本科生毕业论文（设计）

题　　目　基于图像处理的尿常规检测系统

学　　院　计算机学院

专　　业　计算机科学与技术

学生姓名　吴志杰

学　　号　2012141502017　年级　2012

指导教师　陈瑜

教务处制表

二〇一六年五月三十日

基于图像处理的尿常规检测系统

计算机科学与技术

学生 吴志杰 指导老师 陈瑜

[摘要] 有研究表明，在世界范围内成年人群体中慢性肾脏病的患病率约为11%，发病人数众多。如何及时便捷的检测和发现该疾病成为慢性肾脏病治疗环节中一个很关键的问题。尿液检测是一种简单有效的慢性肾病诊断方法。然而，传统尿常规检测的医疗器械体积庞大、价格昂贵而且需要医疗机构的专业人员操作。其次，城市就医困难等问题也导致人们不能够便捷及时的进行尿液检测。为解决这些问题，本文设计和实现了一个基于Opencv图像分析技术和移动智能设备的尿常规检测系统。通过该检测系统，可以较好的实现简单快捷的尿液检测。

本尿常规检测系统主要由三大模块组成：（1）中高端智能手机一台；（2）实验反应盒；（3）图像处理模块。实际的实验测试结果表明本文设计的尿常规检测系统具有较高的精度，较快的速度，成本低，检测较为方便等特点，具有较好的应用前景。

在本文设计的尿常规检测分析系统中，移动端包含Windows Phone和iPad两个版本，实验反应盒为自制的简易化学反应盒，图像处理模块为使用Opencv开源库的后台图像处理程序。最后实验结果表明，我们设计并实现的尿常规检测系统能对尿液中各种生化指标进行准确快速地检测，同时具有性价比高和便捷易用等优势。

[关键词] 尿常规检测；数字图像处理；Opencv；移动客户端

Urine Detection System Based on Image Processing

Computer Science and Technology

Student：Wu Zhi-Jie Adviser: CHEN Yu

[Abstract] Studies have shown that the prevalence of chronic kidney disease has reach to 11% among the adult population in the world. A large number of people are suffered from this disease. How to detect this disease through a quick and convenient way has become a very critical part of the treatment of chronic kidney disease. Urine detection is a simple and effective way for the diagnosis of chronic kidney disease. However,there are some facts that make this detection is inconvenient. Such as, urine detection medical devices are bulky, expensive, and require medical professionals to operate.In addition,people can not get timely and convenient urine detection services due to some problems such as the urban medical care.To solve these problems, in this study, we designed and implemented a urine detection system based on Opencv image analysis technology and mobile smart devices. With our detection system, people can detect them urine themselves.

Ours urine detection system consists of three modules: (1) one smart phones; (2) the experimental reaction box; (3) the image processing algorithm module. Experimental results illustrate that our urine detection system has high precision and speed, with low cost. We believe it can have a good prospect in the future.

Our urine detection system has two versions in the mobile terminal,Windows Phone Version and the iPad Version.The experimental reaction box is a simple chemical reaction cartridge made by myself.The image processing module is a background image processing program using open source library Opencv.

Finally, the experimental results show that our urine detection system can detect many kinds of urine biochemical indicators quickly and accurately. Also, it has the high cost-effective and easy-to-use advantages.

[Key Words] urine detection;digital image processing;Opencv;mobile terminal

目 录

1.综述

1.1 引言

慢性肾病（CKD）是中老年群体中较为常见的一种疾病。医学上，通过尿液分析仪对尿液进行检测来诊断慢性肾病是一种有效的手段。然而，尿液分析仪是一种较为昂贵的专业检测仪器，需要医院专业技术人员操作。此外，人口老龄化的日趋明显和就医困难等问题的困扰，使得传统的去医院进行尿液检测的方式日益暴露出较多的困难。如何便捷、快速地进行尿液检测就成为一个值得研究的问题。

随着临床生化检验技术的快速发展，干式生化分析技术已经逐步取代传统湿化学检验技术，其所具有的精度高、重复性好、检测速度快等优点，正好符合现在临床诊断对生化参数检测的要求。因此，干式生化分析技术已经成为生化分析领域一个新的热点研究方向[1]。

现行的尿常规检测设备是以分光光度法为核心技术的分光检测系统，其先对反应后待检测的试纸或试片上的颜色信息进行采集，然后经过后续的电路分析给出最终结果。但是这种采用分光光度法为核心的尿常规检测设备存在如下几个问题：

（1）单台设备售价较为昂贵。现在普通医院使用的尿常规检测设备单价至少为2000元，对普通百姓而言较为昂贵，难以普及到每户家庭；

（2）体积较大，不利于放置。现行的检测设备体积大概有八台pos机体积大小，不便于携带；

（3）不能进行历史检测信息的存储和对历史监测信息进行分析；

（4）无法精确处理反应不充分的试纸。如果试纸反应不充分，出现了未反应部分和反应不充分部分，基于分光光度法的尿液生化分析仪无法有效、快速地进行区分。在这种情况下，基于分光光度法的检测方法还是把整个反应区，无论是反应充分的还是没有反应或反应不充分的区域进行像素值平均，因此会降低检测精度。

针对以上在传统尿液检测工作中存在的诸多不便之处，本文设计和实现了一个便携式尿液检测系统。该系统有效地克服了传统光电式的尿液分析技术缺陷，将彩色图像处理技术引入到生化分析中。利用图像处理算法对试纸图像数据进行处理，能有效地减少误差、降低尿液检测成本，并具有较好的便携性和性价比。

1.2 尿液检测系统的现状

由相关调查研究表明，在我国40岁以上的人群当中慢性肾脏病的患病率为8%到9%。而同其它重症疾病相比，慢性肾脏病的表现更为隐秘。在发病初期并没有明显症状，而当患者出在现明显症状才去就医时，常常已经被诊断为尿毒症。根据相关统计，尿毒症患者平均每年医疗花费超过十万元。

虽然慢性肾脏病对人们健康危害极大，但是只要能做到“及早诊断，积极预防”就可能使危害减到最小。定期体检、进行简单的尿液、血液或者超声波方面的检查，就可以完全实现早期发现和早期诊断。而在慢性肾病的发病初期，患者的平均医疗花费可以降为每年几千元。因此，以尿常规为代表的尿液检测对慢性肾病等疾病的诊断和发现具有极其重大的意义，人们对方便、快捷和经济的尿常规检测方法拥有巨大的需求。

在慢性肾病诊断过程中，尿常规检查是一项必需的检查内容。现行的尿常规检测仪的主要原理是将分光光度法和干化学方法结合，让涂在尿常规检测试纸表面的化学物质同用户的尿液中相应待测物质发生化学反应，然后测定反应后试纸各个反应区的反射率。通过反射率和待测物质浓度之间的映射关系测算出用户尿液中相应的生化指标浓度，进而给医生提供诊断依据。而从第一台干化学分析仪的诞生到现在，经过数十年的发展，结合分光光度法的干化学技术在医疗检测领域取得了更加广泛的应用。检测试纸的体积不断缩小，检测项目不断增多同时检测精度也不断提升。但是相对应的各种干化学检测仪器一般都体积较大、单价较为昂贵、数据不能保存和需要专业人员操作。

在本文中设计和实现的基于图像分析技术的尿常规检测系统就是针对上述现行的尿常规仪器的缺陷而设计的。一方面可以满足方便、快捷地进行尿常规检测的需求，另一方面又比较经济，还能对检测数据进行存储和分析，以便跟踪病情。

1.3 图像处理技术概述

数字图像处理是一门研究对图像数据进行采集、存储和分析的学科。数字图像第一次于上世纪二十年代被用于图像的远距离传输，同时到了上世纪六七十年代的时候，由于相应数学基础的创立和完善，特别是离散数学理论方面，数字图像技术取得了飞速的发展，对社会的影响愈加深远，在广度和深度上均有很大提升。

近年来，由于社会和技术环境的变化，数字图像处理技术领域不断涌现出新的技术，如实时图像处理、图像搜索和建模等技术。视觉作为人类感知世界最重要的一种方式，随着技术的不断进步和人们对物质和精神领域的不断追求，数字图相处技术一定也会得到更大的发展空间和更加广泛和深远的应用。

由于彩色图像可以描述的信息量要远远大于灰度图像，所以彩色图像相较于灰度图像能更加全面和真实地反映世界和给人以直观的感受。因此，彩色图像处理技术已经逐渐成为数字图像处理领域的新的研究热点。

彩色图像处理过程通常包括图像预处理、图像分割、图像数据压缩、图像重建和结果输出。图像预处理的主要步骤为去噪和提高清晰度，主要用到的方法为中值滤波、低通滤波、带通滤波等。图像分割过程在整个图像处理中的地位十分重要，具有代表性的方法有阈值分割方法、边缘检测分割算法等。

随着各种算法、光学成像部件技术和计算机处理能力的创新和进步，数字图像处理技术，特别是彩色图像处理技术定将拥有更加广阔的发展空间和更广泛的应用。

1.4 本文主要研究内容

在当今社会，人们的时间成本越来越高，所以越来越希望能方便快捷的检测自己的健康状况，并且在保证一定结果准确度的前提下尽量减少成本。这就促使医疗检测设备向着高精度、高速度、小型化、低成本的方向发展。

针对现有的光电式尿常规分析仪存在的设备单价较高、体积较大、不便携带等固有缺陷，本文将彩色图像处理技术引入到尿常规检测当中，以彩色数字图像处理系统作为系统的核心替代传统的光电检测系统，设计并实现了一个便携式的尿液检测系统。该检测系统可有效减少由于模块反应不均产生的颜色误差，提升准确度；同时，能够降低单个设备的检测成本，减小设备体积，便于存放。此外，还可以对历次检测数据进行存储和分析。

论文的主要内容包括以下几个方面：

（1）干式生化分析仪现状分析；

（2）彩色图像分析技术简介；

（3）尿常规检测系统客户端部分——iOS API 技术；

（4）尿常规检测系统后台部分——PHP 后台技术；

（5）尿常规检测系统图像处理部分——Opencv API 技术；

（6）尿常规检测系统的设计与实现；

（7）尿常规检测系统存在的问题。

1.5 基于彩色图像分析的尿常规检测系统的处理步骤

本尿常规检测系统需要经过以下 5 个核心步骤来完成图像分析工作：

（1）使用光学仪器采集彩色图像数据；

（2）对采集到的彩色图像数据进行平滑去噪处理；

（3）采用边缘检测算法将反应模块从固定背景中提取出来；

（4）利用图像分割算法对模块进行进一步的分割提取出完全充分反应的部分出来；

（5）提取这一部分的颜色特征，并与预先存入的标准比色卡中的色样进行比较，最终给出结果。

其处理流程图如下图 1-1 所示。

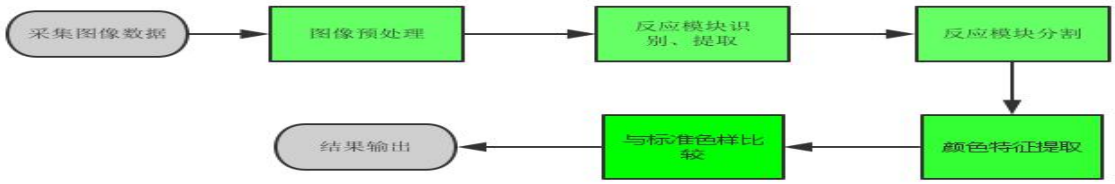

图 1-1 图像处理模块处理流程

1.6 论文组织结构

第一章：绪论。主要介绍论文的选题背景，尿常规检测仪现状，主要研究内容、检测系统的处理流程和论文的组织结构。

第二章：尿常规检测系统的概述。主要介绍尿常规分析仪的原理和结构以及存在的问题。

第三章：尿常规检测系统核心技术讨论。简述彩色图像处理算法、Opencv API、iOS API、Windows Phone API 和 PHP 脚本等技术。

第四章：基于 Opencv 的彩色图像处理尿常规检测系统的设计与实现。主要介绍该系统的架构，各个组成部分以及主要功能实现。

第五章：尿常规检测系统运行效果观察与分析总结。对检测系统运行结果进行比较说明，通过实验数据来表明基于彩色图像处理的尿常规检测系统的有效性。

第六章：总结。

2.尿常规检测系统开发概述

2.1 研究背景

慢性肾病，也被称为慢性肾脏病，是指在数月或数年的时间内，可导致肾功能逐渐丧失的一种症状。在临床上，如果测试者通过肾脏病理学或影像学检查、血液检测和尿液检测(当尿常规检测中潜血、蛋白质、白细胞和葡萄糖等指标检测为阳性时，可被诊断为肾损害)等方式，检查出的肾损害时间大于等于三个月，就可以被诊断为慢性肾脏病。当这种现象出现并持续三个月以上时，慢性肾病患者将发展成慢性肾功能不全、肾衰竭并最终形成尿毒症[2][3]。

