
    public static native int[] toFilmJni(int[] pixels,int w,int h);
    public static native int[] sharpenImageJni(int[] pixels,int w,int h);
    ……
```

在 ImageUtilsJni.java 文件中，定义了所有图像操作的本地调用接口函数，这些接口函数并不需要在定义的类中实现，系统运行时会自动加载 ImageUtilsJni.so 动态链接库。

2.用 C/C++实现本地方法

我们用“javah ImageUtilsJni.java”命令可以生成一个本地代码需要的接口函数 C/C++头文件 com_huangbin_test_diyimage_ImageUtilsJni.h。

有了头文件之后，接下来我们就可以实现 C/C++本地方法了。我们新建了 ImageUtilsJni.cpp 文件，并在文件中实现了 com_huangbin_test_diyimage_ImageUtilsJni.h 中

定义的本地函数接口。

3.编译动态连接库

在本地代码目录 JNI 文件夹下新建 Android.mk 文件，它是用来向编译器描述本地代码的文件。Android.mk 的编写如表 3-2 所示。

表 4-2 Android.mk

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
include ../includeOpenCV.mk
ifeq ("$(wildcard $(OPENCV_MK_PATH))","")
#try to load OpenCV.mk from default install location
include $(TOOLCHAIN_PREBUILT_ROOT)/user/share/OpenCV/OpenCV.mk
else
include $(OPENCV_MK_PATH)
endif
LOCAL_MODULE       := ImageUtilsJni
LOCAL_SRC_FILES    := ImageUtilsJni.c
$(BUILD_SHARED_LIBRARY)
```

4.2 Android 平台 OpenCV 的移植和开发

OpenCV 是一个开源发行的跨平台计算机视觉库，实现了图像处理和计算机视觉方面的很多通用算法。我们在第二章构图优化中大量使用了 OpenCV 库，在本节，我们将介绍如何在 Android 平台上移植 OpenCV 和开发。

4.2.1 Android 平台 OpenCV 的移植

OpenCV2.3 开始已经有编译好的 Android 版本，只需要简单配置一下就可使用了。我们使用的是 OpenCV-2.3.1 版本。把 OpenCV-2.3.1 拷贝到 Eclipse 工作空间的平级目录。在 Android.mk 中加入 OpenCV 库的路径，如表 4-2 所示。再新建一个 Application.mk 文件，如表 4-3 所示，则可以使用 OpenCV 进行开发了。

表 4-3 Application.mk

```
APP_STL:=gnustl_static
APP_CPPFLAGS:=-frtti -fexceptions
APP_ABI:=armeabi armeabi-v7a
```

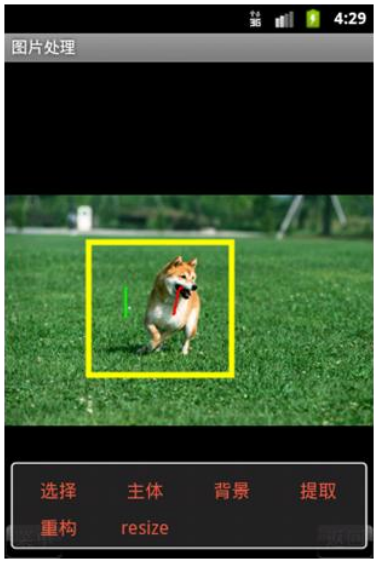
4.3 照片处理软件的框架设计和具体实现

我们设计了一个基于 Android 平台的照片处理工具，实现了本文前面讨论的所有算法。照片处理软件的系统框架可分为照片加载模块、功能导航模块、和照片处理三个模块。

照片加载模块的功能是获取需要处理的照片，可以从手机存储卡中获取，也可以通过相机即时拍摄一张。功能导航模块是功能菜单，我们实现了多个弹出式菜单。照片处理模块中我们采用了多线程技术以及使用了 NDK 和 OpenCV 开发。如图 4-1 是照片处理软件的具体实现。

a）照片加载模块

b）功能导航模块



c）交互式构图优化界面

d）图像修复和自动方式构图优化界面

图 4-1 照片处理软件的具体实现

图 4-1（b）中是软件的主功能导航模块，其中构图子菜单下包含构图相关的功能，如三分法则优化、视觉平衡优化、对象移除等。人脸子菜单下是一些针对人脸的特效如瘦脸，鱼眼肖像等。图 4-2（c）是交互式构图优化界面，我们通过点击相应的菜单就可以在屏幕上设置矩形框以及标记前景背景。图 4-1（d）中是图像修复和自动方式构图优化界面，在这里我们可以进行去水印，对象移除等图像修复相关的操作，也可以点击重构进行自动构

图优化。

4.4 本章小结

本章详细介绍了 Android 平台 NDK 和 OpenCV 的开发，简要介绍了基于 Android 平台的照片处理工具的框架设计和具体实现。我们设计的照片处理工具实现了本文提出的所有算法，并且大多数功能都是采用 NDK 和 OpenCV 实现，我们的工具在真实的设备上测试也获得了很好的效果。