当尿液中出现大量泡沫且长时间不消退以及尿中见血色或夜尿增多时，即尿常规检查尿中出现尿蛋白和红细胞，则可能为慢性肾病早期发生的预警信号。早期慢性肾病的症状不明显，如一般的慢性肾炎患者都没有特别明显的不适症状。故有尿液异常时，需要尽快就医检查诊断。一旦感觉自身有明显疲劳、贫血等非常明显的症状出现才去就医时，可能慢性肾病病情已经过了早期阶段了，这样将为慢性肾病的后期治疗带来更大的难度。

而据相关研究显示，目前中国成人慢性肾病的患病率为 10.8%，总数高达 1.2 亿人，即平均每 10 人就有 1 名慢性肾脏病患者，但知晓率只有 12.5%。同时慢性肾病的增长率却将超过 17%[4][5]。因此，慢性肾病的监控和防治无论对普通百姓还是政府、医院等大型部门都显得格外重要。新型的尿常规检测设备将在这个过程中将会发挥重要的作用。

2.2 尿常规检测方法的现状

肾病类尿常规是一项用于检查是否患有肾病的检查方法。尿常规检查简便易行，是临床上最常用的检查方法之一。并且它可直接、迅速地反映泌尿系统的情况，因此，经常做尿常规检查，对及时发现泌尿系统疾病具有重要意义[4]。

尿液分析仪是进行尿常规检测的有效仪器。目前的尿液分析仪则主要利用检测光对试纸条上的反应色块进行逐项扫描，检测系统将光信号转换成电信号，经过处理器计算出对应色块的反射率。因为检测光对具有不同的色块颜色和不同的深浅程度的颜色的物体进行照射，其反射率不同，因此，通过测定反射率的梯度进而可确定尿液中各种生化成分的含量，从而实现尿液成分的识别。

2.3 尿常规检测设备特点与不足之处

尿常规检测设备对于慢性肾病的临床诊断具有极其重要的指导作用，其具有以下四方面的特点：

（1）实时性：尿常规检测设备要求系统检测速度较快，通常进行一次检查，从开始到得到最终结果所花时间不超过十分钟；

（2）检测指标较多：一般尿常规检测系统可以检测酮体、隐血、蛋白质、胆红素、白细胞、PH 值、尿酮、比重、葡萄糖和亚硝酸盐等十多种生化指标；

（3）经济成本：用户单次检测成本较低；

（4）检测结果意义重大：尿常规作为最常见的体液检测手段，对检测众多肾脏疾病拥有极高的指导意义。

然而，常规的尿常规检测设备也存在以下五方面典型问题：

（1）检测设备单价对个人而言较为昂贵。现在一台尿液分析仪器在市场常见售价为 2000 元左右，对用户来说成本较高。

（2）现行检测设备体积较大。一般医院现行使用的尿常规检测设备体积接近于 8 台 pos 机的体积大小，不利于个人的存放。

（3）用户无法从检测报告中得到直观信息。用户对各个检测数值并无经验，无法准确判断检测结果对其健康具有的意义。

（4）无法建立待测样本和用户之间的联系，进而无法对该用户的数据进行存储和管理，跟踪用户的病情发展。

（5）不能有效区分试纸是否反应充分和提取出反应充分部分的图像，只能简单地对该反应区图像进行颜色求均值。这在多种情况下会引入误差。

针对以上问题，本文设计并实现了一种新型的便携式尿液分析系统，其可以方便快捷地对尿液进行尿常规检测。

3.尿常规检测系统核心技术介绍

尿常规检测系统的目标是为了对用户所采集的尿液进行分析，检测出用户采集的尿液标本中相应生化指标的浓度，进而反映用户的健康状况。本文所设计和实现的尿常规检测系统包括如下几个部分：图像采集部分(如手机相机)、客户端、后台、彩色图像分析部分和反应实验盒等。

在客户端层面使用了 Windows Phone API、iOS API 和第三方开源组件来提升开发效率。如在 iOS 版本的客户端中使用了 AFNetworking 来进行网络请求(如将用户登录和注册信息上传到后台并接受后台传回的数据)，使用 Realm 用于进行移动端的数据持久化处理(如在客户端本地存储用户名、用户性别、检测历史记录等信息)，使用 Mantle 来更高效地进行 json 数据解析进而构造出相应的模型(如将图像分析返回的 json 数据自动转换成一个类模型对象)，使用 Masonry 来简化 iOS 中的 autoLayout 语法，进而更为方便、快捷和高效地实现页面布局。在后台主要使用了 LAMP 架构，以 PHP 为后台脚本，MySQL 为数据库存储数据，Linux 为后台服务器操作系统，Apache 为服务器软件。同时，当用户发现检测结果异常时，为方便用户就医，在应用中使用了百度地图 API，用于定位、搜索附近医院和路径导航。

尿常规检测系统最根本的功能是对用户提供的尿液样本进行准确和方便地分析、检测。因此，本系统的核心技术是图像处理模块相关技术和 Opencv 开源视觉库。由于篇幅限制，故在核心技术部分暂不对 Windows Phone 和 IOS 客户端进行阐述。下面分别对系统中所使用的图像处理技术、蚁群算法和 Opencv 库做一个简单的介绍。

3.1 彩色图像处理技术

3.1.1 彩色空间

选择彩色空间时，需要考虑诸多因素，如实际图像的特性、所采集图像的质量以及希望应用的领域。每种彩色空间都适用于不同的场景，且都有各自的局限性。因此，在对采集到的图像进行分析的第一步就是选择一个合适的彩色空间。表 3-1[1]是对常用彩色空间的优缺点和适用范围进行比较。

本系统图像分析模块的主要功能是提取出试纸反应区的图像信息以得到反应最为充分的部分，最后根据反应充分区域的颜色信息来计算用户尿液中的相应待测生化指标的数值。而在图像的分割和提取反应最为充分的反应区域时，试纸模块上颜色差别很小。

根据表 3-1 中对各彩色空间优缺点的对比，结合系统的实验环境和图像处理条件，为了能准确地测量颜色之间微小的差别，本文选取 CIE 的非线性变换 L*a*b*彩色空间作为图像处理单元的运算空间。

表 3-1 彩色空间对比

彩色空间	优势	劣势
RGB	易于显示，且较为直观	R、G、B 三个分量高度相关不太适合进行图像分割
YIQ	部分去除 RGB 的相关性，运算时间减少，Y 分量适合边缘检测	分量之间仍然存在一定的相关性但比 RGB 空间相关度低
YUV	部分去除 RGB 的相关性，运算时间减少	分量之间仍然存在一定的相关性但比 RGB 空间相关度低
HSI	H 分量对光线、阴影不敏感，可用以区分不同颜色的物体，适用于光照发生变化时	RGB 非线性变换，在低饱和度时颜色的数值不稳定，可能存在着奇异点
Nrgb	彩色分量独立于图像亮度，对亮度变化不敏感，适合于表示调色板	RGB 的非线性变换，在强度很低时会产生很多噪声
CIE(L*a*b*)	颜色和强度相互独立，颜色差异大小能由几何距离表示，尤其适合颜色差异较小时	存在奇异点

3.1.2 图像预处理技术

成像光学系统中的光电传感器等器件造成的干扰和传输过程中可能引起的错误，使得图像中形成了各种噪声。噪声会对后续的图像分析处理造成较大的影响。因此一个良好的图像处理系统的主要目标即为把噪声的不良影响降到最低，而去噪也成了图像处理中极为重要的步骤[9]。

在图像处理技术发展过程中，平滑去噪技术分为两类：一类是全局处理技术，另一类是局部算子技术。由于全局算子计算量较大，所以一般不采用该方法。采用局部算子平滑图像时，只是在一定的小区域对每一个像素进行平滑处理，因此具有较高的计算能力，并且能并行处理多个像素数据点。因此局部算子平滑去噪技术十分适合实时或者准实时图像的预处理[1]。

常见的图像预处理技术有线性模糊滤波器、简单无缩放变换的模糊滤波器、中值模糊滤波器、高斯模糊滤波器、双边模糊滤波器、线性滤波器、均值滤波器、低通滤波器

和带通滤波器等。考虑到图像采集模块采集到的图像质量和后续图像分割及颜色提取操作对图像质量的要求，本文中选用了中值滤波技术作为图像预处理部分的滤波技术。

中值滤波技术是一种在空域上进行的局部非线性平滑技术。其对脉冲噪声的滤波效果较好，并且能保证目标图像边缘的完整性。其计算方式是使用一个大小为奇数的滑动窗口，用模板中各点灰度值的中值代替窗口中心点的灰度值。中值滤波中的窗口大小对最终的预处理效果至关重要，若窗口太小，则在去噪时，对噪声十分敏感。一般通过实验来选取窗口大小。

3.1.3 图像分割技术

为从背景中提取出感兴趣部分，必须得对源图像进行分割。图像分割的作用是将图像分割成几个互不重叠的空间区域。同一区域内的像素具有相似的特性，不同区域的像素特性差异较大。

常用的图像分割方法主要有基于边缘检测的方法和基于区域的方法两类。从不同角度来说，这些方法又可以继续分成如下三类：阈值分割法、边缘分割法和区域跟踪法。

由于在本文设计的系统中，反应试纸的颜色特征和实验采集图像的背景环境颜色特征差异较大，故使用边缘检测法来作为目标图像的分割方法来提取出反应模块。在边缘检测算法中，某种特性在图像各区域间存在的不连续部分就是边界，然后边缘检测就根据检测到的边界对图像进行分割。

边缘检测的优点是计算速度较快、边缘定位准确，其缺点无法保证分割区域内部颜色特征的一致性，抗噪声能力差，不能产生连续、闭合的区域轮廓。因此，为了能更加有效地对采集到的图像数据进行分割，本文在设计尿常规检测系统时采用了固定试纸放置的位置和手机拍照的位置等方法，在边缘检测分割出完整的试纸图像后，按照相对应的坐标准确地分割出每种待检测指标的反应区图像。通过将这两种图像分割方法的结合，本文设计的尿常规检测系统已经能十分准确地分割出每种物质的反应区的图像。

3.1.4 FCM 算法

为了提取出反应最充分的区域，考虑到采集到的图像数据的质量以及对后续图像处理的需求，故使用 FCM 模糊 C 均值算法来进一步提取反应最充分的区域，使用该算法主要是为了解决在各个反应区块内颜色差异较小的问题。

FCM 算法是以模糊数学和均值聚类算法为理论基础的模糊聚类算法，其主要思想是不断优化目标函数以使得被划分到同一簇的对象之间相似度最大，而不同簇之间的相似度最小。FCM 是对普通的硬性 C 均值算法的改进，普通 C 均值算法对数据的划分是硬性的，而 FCM 则是一种柔性的模糊划分。FCM 是种用隶属度来确定每个数据点属于某个聚类的程度的一种聚类算法。FCM 的大概操作为，把 n 个向量 x_i（i=1,2,…,n）分为 c 个模糊组，并求每组的聚类中心，使得非相似性指标的价值函数达到最小。FCM 与 HCM 的主要区别在

于 FCM 用模糊划分，使得每个给定数据点用值在 0，1 间的隶属度来确定其属于各个组的程度。与引入模糊划分相适应，隶属矩阵 U 允许有取值在 0，1 间的元素。不过，加上归一化规定，一个数据集的隶属度的和总等于 1：

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \ldots, n \quad \cdots\cdots (3.1)$$

则 FCM 的价值函数为

$$J(U, c_1, \ldots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} \quad \cdots\cdots (3.2)$$

利用 d_{ik} 来计算 $u_{ik,}$ 有如下公式

$$u_{ik} = \left[\sum_{j=1}^{c} \left[\frac{d_{ik}(x_k, v_i)}{d_{jk}(x_k, v_j)} \right]^{2/(m-1)} \right]^{-1} \cdots\cdots (3.3)$$

进而更新聚类中心矩阵

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \quad \cdots\cdots (3.4)$$

但是，当 FCM 方法应用于图像分割时，像素数据在空间域的关系完全被忽略掉，若图像存在较大噪声，往往会使分割出的区域中出现许多离散空洞或杂乱点。一种解决方法是对原图像进行低通滤波预处理，而后进行 FCM 分割。然而，标准的图像平滑处理可能会导致图像细节的缺失，而且没有一种合适的方法在图像平滑和图像特征聚类之间进行有效地权衡[7]。

为克服 FCM 分割方法的上述缺陷，在标准 FCM 算法中引入了像素的空间信息并定义了一套新的价值函数，利用像素邻域的约束信息，提出一种改进的空间约束 FCM 分割方法[18]。

在标准的 FCM 聚类算法中，隶属度和相似性度量是影响图像最后的分割好坏的重要因素。因此，将空间信息同时引入到隶属度函数和相似性度量中，每次迭代运算都将对它们进行修正。

新的相似性度量由临近像素的强度信息决定，而新的隶属度函数由邻近像素隶属度和空间位置信息共同决定。

定义新的相似性度量：

$$D_{ij}^{*} = \lambda_{ij} \| x_j - v_i \|^2 \quad \cdots\cdots (3.5)$$

其中 λ_{ij} 表示像素 x_j 的空间邻域信息：

$$\lambda_{ij}=1-\frac{\sum_{i=1}^{Nr}u_{ir}d_{jr}}{\sum_{r=1}^{Nr}d_{jr}} \quad (3.6)$$

其中 Nr 为像素 xj 邻域内像素点的个数，d_{jr} 为 x_j 与邻域像素 x_r 间的距离。

$$d_{jr}=\left\|x_j-x_r\right\| \quad (3.7)$$

新的隶属度函数定义为

$$u_{ij}^{*}=\frac{u_{ij}h_{ij}^{2}}{\sum_{k=1}^{c}u_{kj}h_{kj}^{2}} \quad (3.8)$$

其中 h_{ji} 为

$$h_{ij}=\sum_{r=1}^{Nr}u_{ir} \quad (3.9)$$

改进后的 FCM 算法的具体操作步骤为：

（1）确定聚类类数 c，模糊权值 m，终止条件 ε 和邻域大小 n；

（2）初始化聚类中心 vi，并使用标准 FCM 聚类算法来计算隶属度 uij；

（3）计算新的相似性度量 D_{ij}^{*}；

（4）根据（3）所得的相似性度量 D_{ij}^{*} 计算新的隶属度 u_{ij}；

（5）计算新的隶属度 u_{ij}^{*}；

（6）将第五步所得的隶属度 u_{ij}^{*} 代入更新聚类中心 v_i；

不断重复步骤（3）到步骤（6）直到符合约束条件。

由于算法最终效果跟初始聚类中心和最佳聚类类数等初始参数高度相关，因此在 FCM 算法的具体实施过程中，使用蚁群算法来训练得到 FCM 聚类算法的初始聚类中心。同时使用 Xie-Beni 聚类有效性指标来确定最佳聚类类数。这两种方法能有效降低算法对初始参数的敏感度。

3.2 蚁群优化算法

3.2.1 蚁群算法介绍

蚁群算法首先由意大利学者 Dorigo M 等人提出[8]。它是通过对自然界中真实蚂蚁集体行为进行研究，进而得到启发而提出的一种基于种群的模拟进化算法。研究表明蚁群算法在求解很多优化问题方面具有优越性，其具有很强的求解能力。此外，蚁群算法是一种并行算法，不同个体通过“信息素”这种介质不断进行信息的交流和传递，也有利

于发现较好的解决[9]。

3.2.2 使用蚁群算法进行聚类基本思想

在使用蚁群算法进行聚类时，应该把每个反应区的像素点看成具有不同属性的蚂蚁，聚类中心则是蚂蚁所要寻找的“食物源”。所以，反应区各个像素点进行聚类的过程就类似于蚂蚁寻找食物源的过程。其中设 $X = \{X_i | X_i = (x_{i1}, x_{i2}, x_{i3}, \cdots, x_{im})\}$ 为某个待测物的反应区图像的像素集合。令 $d_{ij} = ||P(X_i - X_j)||_2$。使用蚁群算法来对图像进行聚类的具体算法如下所示[10]：

(1)初始化蚁群算法所需要的各个参数：设定 N，m，r，p_0, ε_0, β, α, $\tau_{ij}(0)_{=0}$ 等参数。其中 N 为待聚类的图像数据中的元素个数，m 为一个像素点所对应的值的维数，在本文中 m 取值为，r 代表聚类半径，p_0 为预设的阈值概率，ε_0 代表总体误差阈值，β 代表期望值启发式因子，α 代表信息启发式因子，$\tau_{ij}(0)$ 代表是 t 时刻像素点 X_i 到像素点 X_j 上残留的信息量，因为刚开始没有蚂蚁爬，所以所有的 $\tau_{ij}(0)$ 全都置为 0。

（2）计算任意点 X_i 到 X_j 的距离：

$$d_{ij} = \left\| P(X_i - X_j) \right\|_2 = \sqrt{\sum_{k=1}^{m} p_k (x_{ik} - x_{jk})^2} \quad \cdots\cdots (3.10)$$

其中 p_k 为第 k 个因子对应的权值，可以根据各个分量在聚类中的贡献不同而设定。

（3）计算各条路径上的信息量：

$$\tau_{ij}(t) = \begin{cases} 1, d_{ij} \le r \\ 0, d_{ij} > r \end{cases} \quad \cdots\cdots (3.11)$$

（4）计算 X_i 归并到 X_j 的概率：

$$p_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{s \in S} \tau_{sj}^{\alpha}(t)\eta_{sj}^{\beta}(t)} \quad \cdots\cdots (3.12)$$

(5)判断 $p_{ij}(t) \geqslant p_0$ 是否成立 $\cdots\cdots (3.13)$

(6)按照下式计算该类的聚类中心

$$\bar{C}_j = \frac{1}{J} \sum_{k=1}^{J} X_k \quad \cdots\cdots (3.14)$$

(7)按照下式计算第 J 个聚类的偏移误差；其中 c_{ji} 表示第 J 个聚类中心的第 i 个分量。

$$D_j = \sum_{k=1}^{J} \sqrt{\sum_{i=1}^{m} (x_{ki} - c_{ji})^2} \quad \cdots\cdots (3.15)$$

(8)计算总体误差

$$\varepsilon = \sum_{J=1}^{k} D_J \quad \cdots\cdots (3.16)$$

(9)判断$\varepsilon \leqslant \varepsilon_0$是否成立，若成立则停止迭代，并输出聚类个数 k 和聚类中心$\bar{C}_j$;若不成立，转(2)继续迭代。

3.3 Opencv 开源视觉库

OpenCV 库（Open Source Computer Vision Library），是一个基于 BSD 许可发行的跨平台计算机视觉库，可以运行在 Linux、Windows 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法[11]。

Opencv 主要拥有以下优势：

（1）纯 C 代码，源代码开放；

（2）拥有丰富的函数功能和强大的图像和矩阵运算能力。

（3）跨平台。Opencv 代码可以直接在 windows、Unix、Linux、MacOS、Sloaris 和 HP 平台之间相互移植。

（4）程序运行实时性。

（5）方便灵活的用户接口。

由于拥有上述优势特性，作为一个计算机视觉和模式识别的开源项目，Opencv 是一个理想的二次开发平台。而在本文中需要用到大量的图像处理工作，因此选用 Opencv 这个业界皆知的开源视觉库来辅助尿常规检测系统的开发工作。在本次开发中，主要使用了 Opencv 中的图像预处理函数、图像分割函数和很多基本的图像数据结构。

4. 尿常规检测系统的设计与实现

在本文中，设计并实现了一个基于图像分析的尿常规检测系统。该系统基于彩色图像处理技术和 Opencv 开源视觉库来进行尿常规检测和分析。该系统的核心功能包括：

（1）对用户采集的尿液样本进行分析，给出尿液中相应生化指标的数值并记入历史数据；（2）在系统客户端，用户登录之后可查看其它用户信息、历次检测的历史数据和以图表的形式对比历史数据，并且可通过系统提供的地图服务进行定位，查找周围的医院信息。

系统采用到的技术工具主要包括：

a)客户端开发：iOS API、Windows Phone API。

b)后台开发：ThinkPHP 框架。

c)图像处理：Opencv API。

d)第三方库：Masonry、AFNetworking、Realm、Mantle、BaiduMap 等。

e)系统环境：Linux、PHP、C++、XCode、MySQL、Apache 等。

下图 4-1 是系统的架构图。

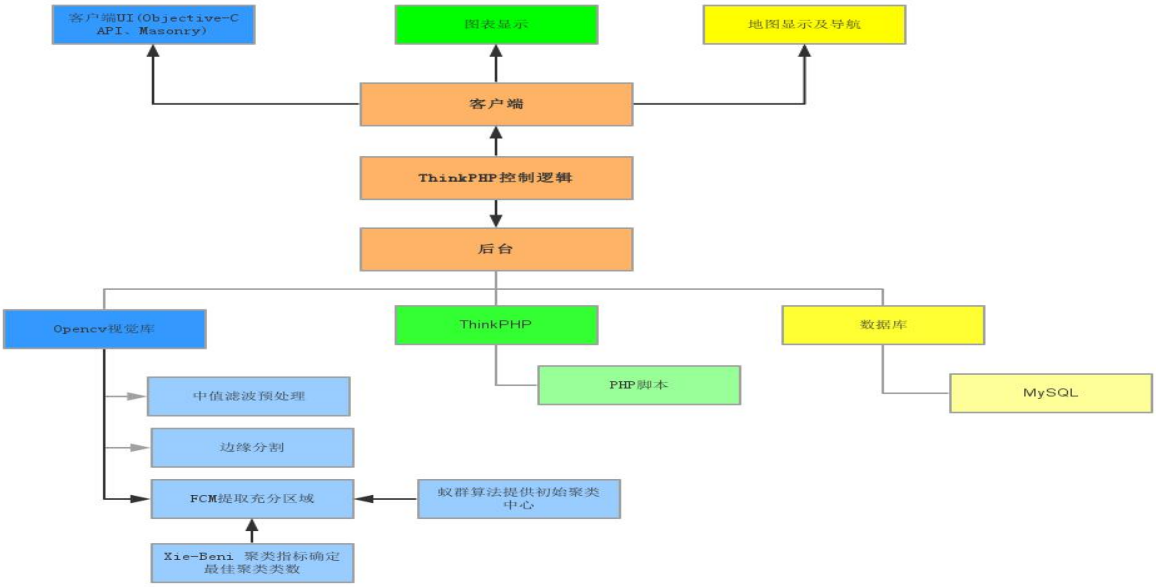

图 4-1　尿常规检测系统系统架构图

4.1 系统整体设计

本文设计并实现的尿常规检测系统主要由后台逻辑控制模块、数据持久化存储模块、后台图像处理模块和 iOS/Windows Phone 客户端模块等四个部分组成。后台逻辑控制模块主要是为了对从客户端传送来的图像数据、历史记录请求、健康贴士等信息进行加工处理，然后是将结果数据返还客户端或是进行持久化存储。数据持久化存储模块主要是存储并返还系统的注册用户信息、历史检测结果和健康贴士等数据。后台图像处理模块的作用则是对从客户端传送来的图像数据进行分割，计算提取反应最充分区域时的初始聚类中心，切割出反应最为充分的区域等。而客户端则用于提供系统使用说明和肾病相关的健康贴士、搜索附近的医院并提供相关导航、用户注册/登录并显示详细用户信息、采集图像数据并得到分析结果和查看检测的历史数据等。