结　论

摄影是人们发现美，记录美的途径。而随着互联网的普及，尤其是移动互联网和移动社交的发展，可以访问的照片越来越多，人们可以在自己的社交圈子里上传和分享自己的照片。但是，大多数人缺少摄影的专业知识，这使得拍摄出来的照片缺少美感，不能另人们满意。因此，设计出一个基于移动终端平台的照片处理工具变得非常有意义，使他能够方便的修复不符合人们视觉感知的缺少美感的照片。

构图是照片中图形元素的组合，它对照片的视觉感知非常重要，在很大程度上决定了照片的美感。它也是拍摄者很难掌握的技巧，人们拍摄出来的照片通常是构图不好的，这就限制了照片修复的空间。因此，本文提出了一种构图优化方法，实现了三分法则优化和视觉平衡优化。而在构图优化的基础上，本文实现了一些图像特效算法，从多方面提升了照片的美感。

对于单主体构图的照片，构图优化的关键是主题区域的提取和有效的背景修复算法。本文采用了显著区域检测和 mean-shift 分割算法结合的方法来提取主体区域，并且自动方式提取算法的测试成功率为 81%。对于背景修复问题，我们改进了一种基于样例的图像修复算法，可以有效的对主体区域进行背景填充。最后我们根据最熟悉的三分法构图法则使主体移动到力量点从而优化了照片的构图。对于垂直构图的照片，构图优化的关键是水平线的检测以及有效的照片拉伸算法。我们设计了一个水平线检测算法，可以有效地检测出水平线在照片中的位置，并且在我们的测试中获得了 91.2%的成功率。对于照片拉伸，我们采用了图像空间重构和图像修复相结合的方法，也获得了令人满意的效果。

对于移动终端而言，人们通常追求简单、方便。因此我们实现了一些简单但又特别、实用的图像特效算法。如针对人脸的瘦脸、鱼眼肖像算法，调节照片色彩的怀旧、LOMO 等滤镜，从照片的色彩，亮度等视觉特征方面提升照片的美感。

由于时间和本人水平所限，本文提出的多种照片处理算法仍然有很大的改进空间。其中本文提出的构图优化算法目前主要适用于单主体和垂直构图的照片，将来可以研究多主体、中心构图等较为复杂构图类型的照片。本文所采用的主体提取算法也仍然存在一些缺陷，对背景较为复杂的照片成功率仍然不够令人满意。设计出一个更加有效的显著区域检测算法是将来可以研究的方向之一。

参考文献

[1] ROTHER C, KOLMOGOROV V, BLAKE A. Grabcut——interactive foreground extraction using iterated graph cuts ［J］. ACM Transactions on Graphics, 2004, 23(3): 309 314

[2] R. Achanta, F. Estrada, P. Wils, and S. S ¨usstrunk. Salientregion detection and segmentation. International Conference on Computer Vision Systems, 2008.

[3] A.Criminisi, P.Perez,andK. Toyama,:Region Filling and Object Removal by Exemplar-Based Image Inpainting, IEEE Trans.Image Processing, 13(9),pp. 1200-1212,September 2004.

[4] AVIDAN, S., ANDSHAMIR, A. Seam carving for content-aware image resizing.ACM Transactions on Graphics 26, 3(2007).

[5]Andreas Gustafsson. *Interactive Image Warping*. Master's thesis, Department of Computer Science, Helsinki University of Technology, 1993.

[6] L IU L., CHENR., WOLFL., COHEN-OR D.: Optimizing photo composition. Computer Graphics Forum (Euro-graphics) 29, 2 (2010), 469–478.

[7] BHATTACHARYA S., SUKTHANKAR R., SHAHM.: A framework for photo-quality assessment and enhancement based on visual aesthetics. In ACM Multimedia (2010), pp. 271–280.

[8] NISHIYAMA M., OKABE T., SATO Y., SATO I.:Sensation-based photo cropping. In ACM Multimedia(2009),pp. 669–672.

[9]. R UBINSTEIN M., SHAMIR A., AVIDAN S.: Multi-operator media retargeting. ACM Transactions on Graphics (Sig-graph) 28 , 3 (2009).

[10] Yiwen Luo and Xiaoou Tang,Photo and Video Quality Evaluation: Focusing on the Subject [J].D.Forsyth, P.Torr,and A.Zisserman (Eds.): ECCV 2008,Part III,LNCS 5304, pp.386–399, 2008. Springer-Verlag Berlin Heidelberg 2008.

[11]杨丰盛.Android 应用开发揭秘[M].北京：机械工业出版社，2011:3.

[12]张宏林.精通 Visual C++数字图像处理典型算法及实现.[M].北京：人民邮电出版社，2008:134

致谢

本文是在我的导师王伟凝老师的悉心指导下完成的，首先，我要向王伟凝老师表示衷心的感谢。她不仅治学严谨，要求严格，而且为人随和，经常给以学生鞭策和鼓励。她教会了我怎样去思考问题，解决问题，也教会了我对待学术要有严谨的态度，创新的精神。在她的指导下，我的能力得到了很好的锻炼和提高，让我树立了解决问题的信心。

我还要感谢蔡东师兄和蚁静缄师姐，他们给了我很大的帮助和支持。同时感谢我的同学们，是你们给了我美好的大学时光。

最后，感谢我的父母，是他们无尽的爱让我走到今天。