使用本尿常规检测系统时，用户需要先打开本检测系统的客户端(如 iOS 客户端)并成功登录。然后使用客户端的后置摄像头采集图像数据，传送至后台逻辑控制模块。接下来后台的逻辑控制模块将图像数据转送至后台图像处理模块处理并将分析结果存入数据持久化存储模块。最后再将分析结果反馈给客户端并进行显示。相应过程的时序图如下图 4-2 所示。

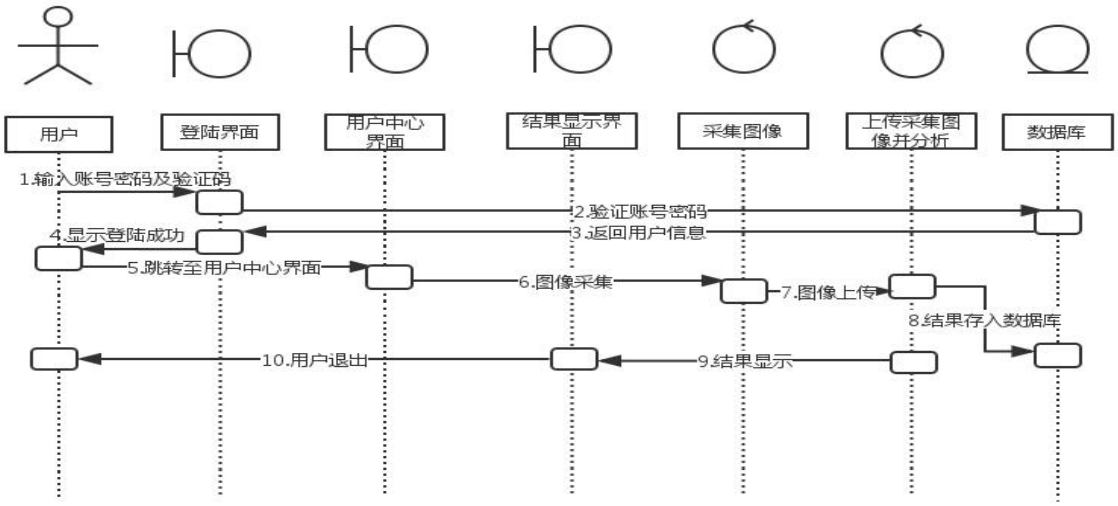

图 4-2 尿液检测操作的时序图

下面对系统中的各个模块进行简单介绍。

4.1.1 逻辑控制设计

本系统的逻辑控制设计采用的是轻量级的 PHP 开发框架 ThinkPHP。ThinkPHP 是一种典型的 MVC 开发架构模式。

MVC（Model View Controller，MVC）是一种软件设计典范。用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面。在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

ThinkPHP 是一个快速、兼容而且简单的轻量级国产 PHP 开发框架，诞生于 2006 年初，原名 FCS，2007 年元旦正式更名为 ThinkPHP。遵循 Apache2 开源协议发布，从 Struts 结构移植过来并做了改进和完善，同时也借鉴了国外很多优秀的框架和模式，使用面向对象的开发结构和 MVC 模式，融合了 Struts 的思想和 TagLib(标签库)、RoR 的 ORM 映射和 ActiveRecord 模式[12]。ThinkPHP 的核心处理流程如下图 4-3[12]所示：

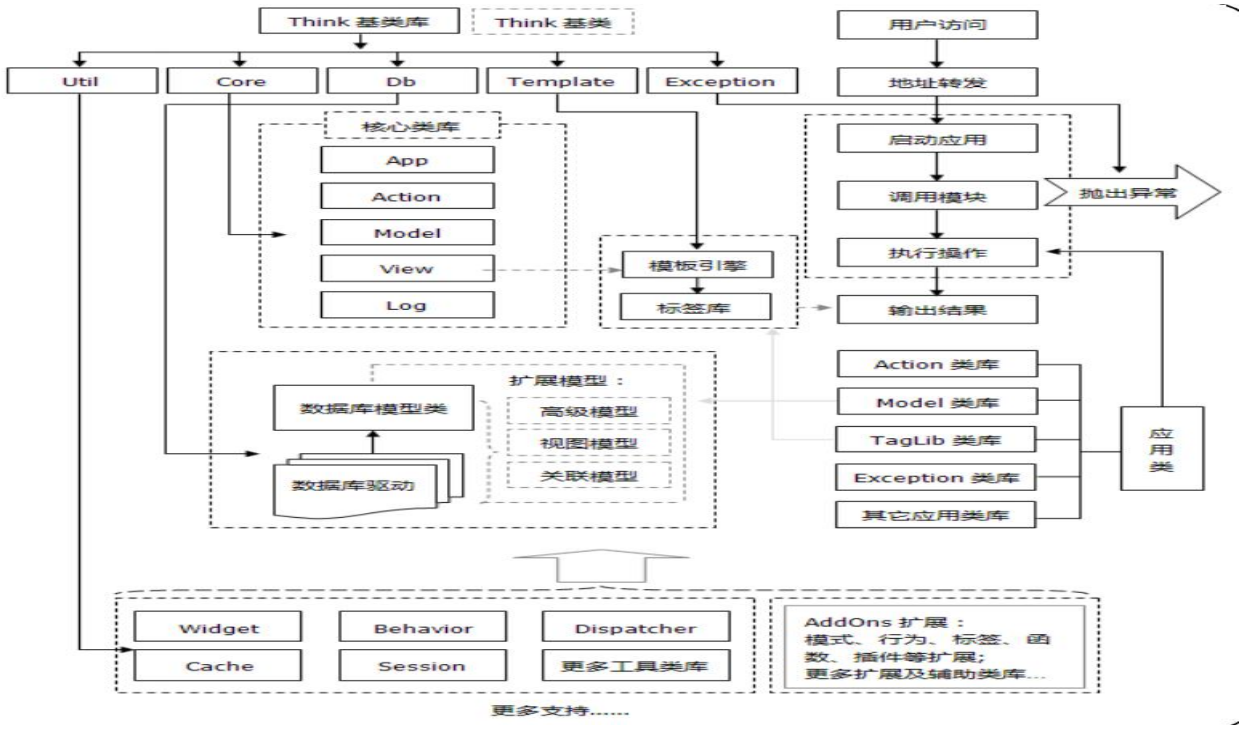

图 4-3 ThinkPHP 处理流程图

具体操作过程描述：

（1）加载公共入口文件。在后台程序刚开始调用 ThinkPHP 框架时所作的第一件事情就是加载 ThinkPHP 已经定义好的公共入口文件；

（2）项目初始化 Init；

（3）项目预编译；

（4）因为 ThinkPHP 统一了 URL 入口地址，所以需要进行 URL 分析 Dispatcher；

（5）获取模块和操作名；

（6）项目执行 exec；

（7）执行控制器的操作；

（8）调用模块获取数据；

（9）返回数据给客户端，用于显示。

4.1.2 数据持久层设计

由于在本系统中使用的 ThinkPHP 框架对 MySQL 的操作又进行了一层封装，其进行 CURD 操作时跟直接使用 PHP 来进行 SQL 操作有较大的不同。在 ThinkPHP 中基础的模型类就是 Think\Model 类，该类完成了基本的 CURD、ActiveRecord 模式、连贯操作和统计查询等功能，同时一些高级特性也被封装到另外的模型扩展中。

在本系统中，我们主要使用了 ThinkPHP 的如下功能的 API：模型定义、模型实例化、连接数据库，进行 CURD 操作和字段映射等。ThinkPHP 通过定义一个模型类来构造和初始化一张数据表，然后将增、删、查、改等操作封装到已定义的模型类中。

MySQL 的查询执行流程图如下图 4.4 所示：

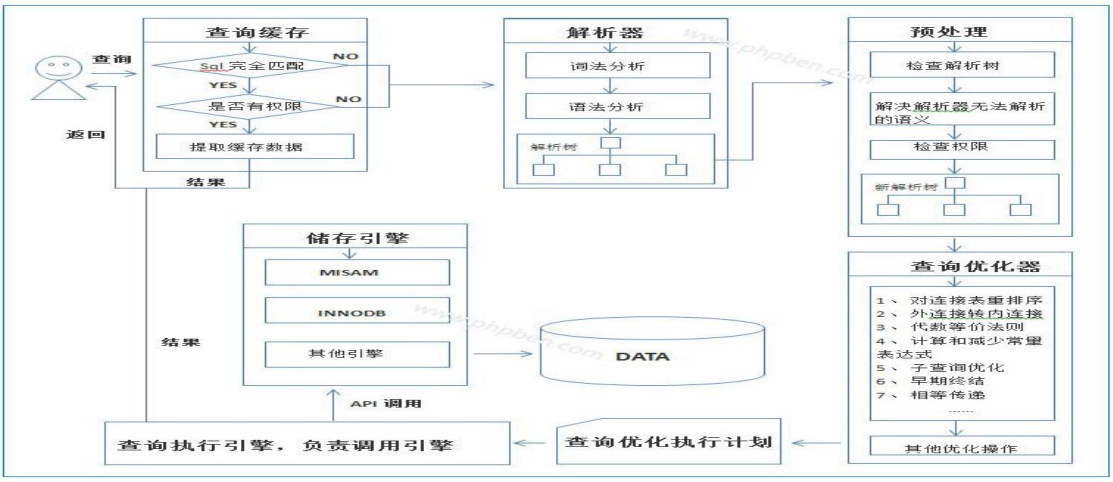

图 4-4 MySQL 查询流程图

4.1.3 尿常规检测系统客户端设计

本尿常规系统的客户端有 iOS 和 Windows Phone 两个版本。两个 APP 客户端的大致架构大致相同，故重点只介绍 iOS 版本。

在 Windows Phone 版本中主要实现了使用指导说明、采集试纸图像以及将图像上传、显示分析结果、登录/注册和对历史数据进行存储等功能。

而 iOS 版本则实现了使用说明，用户登录/注册、采集试纸图像以及将图像上传、显示分析结果、查看历史记录和进行定位，查看附近医院并进行路径规划等功能。

在 iOS 版的 APP 中，应用主要分成如下几层：网络层，用于向后台发送采集到的图片和从后台接收数据；模型层，用于将从后台取回的数据转成合适的格式；控制器层，用于协调视图层和模型层的数据交换。

下图 4-5 为客户端的架构设计图。

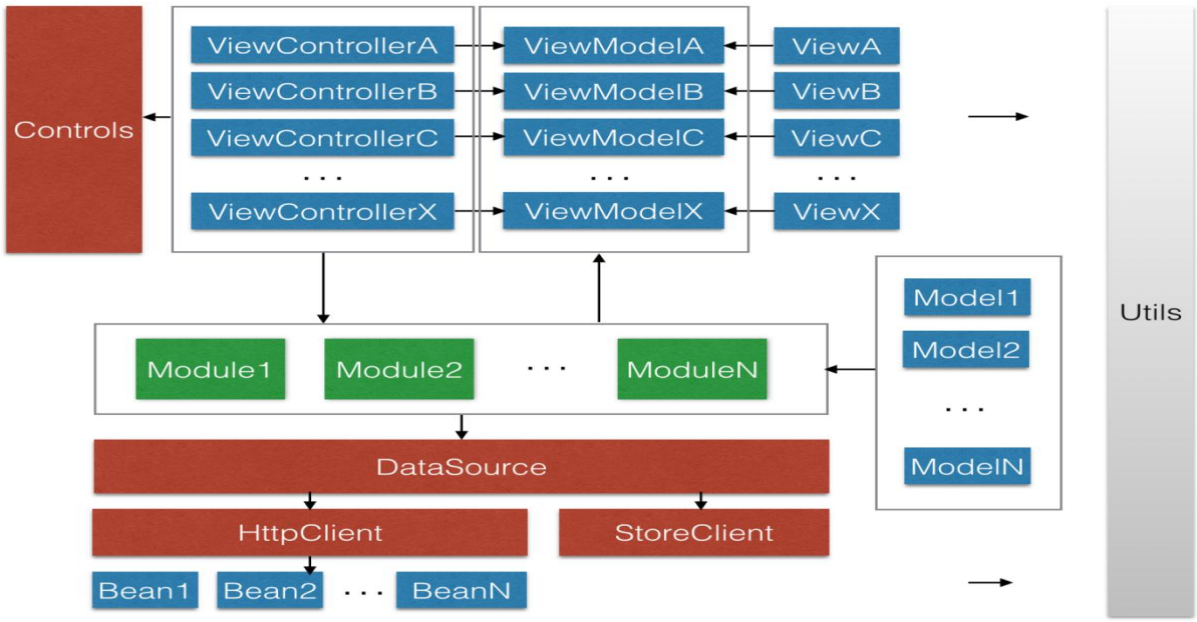

图 4-5 客户端架构图

同时在开发 iOS 版的 APP 时还使用了 AFNetworking、Masonry、Realm 等第三方开源库。下面分别对其进行简单介绍：

（1）AFNetworking：AFNetworking 是一个轻量级的网络库,适用于 iOS 以及 Mac OS X。它构建于在 NSURLConnection、NSOperation、以及其它熟悉的 Foundation 技术之上。它拥有良好的架构，丰富的 API，以及模块化构建方式，使得其使用起来非常轻松[13]。

在项目开发过程中，系统客户端使用了 AFNetworking 的 AFURLSessionManager、AFHTTPSessionManager 和 〈AFURLRequestSerialization〉 下面的 AFHTTPRequestSerializer 等组件的 API 来完成网络传输中的上传和下载等功能。

（2）Masonry：Masonry 是一个轻量级的布局框架，它使用了一个对使用者更加友好的语法包装了 iOS 中的 AutoLayout。Masonry 拥有自己的布局 DSL，以提供一种链式的更加准确和可读的方式来描述使用者自己的 NSLayoutConstraints。Masonry 支持 iOS 和 Mac OS X 两种系统[14]。

（3）Realm：Realm 是一个可以直接运行在手机、平板电脑和可穿戴式设备的移动数据库，它具有移动优先(Realm 是第一个直接落地，能够直接运行在手机、平板电脑和可穿戴式设备的数据库)、简单、现代(Realm 支持关系型、通用型、向量型数据库甚至 Swift)和快速等特点[15]。

另一方面，为了在系统客户端中引入地图功能，在实现过程中引入百度地图的第三方库的 API，以方便更加高效、快速地进行地图相关的开发。

（4）百度地图 iOS SDK 是一套基于 iOS 5.0 及以上版本设备的应用程序接口，不仅提供展示地图的基本接口，还提供 POI 检索、路径规划、地图标注、离线地图、定位、周边雷达等丰富的 LBS 能力。在开发过程中，可以使用百度地图 iOS SDK 开发适用于移动设备的地图应用，通过接口，可以轻松访问百度服务和数据，构建功能丰富、交互性强的地图应用程序。而且该套 SDK 提供的服务是免费的，任何非营利性程序均可使用[16]。

4.1.4 尿常规检测系统后台图像处理模块设计

在本文设计并实现的尿常规检测系统中，图像处理模块是整个工作的核心，这一部分的实现的质量将直接并极大地影响着系统的检测结果的准确度。图像处理模块的大致处理流程如下所示：

（1）将采集到的图像进行平滑去噪操作；

（2）将图像从 RGB 彩色空间变换到 L*a*b*彩色空间；

（3）使用 Canny 边缘检测方法和固定坐标法对试纸部分进行分割；

（4）再使用蚁群算法来提取各种待测指标反应模块的聚类初始中心点；

（5）使用改进的 FCM 算法提取出试纸中各种待测指标的反应模块中的反应最充分的部分进而提取出颜色信息并和样本进行匹配。

在后台图像处理模块中使用了蚁群算法、改进过的 FCM 算法和 Opencv 开源视觉库。图像处理模块中各个部分的关系和信息流动顺序如下图 4-6 所示。

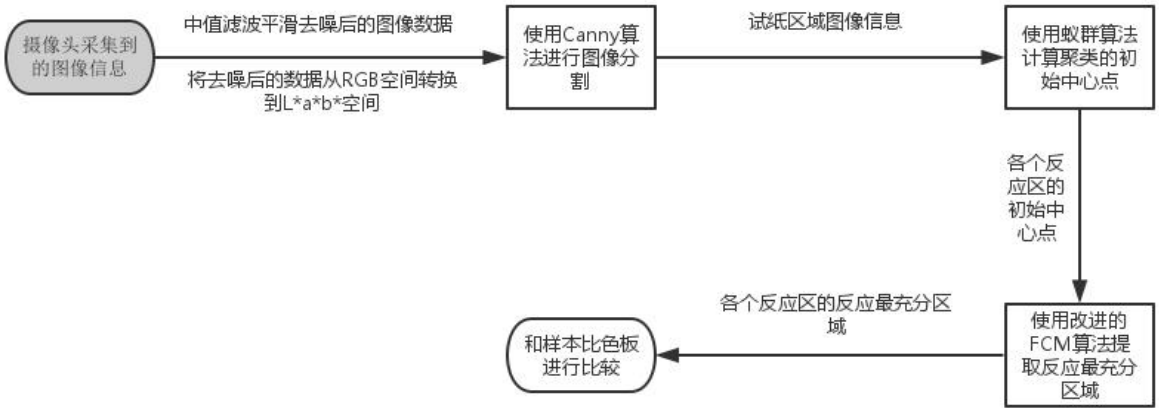

图 4-6 图像处理模块信息流动图

4.2 系统详细设计

本尿常规检测分析系统的核心工作就是对采集到的试纸图像信息进行分析处理，图像分析是本检测系统的基础。

4.2.1 图像处理部分

图 4-7 给出了图像处理的执行流程，其主要步骤如下。

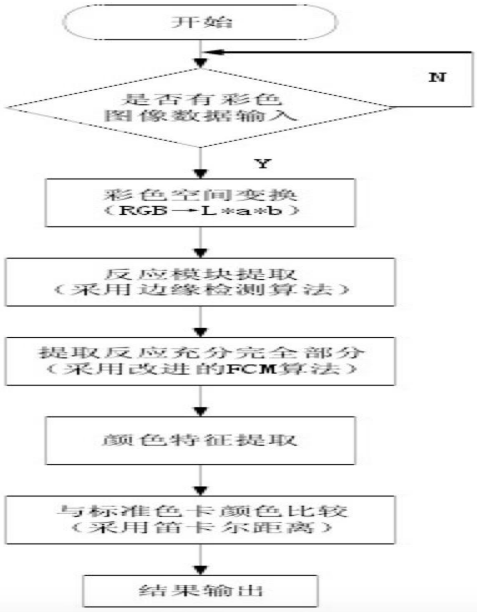

图 4-7 图像处理流程图

a)图像载入部分

图像处理算法的执行步骤为先判断是否有彩色图像输入，调用 Opencv 中的函数 cvLoadImage 之后判断其返回值是否为 null 来判断是否载入成功。其函数原型为

CVAPI(IplImage*) cvLoadImage(const char* filename, int iscolor);

具体示例代码如表 4-1 所示。

表 4-1 载入图像代码

```
IplImage* src=cvLoadImage("e:\\demo.jpg");
if (src == NULL){
return;
}else{
IplImage* dst=cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
cvCopy(src,dst);
FeatureData *pre=new FeatureData(src);
}
```

b)预处理部分

成功载入图像数据之后需要对采集到的图像进行图像预处理操作，直接调用 Opencv 封装好的标准函数 cvSmooth。cvSmooth 函数的函数原型为

```
CVAPI(void) cvSmooth( const CvArr* src, CvArr* dst,
                      int smoothtype CV_DEFAULT(CV_GAUSSIAN),
                      int size1 CV_DEFAULT(3),
                      int size2 CV_DEFAULT(0),
                      double sigma1 CV_DEFAULT(0),
                      double sigma2 CV_DEFAULT(0));。
```

在本系统中使用了中值滤波算法来对采集到的图像信息进行去噪处理。这可通过将 cvSmooth 函数中的 smoothtype 参数赋值 CV_MEDIAN 来实现。具体实现代码表 4-2 所示：

表 4-2 预处理代码

```
void Pretreatment::Denoising()
{
Img_Denoising=cvCreateImage(cvGetSize(this->Img_Src),this->Img_Src->depth,this->Img_Src->nChannels);
// cvCvtColor(this->Img_Src,this->Img_Denoising,CV_RGB2GRAY);
cvSmooth(this->Img_Src,this->Img_Denoising,CV_GAUSSIAN,3,3);
}
```

c)彩色空间的转换

接下来是对图像的颜色空间进行转化，将图像的 RGB 空间转换为 CIE 定义的 L*a*b*空间。通过调用 Opencv 中的色彩空间转换函数来完成，其函数原型为

```
CVAPI(void)  cvCvtColor( const CvArr* src, CvArr* dst, int code );
```

其中的 code 参数为 CV_RGB2Lab。

具体实现代码如表 4-3 所示。

表 4-3 彩色空间转换

```
Img_Lab=cvCreateImage(cvGetSize(this->Img_Src),this->Img_Src->depth,
this->Img_Src->nChannels);
    cvCvtColor(this->Img_Src,this->Img_Lab,CV_RGB2Lab);
```

d)边缘分割部分

在试纸彩色图像经过颜色空间变换后，需要把发生特异性颜色反应的模块从背景中提取出来。由于反应模块颜色与背景颜色差别很大，有较明显的边界线，因此选用边缘

检测算法来完成模块提取工作。在本文中，边缘分割部分采用了 Canny 微分算子来检测边缘。Canny 算子只涉及加减法和求梯度运算，不仅计算简单方便而且具有非线性特征和平滑图像的功能，因此在图像分割中得到了广泛的应用。边缘检测算法流程如下图 4-8 所示。

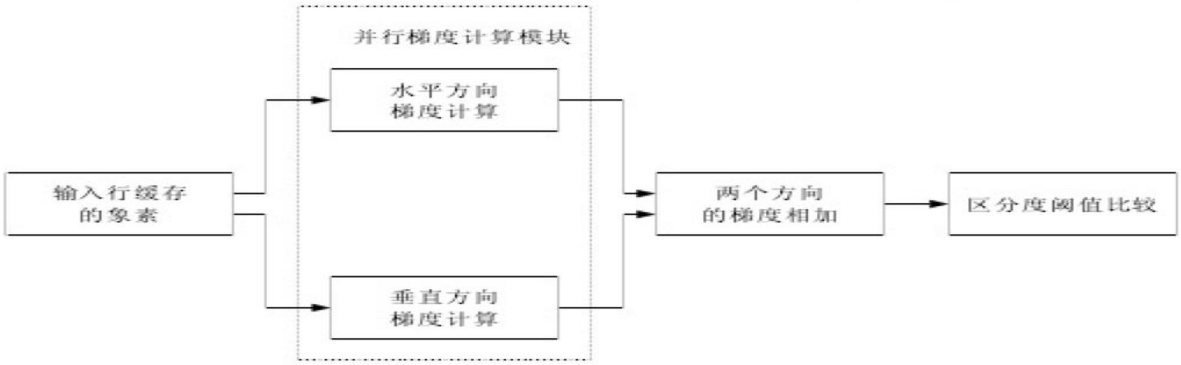

图 4-8 边缘检测算法流程图

Canny 边缘检测的基本原理为根据对信噪比与定位乘积进行测度，然后得到最优化逼近算子。在 Canny 算子边缘检测之前得对图像数据进行去噪处理。

在本尿常规检测系统中使用的 Canny 边缘检测算法选取的窗口大小是 3*3，对 3*3 窗口中的图像数据进行一阶偏导的有限差分计算梯度的幅值和方向，然后使用双阈值算法检测和连接边缘，最后输出分割结果。

使用 Canny1 微分算子来对图像进行分割亦可调用 Opencv 库函数，其函数原型为

```
(void)  cvCanny( const CvArr* image, CvArr* edges, double threshold1,
                 double threshold2, int  aperture_size CV_DEFAULT(3) )。
```

项目中的实例代码如表 4-4 所示：

表 4-4 边缘分割代码

```
pCannyImg = cvCreateImage(cvGetSize(pImg), IPL_DEPTH_8U, 1);
cvSmooth(pImg,dstImg,CV_MEDIAN,3,3);
cvCanny(pImg, pCannyImg, 32, 80,3);
cvNamedWindow(“Canny");
cvShowImage("Canny", pCannyImg);
```

e)FCM 提取充分反应部分

在对图像进行边缘分割之后，接下来就是提取反应充分完全部分。在基于图像处理的尿常规分析系统中，彩色图像处理方法是整个系统的重点，同时也是软件系统设计的核心。因为模块内部颜色特点(颜色差别较小，边界较为模糊)，本文采用了模糊 C 均值

聚类算法(FCM 算法)来对其进行分割。为了提高检测精度，还针对本系统所采集的图像的特点进行了一定改进，将空间信息引入了标准的 FCM 聚类算法中。

在 FCM 算法中，根据相关研究[17]，模糊加权指数 m 的最佳取值范围为[1.5,2.5]，所以在本系统中取 m=2。而最大迭代次数取 150 次，目标函数 J_m的收敛误差为 2.0，计算两点距离所用的 A 取单位矩阵，即两点之间的距离为欧式距离。聚类类数 C 根据 Xie-Beni 指标选取，取值为 2。初始聚类中心采用蚁群算法选取，邻域大小为 3*3 个像素，彩色空间选用 CIE 推荐的 L*a*b*空间。

图 4-9 为 FCM 聚类算法的执行流程图。

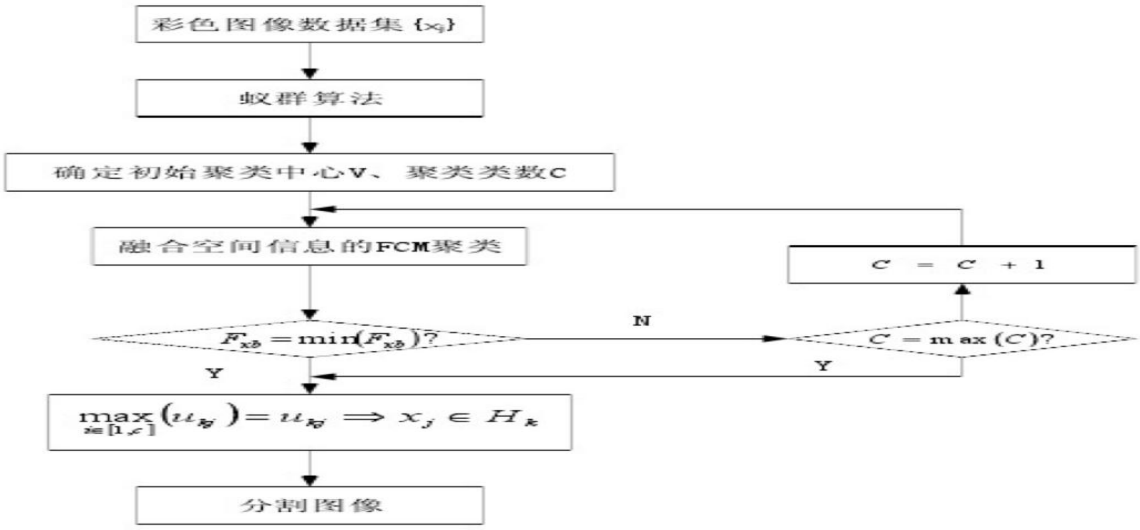

图 4-9 FCM 聚类算法流程图

在得到反应最充分完全的区域后进行的操作就是将该区域所有像素的 LAB 值求平均，然后和比色板上的颜色比较，取其欧式距离最近的一点。最后输出图像分析的结果。

FCM 算法部分的头文件如表 4-5 所示。

表 4-5 FCM 头文件

```
    typedef enum {
    DUDFCMSegmentInitKmeansPP = 0,
    DUDFCMSegmentInitACOPP
}DUDFCMSegmentInitType;
    //FCM 算法模块应该是一个工具类，此时可以将其设计成一个单例类
    class   DUDFCMSegment{
    public:
    static DUDFCMSegment * GetInstance();
    int getcheck();
    IplImage* getFTA(IplImage* src);        //返回经过 FCM 算法处理的图像数据
    public:
```

```
void setParameters(const cv::Mat &rows,
const int number_clusters,
const float fuzziness,
const float epsilon,
const DUDFCMSegmentInitType init_type);                    //设置FCM算法需要的参数，
如图像矩阵、聚类数、模糊指数和算法停止阈值
bool canStop();         //判断是否停止算法
void clustering (const unsigned int num_iteration = 10000);     //进行循环计算
void initEverything ();              //初始化聚类中心
void initKmeansPP ();                //使用K-means算法来计算初始聚类的中心点
void initACOPP ();                   //使用蚁群算法来计算初始聚类的中心点
void computeCentroids ();            //迭代计算每个反应区图像的中心点
bool updateMembership ();            //用于更新像素点属于哪个聚类群
inline const Mat get_centroids_ () { return centroids; }
inline const Mat get_membership_ () { return membership; }
inline const Mat get_new_membership_ () { return new_membership; }
private:
//用于构造单例类
DUDFCMSegment(){};
static DUDFCMSegment* m_pInstance;          //全局类实例
private:
//用于设置FCM算法运行时的部分参数
int check;
static const int GET_IMAGE_OK = 1;
static const int GET_IMAGE_FAILED = 2;
float instanceFuzziness;    //模糊加权指数，决定算法的模糊程度
float threshold;            //FCM算法停止运行的阈值
int numberOfClusters;       //每个反应区图像的聚类数
int number_points;          //每个待测物反应区图像的像素个数
int dimension;              //每个像素点向量的维数
cv::Mat centroids;          //每个聚类区域的中心点
cv::Mat membership;         //每个像素点的隶属度矩阵
cv::Mat new_membership;     //更新后的隶属度矩阵,两者交替来计算是否需要停止算法
cv::Mat rows;               //这里就是图像数据
```

```
DUDFCMSegmentInitType initType;
private:
void FillInternalContours(IplImage *pBinary,double dAreaThre);      //对边缘进行填充，如果没有返回封闭的边缘，则使其闭合
float calc_dist(const cv::Mat &point,const cv::Mat &center);         //计算某点与中心点的距离
};
```

迭代计算每个反应区图像的中心点

```
void DUDFCMSegment::computeCentroids(){
cv::Mat u_ji_m  = cv::Mat::zeros (number_points, numberOfClusters, CV_32FC1);
//  Initialization
for (int j = 0; j < numberOfClusters; j++)
for (int f = 0; f < dimension; f++)
centroids.at<float> (j, f) = 0.0;
// weight ** fuzziness
for (int j = 0; j < numberOfClusters; j++)
for (int i = 0 ; i < number_points; i++)
u_ji_m.at<float> (i, j) = pow (membership.at<float> (i, j), instanceFuzziness);
// Update centroids
for (int j = 0; j < numberOfClusters; j++)
for (int i = 0 ; i < number_points; i++)
for (int f = 0; f < dimension; f++)
centroids.at<float> (j, f) += u_ji_m.at<float> (i, j) * rows.at<float> (i, f);
// Normalization
float normalization;
for (int j = 0; j < numberOfClusters; j++){
normalization = 0.0;
for (int i = 0 ; i < number_points; i++)
normalization += u_ji_m.at<float> (i, j);
for (int f = 0; f < dimension; f++)
centroids.at<float> (j, f) /= normalization;
}}
```

更新像素点属于哪个聚类群

```
bool DUDFCMSegment::updateMembership(){
```

```
    cv::Mat matrix_norm_one_xi_minus_cj = cv::Mat::zeros (numberOfClusters,
number_points, CV_32FC1);
    // Initialization
    for (unsigned int i = 0 ; i < number_points; i++)
    for (unsigned int j = 0; j < numberOfClusters; j++)
    matrix_norm_one_xi_minus_cj.at<float> (j, i) = 0.0;
    for (unsigned int i = 0 ; i < number_points; i++) {
    // Calculate distances from each cluter.
    cv::Mat point = rows.row (i);
    for (unsigned int j = 0; j < numberOfClusters; j++) {
    cv::Mat center = centroids.row (j);
    matrix_norm_one_xi_minus_cj.at<float> (j, i)
    = calc_dist (point, center);
    }
    }
    float coeff;
    //计算每个点对每个聚类的隶属度
    for (unsigned int i = 0 ; i < number_points; i++) {
    for (unsigned int j = 0; j < numberOfClusters; j++) {
    coeff = 0.0;
    for (unsigned int k = 0; k < numberOfClusters; k++) {
    coeff +=
    pow ( (matrix_norm_one_xi_minus_cj.at<float> (j, i) /
    matrix_norm_one_xi_minus_cj.at<float> (k, i)) ,
    2.0 / (instanceFuzziness - 1.0) );
    }
    new_membership.at<float> (i, j) = 1.0 / coeff;
    }}
    if (!canStop() ){
    membership = new_membership.clone ();
    return false;
    }
    return true;
}
```

f)蚁群算法部分

蚁群算法(ant colony optimization, ACO)，是一种用来在图中寻找优化路径的概率型算法。蚁群算法是一种模拟进化算法，初步的研究表明该算法具有许多优良的性质。众多研究结果表明，蚁群算法具有一种新的模拟进化优化方法的有效性和应用价值。根据相关研究，蚁群算法在数据聚类方面能够取得较好的效果。因此，在本系统中，利用改进后的蚁群算法来对各个反应区图像进行聚类分析，计算出每个反应区的聚类中心点，以供后面为了提取反应最充分部分而进行的 FCM 算法操作使用。在蚁群算法部分需要设定的初始值有 N、m、r、p_0、ε_0、β、α、$\tau_{ij}(0)=0$ 等。其中 N 代表每个反应区的像素个数，m 代表每个像素向量的维数，r 代表聚类半径、p_0代表概率阈值、ε_0代表统计误差阈值、β 和 α 是蚁群算法中的启发因子参数。在本系统中 N 被设定为，m 因为向量有三维，所以被设定为 3。同时根据相关实验结果，r 被设为 12.3，p_0被设为 0.60，ε_0被设定为 0.18，β 和 α 都被设定为 1。蚁群算法的执行流程图如下图 4-10 所示。

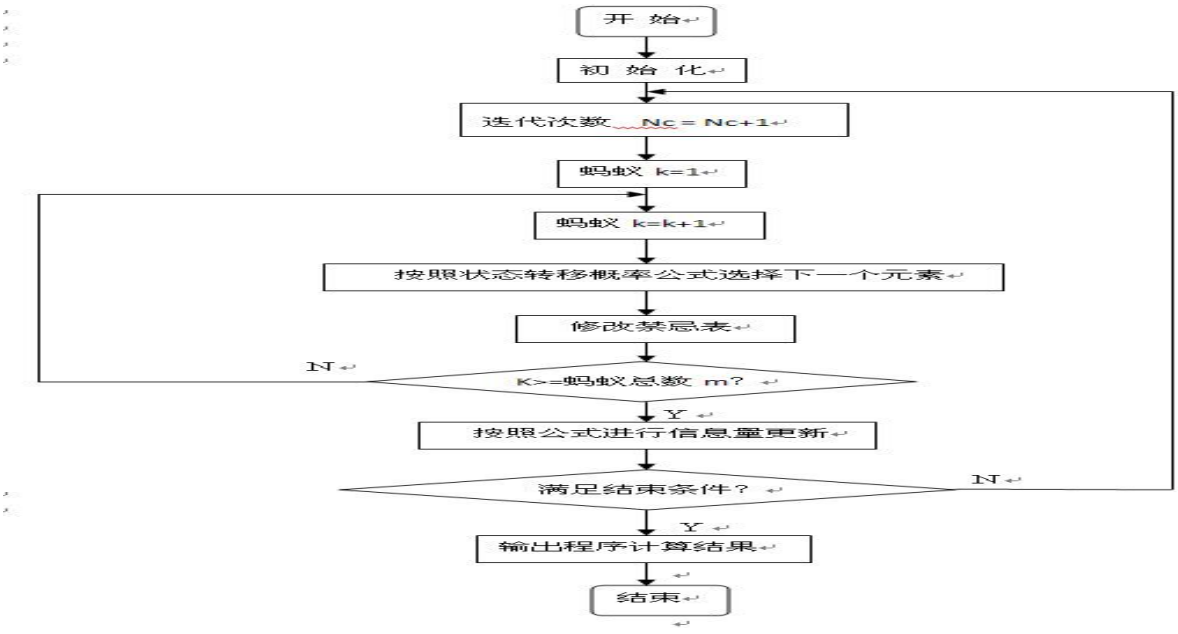

图 4-10 蚁群算法流程图

在本尿液检测系统中，按照图 4-10 所示的算法流程图所实现的蚁群算法如下表 4-6 所示(部分代码)。

(A) 头文件

表 4-6 蚁群算法头文件

```
#ifndef DUDACOClustering_hpp
#define DUDACOClustering_hpp
#include <opencv2/opencv.hpp>
#include <iostream>
```

```
#include "DUDImageCommon.h"
using namespace std;
using namespace cv;
typedef enum{
    DUDACOClusteringTypeURO = 0,                    //尿胆原
    DUDACOClusteringTypeBIL,                        //胆红素
    DUDACOClusteringTypeKET,                        //酮体
    DUDACOClusteringTypeBLD,                        //潜血
    DUDACOClusteringTypePRO,                        //蛋白质
    DUDACOClusteringTypeNIT,                        //亚硝酸盐
    DUDACOClusteringTypeWBC,                        //白细胞
    DUDACOClusteringTypeGLU,                        //葡萄糖
    DUDACOClusteringTypeSG,                         //比重
    DUDACOClusteringTypePH,                         //pH
    DUDACOClusteringTypeVC                          //抗坏血酸
}DUDACOClusteringType;
typedef struct {
    double LWeight;                                  //L 分量的权重
    double AWeight;                                  //a 分量的权重
    double BWeight;                                  //b 分量的权重
}DUDACOClusteringWeight;
typedef struct {
    int elemNumber;                                  //该聚类中数据的个数
    int *elementPoint;                               //这个动态数组的元素就是属于该聚类的
像素点的下边索引
}DUDACOClusteringInfo;
DUDACOClusteringWeight DUDACOClusteringWeightArray[12] = {
    {0.9,1.1,1},
    {1.2,1,0.9},
    {1.2,1,1},
    {1,1.2,1.1},
    {1.3,0.97,1.1},
    {1.1,2.2,1},
    {1.2,0.9,1.35},
```

```
    {1.53,2.5,0.9},
    {1.7,1.2,1.1},
    {0.97,1.5,1.1},
    {1,2.4,0.9},
    {1.4,1.1,2}
};
class DUDACOClustering{
public:
    static  DUDACOClustering * GetInstance();
    void setParameters(const int numberOfPixels,
                       const int numberOfDimension,
                       const double radiusThreshold,
                       const int errorThreshold,
                       const double alphaFactor,
                       const double betaFactor,
                       const double probabilityThreshold,
                       const DUDACOClusteringType ACOType,
                       IplImage *srcImage);          //设置各种预定参数
    cv::Mat getCenterPoint(DUDACOClusteringType type);
//得到聚类的初始中心点
    void acoAlgorithm();                                          //启动蚁群算法
private:
    DUDACOClustering(){};
    static DUDACOClustering* m_pInstance;                         //全局单例
private:
    void calc_Dist(int indexI,int indexJ);                        //计算第 i 个点到第 j
个点的距离
    void calc_Info(int indexI,int indexJ);                        //计算第 i 个点到第 j
个点的信息量
    void calc_Probability(int indexI,int indexJ);                 //计算将 Xi 归并到 Xj
的概率
    bool isGreaterThanThreshold(double probability);              //判断一个概率是否大
于预制的概率阈值
    cv::Mat calc_Centroid(int clusterIndex);                      //计算第 J 个聚类的中
```

```
心点
    double getErrorDistance(int cluster);                    //计算第 J 个聚类的偏
离误差
    double getTotalErrorDistance();                          //计算总体误差
private:
    //用于设置蚁群算法运行时的参数
    int numberOfPixels;                                      //总的像素个数
    int numberOfDimension;                                   //每个像素点的维数
    double radiusThreshold;                                  //计算信息量所需要的
半径阈值
    int errorThreshold;                                      //总体误差阈值
    double alphaFactor;                                      //信息启发因子
    double betaFactor;                                       //期望启发因子
    double probabilityThreshold;                             //判断像素 i 和像素 j
是否是一类所使用的阈值
    int imageWidth;                                         //这幅图像的宽度
    int imageHeight;                                        //这幅图像的高度
    int imageSize;                                          //图像的像素个数
private:
    DUDACOClusteringType acoType;                           //不同的检测物质在不同的
分量上拥有不同的权重
    int totalCluster;                                        //聚类个数
    cv::Mat *arrayPoint;                                     //指向一个动态的保存
各个聚类中心点的数组
    IplImage *srcImage;                                     //保存载入的图像数据
    float **distance;                                       //保存任意两个像素点的距离
    int **infoMatrix;                                       //每个像素点的信息素矩阵
    int *clusterIndexArray;                                 //保存每个像素点属于的聚类
类数的索引，初始值为-1
    DUDACOClusteringInfo *infoArray;                        //这个数组保存着属于某聚
类的所有像素点索引
    float **probMatrix;
};
#endif /* DUDACOClustering_hpp */
```

(B)核心函数

计算图像数据的任意两点的距离的代码如表 4-7 所示。

表 4-7 蚁群算法计算两点的距离

```
//计算任意两点的加权距离
void DUDACOClustering::calc_Dist(int indexI, int indexJ){
    int rowIndexI = indexI / this->imageWidth;
    int columnIndexI = indexI % this->imageWidth;
    int rowIndexJ = indexJ / this->imageWidth;
    int columnIndexJ = indexI % this->imageWidth;
    CvScalar pixelI = cvGet2D(this->srcImage, rowIndexI, columnIndexI);
    CvScalar pixelJ = cvGet2D(this->srcImage, rowIndexJ, columnIndexJ);
    DUDACOClusteringWeight pixelWeight = DUDACOClusteringWeightArray[acoType];
    float f_dist = 0.f;
    for (int d = 0; d < this->numberOfDimension; d++) {
        float t = 0;
        switch (d) {
            case 0:
                t = pixelWeight.LWeight * (pixelI.val[d] - pixelJ.val[d])*(pixelI.val[d] - pixelJ.val[d]);
                break;
            case 1:
                t = pixelWeight.AWeight * (pixelI.val[d] - pixelJ.val[d])*(pixelI.val[d] - pixelJ.val[d]);
                break;
            case 2:
                t = pixelWeight.BWeight * (pixelI.val[d] - pixelJ.val[d])*(pixelI.val[d] - pixelJ.val[d]);
                break;
            default:
                break;
        }
        f_dist += t;
    }
    this->distance[indexI][indexJ] = f_dist;
    //在计算两点距离的时候就计算信息素矩阵
    calc_Info(indexI, indexJ);
}
```

计算信息素矩阵的代码如表 4-8 所示。

表 4-8 计算信息素矩阵

```
oid DUDACOClustering::calc_Info(int indexI, int indexJ){
    if (this->distance[indexI][indexJ] <= this->radiusThreshold) {
        this->infoMatrix[indexI][indexJ] = 1;
    }else{
        this->infoMatrix[indexI][indexJ] = 0;
    }
}
```

计算像素点 X_i 归并到 X_j 概率的代码如表 4-9 所示。

表 4-9 计算像素点 X_i 归并到 X_j 概率

```
void DUDACOClustering::calc_Probability(int indexI, int indexJ){
    float sum = 0,numerator = 0;
    numerator = pow(this->infoMatrix[indexI][indexJ], this->alphaFactor) + pow(1/this-
>distance[indexI][indexJ], this->betaFactor);
    //获取当前聚类的索引数
    int clusterIndex = this->clusterIndexArray[indexJ];
    //获取当前聚类的结构数据
    DUDACOClusteringInfo currentCluster = this->infoArray[clusterIndex];
    int elementNum = currentCluster.elemNumber;
    //计算分母
    for (int i = 0; i < elementNum; i++) {
        sum += pow(this->infoMatrix[currentCluster.elementPoint[i]][indexJ], this->alphaFactor)
                + pow(1/this->distance[currentCluster.elementPoint[i]][indexJ], this->betaFactor);
    }
    this->probMatrix[indexI][indexJ] = numerator / sum;
}
```

4.2.2 后台部分

由于后台使用的 ThinkPHP 开源框架，其中包含了很多预定义的文件，所以在此只描述和系统关系密切的后台部分实现。

（1）数据表设计部分

在本系统中共有两张数据表，一张是 user 表，字段如下表 4-10 所示。

表 4-10 user 字段表

userName	password	age	telephone

另一张表是 historyInformation，其字段如下表 4-11 所示。

表 4-11 historyInformation 字段表

detectionTime	userName	niaodanyuanValue	danhongsuValue	tongtiValue	qianxueValue	danbaizhiValue
yaxiaosuanyanValue	baixibaoValue	putaotangvalue	bizhongValue	phValue	kanghuaixuesuanValue	jiaozhunquValue

两张表字段类型的详细说明如表 4-12 和 4-13 所示。

表 4-12 User 表字段说明

User 表	
userName	text
password	text
age	int(11)
telephone	text

表 4-13 historyInformation 表字段说明

historyInformation 表	
detectionTime	timestamp
userName	text
niaodanyuanValue	text
danhongsuValue	text
tongtiValue	text
qianxueValue	text
danbaizhiValue	text
yaxiaosuanyanValue	text
baixibaoValue	text
putaotangvalue	text
bizhongValue	text
phValue	text
kanghuaixuesuanValue	text
jiaozhunquValue	text

（2） 后台文件部分

在本系统的后台中，处理相关业务逻辑的主要有四个 PHP 文件，分别为用户注册文件、用户登录文件、获取历史数据文件和后台图像处理部分文件。

用户注册部分文件主要用于处理用户的注册细节，包括接受用户客户端上传的用户名、密码、用户年龄和电话号码等信息，判断是否已经存在相同用户名的用户、接收用户头像并将这部分信息存入数据库中。

用户登录部分文件主要处理用户登录相关的事宜，其中主要包括以下操作：接收从客户端传来的用户名和密码、连接数据库、执行 SQL 语句进行验证然后将相关信息返回到客户端。

获取历史数据部分主要处理用户从后台获取账号对应的历史数据的事宜。具体操作包括接收从客户端传来的用户名和密码、连接数据库、从数据库中取回相关信息并返回给客户端。

后台的图像处理部分主要用于接收从客户端传来的试纸反应图像，并将结合 Opencv 编写的图像处理模块载入以进一步对图像进行处理，最后将检测分析的结果返回给用户的客户端并存入历史数据的数据库中。

（3） 后台 PHP 调用 C++静态库

由于后台是使用 PHP 脚本语言写成的，而在本检测系统中的 Opencv 图像处理模块是 C 或者 C++编写的。因此必须得把所写的 C++代码编译成动态链接库(如.so 文件)。然后 php 通过创建一个新的扩展，并在扩展里面调用该.so 文件，同时对外暴露出 php 的函数接口。其具体操作步骤如下所示。

(a)将共享库.so 添加到系统配置中；

(b)在 php/ext 目录下创建扩展头文件,取名为 imageAnalysis.def；

(c)使用 ext_skel 搭建扩展骨架;

(d)打开 config.m4 中的 enable 开关;

(e)重新配置 PHP；

(f)当扩展编译进去了之后，就可以开始修改扩展里的 c 文件，在里面可以添加 php->c 的转接函数，在转接函数里可以调用.so 内的函数；

(g)每次修改完上面的 c 文件，都要重新 make;

(h)重启 apache 服务器。

操作对应的流程图如图 4-10 所示。

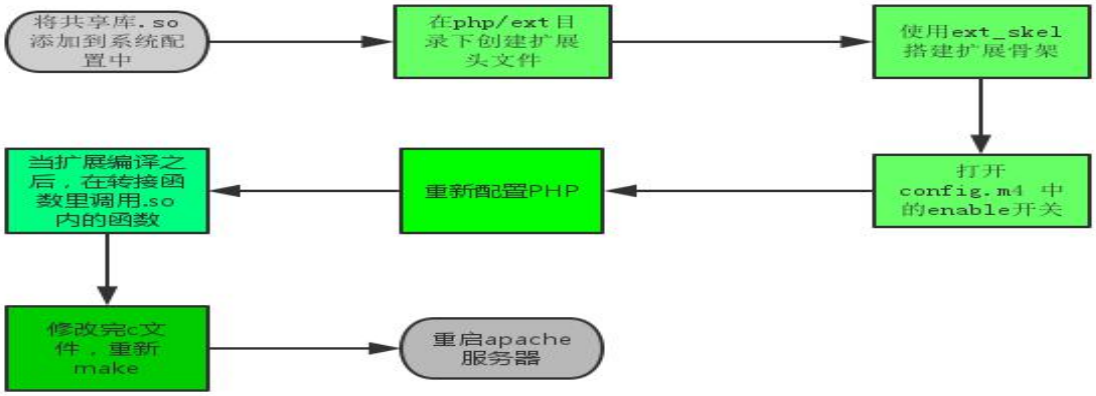

图 4-10 后台调用 Opencv 流程图

4.2.3 客户端部分

系统客户端有 iOS 和 Windows Phone 两个版本。其中两个版本的架构相似，只是相应的编程语言和调用的 API 不同。在 iOS 版本的客户端中，APP 分成网络层、视图控制器层、视图层、公共模块和一些第三方开源工具等部分来做介绍。

4.2.3.1 iOS 版本部分

iOS 版本的开发环境为 mac OS EI Capitan，开发编辑器为 XCode7.2，使用的编程语言为 Objective-C。

(1)网络层实现部分

在本系统中，网络层是基于对 AFNetworking 的进一步封装使之接口更加统一。在网络层主要有 get 和 post 两个方法，用于从后台获取数据和向后台发送数据。

网络层头文件定义如表 4-14 所示：

表 4-14 客户端网络层实现

```
@class UIImage;
  typedef NS_ENUM(NSInteger,DUDNetworkImageType) {
    DUDNetworkImageTypePNG,
    DUDNetworkImageTypeJPEG
};
@interface DUDNetwork : NSObject
// GET 网络请求(做登录验证和从后台获取数据使用)
+ (void)GetDataWithURL:(NSString *)str parameter:(NSDictionary *)dict success:(void(^)(id responseObject))successResponse error:(void(^)(NSError *error))errorResponse;
// POST 网路请求
```

```
+ (void)PostDataWithURL:(NSString *)str parameter:(NSDictionary *)dict image:(UIImage
*)image imageType:(DUDNetworkImageType)type uploadName:(NSString *)uploadName
progress:(void (^)(NSProgress *uploadProgress))uploadProgress success:(void (^)(id
response))successResponse error:(void (^)(NSError *error))errorResponse;
@end
```

头文件具体实现：

```
#import "DUDNetwork.h"
#import <AFNetworking.h>
@implementation DUDNetwork
+ (void)GetDataWithURL:(NSString *)str parameter:(NSDictionary *)dict success:(void(^)(id
responseObject))successResponse error:(void(^)(NSError *error))errorResponse{
    AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
     manager.responseSerializer = [AFHTTPResponseSerializer serializer];
    [manager GET:str parameters:dict progress:nil success:^(NSURLSessionTask *task, id
responseObject) {
        // 这里返回回来的数据不是 JSON 是 data, 但是不能直接转化成 JSON, 而先得转化成字符
串
     NSString *s = [[NSString alloc] initWithData:responseObject
encoding:NSUTF8StringEncoding];
        NSData *data = [s dataUsingEncoding:NSUTF8StringEncoding];
        NSDictionary *dic = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingMutableContainers error:nil];
        successResponse(dic);
    } failure:^(NSURLSessionTask *operation, NSError *error) {
        errorResponse(error);
        NSLog(@"\n\n\n\nError: %@", error);
    }];
}
+ (void)PostDataWithURL:(NSString *)str parameter:(NSDictionary *)dict image:(UIImage
*)image imageType:(DUDNetworkImageType)type uploadName:(NSString *)uploadName
progress:(void (^)(NSProgress *uploadProgress))uploadProgress success:(void (^)(id
response))successResponse error:(void (^)(NSError *error))errorResponse{
    //POST 表单上传图片
    AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
```

```
    manager.responseSerializer = [AFHTTPResponseSerializer serializer];

    [manager POST:str parameters:dict constructingBodyWithBlock:^(id<AFMultipartFormData>
_Nonnull formData) {
        // uploadfile 是后台定义的
        // image/jpeg 代表你上传的是图片
        NSData *imageData;
        if (type == DUDNetworkImageTypeJPEG) {
            imageData = UIImageJPEGRepresentation(image, 1);
            [formData appendPartWithFileData:imageData name:uploadName
fileName:@"imgfileStyle.png" mimeType:@"image/jpeg"];
        }else if (type == DUDNetworkImageTypePNG){
            imageData = UIImagePNGRepresentation(image);
            [formData appendPartWithFileData:imageData name:uploadName
fileName:@"imgfileStyle.png" mimeType:@"image/png"];
        }
    } progress:uploadProgress success:^(NSURLSessionDataTask * _Nonnull task, id
_Nullable responseObject) {
        // 这里返回回来的数据不是 JSON 是 data, 但是不能直接转化成 JSON, 而先得转化成字符
串
        NSString *s = [[NSString alloc] initWithData:responseObject
encoding:NSUTF8StringEncoding];
        NSLog(@"request result:%@", s);
        NSData *data = [s dataUsingEncoding:NSUTF8StringEncoding];
        NSDictionary *dic = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingMutableContainers error:nil];
        successResponse(dic);
    } failure:^(NSURLSessionDataTask * _Nullable task, NSError * _Nonnull error) {
        if (errorResponse) {
            errorResponse(error);
        }
        NSLog(@"Error:%@", error);
    }];
}
                                        @end
```

（2）视图控制器部分

在客户端中，视图控制器主要是为了处理业务逻辑，解决视图层和模型、网络数据的交流和协调的问题。由于在本系统的客户端中视图控制器较多，但是执行过程大致相同。大概过程为在视图控制器中的 ViewDidLoad 函数中执行视图中相关控件或是视图的加载、然后在 viewWillAppear、viewDidAppear、viewWillDisappear、viewDidDisappear 等生命周期函数中执行相应的操作，比如为各个控件增加相应的事件、控制各个控件之间的距离、布局、书写相关逻辑来控制和传输从用户传来的信息等。由于此部分代码较多且无太多技巧，遂在此不给出实例。

（3）视图部分

由于视图的编写跟视图控制器类似，在视图的 init 方法中初始化并添加控件，为这些控件添加相应事件来处理用户交互时的需求，然后对在视图上的控件进行布局。

（4）第三方开源库部分

在编写客户端相关代码时使用了一些知名、方便的第三方库来提升开发效率。这里对其分别简单介绍。

（a）AFNetworking。AFNetworking 是一个知名的网络库。其几乎覆盖了所有的 iOS 提供给开发者的网络请求功能，在 iOS 和 Mac OS 中均可运行。

表 4-15 的内容是一个示例，用于向后台请求数据。

表 4-15 AFNetworking 的使用

```
AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
    manager.responseSerializer = [AFHTTPResponseSerializer serializer];
   [manager GET:str parameters:dict progress:nil success:^(NSURLSessionTask *task, id
responseObject) {
      // 这里返回回来的数据不是 JSON 是 data, 但是不能直接转化成 JSON, 而先得转化成字符串
      NSString *s = [[NSString alloc] initWithData:responseObject
encoding:NSUTF8StringEncoding];
      NSData *data = [s dataUsingEncoding:NSUTF8StringEncoding];
      NSDictionary *dic = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingMutableContainers error:nil];
      successResponse(dic);
   } failure:^(NSURLSessionTask *operation, NSError *error) {
      errorResponse(error);
      NSLog(@"\n\n\n\nError: %@", error);
   }];
```

（b）Masonry。Masonry 是一个轻量级的布局框架，它提供了比苹果公司系统的自动布局语法更加友好和方便的语法。下表 4-16 是使用 Masonry 布局的一个核心方法示例。

表 4-16 Masonry 使用示例

```
[self.textLabel mas_makeConstraints:^(MASConstraintMaker *make) {
    make.left.equalTo(self.view.mas_left).offset(50);
    make.right.equalTo(self.view.mas_right).offset(-50);
    make.top.equalTo(self.view.mas_top).offset(100);
    make.height.equalTo(@50);
}];
```

（c）Realm。Realm 是一个轻量级的移动数据库，其具有方便、快捷、现代、移动优先等特点。下表 4-17--表 4-21 是使用 Realm 在移动端本身进行数据库存储的示例。

表 4-17 数据表的定义

```
@interface SPXAccountObject : RLMObject
@property NSString *accountId;
@property BOOL isFollow;
@property NSString *accountName;
@property NSString *avatarURL;
@property NSString *country;
+ (instancetype)getInstance;
- (void)addAccountCollectList:(SPXAccountModel*)account;
- (void)deleteAccountCollectList:(SPXAccountModel*)account;
- (NSArray *)selectAllCollectAccount;
- (id)selectAccountWithId:(NSString*)accountId;
@end
```

表 4-18 向数据库中增加一条记录

```
- (void)addObject:(RLMObject*)obj{
    // Add to Realm with transaction
    [self.realm beginWriteTransaction];
    [self.realm addObject:obj];
    [self.realm commitWriteTransaction];
}
```

表 4-19 向数据库中删除一条记录

```
- (void)deleteObject:(RLMObject*)obj{
    // Delete an object with a transaction
    RLMResults *results = [[obj class] allObjects];
    if (results.count == 0) {
        NSLog(@"There is no elements in the database");
        return;
    }
    // Delete an object with a transaction
    [self.realm beginWriteTransaction];
    [self.realm deleteObject:obj];
    [self.realm commitWriteTransaction];
}
```

表 4-20 更新数据库中的一条数据记录

```
- (void)updateWithObject:(RLMObject*)obj{
    [self.realm beginWriteTransaction];
    [self.realm addOrUpdateObject:obj];
    [self.realm commitWriteTransaction];
}
```

表 4-21 根据某一个字段来查询某条记录

```
- (id)queryObjectWithResult:(RLMResults *)results withPredict:(NSPredicate *)predicate
sortedBy:(NSString*)property ascending:(BOOL)isAscending{
    RLMResults *tempResults = [results objectsWithPredicate:predicate];
    tempResults = [tempResults sortedResultsUsingProperty:property ascending:isAscending];
    return tempResults;
}
```

（d）BaiduMapAPI。百度地图 iOS SDK 提供了展示地图的基本接口，还提供 POI 检索、路径规划、地图标注、离线地图、定位、周边雷达等丰富的 LBS 能力，由于代码太长故在此不做展示。

4.2.3.2 Windows Phone 版本部分

Windows Phone 版本的开发环境为 Windows 7，开发编辑器为 Visual Studio 2010，使用的编程语言为 C#。

Windows Phone 版本的第三方开发工具较少，而且在开发过程中较为仓促，虽然程序拥有 iOS 版本的相应功能，但是没有细化分层。此外在开发过程中也没有使用第三方插件。由于 Windows Phone 版本的开发过程同 iOS 版本类似，两者仅调用的系统 API、编程语言不同，故在此不再赘述，而仅展示 Windows Phone 版本的部分核心代码。代码如下表 4-22-表 4-23 所示。

表 4-22 Windows Phone 版本采集图像部分代码

```
namespace Healthmancer
{
    public class CameraCapture : IDisposable
    {
        MediaCapture mediaCapture;//捕?获?
        ImageEncodingProperties imgEncodingProperties;
        //MediaEncodingProfile videoEncodingProperties;
        public VideoDeviceController VideoDeviceController
        {
            get { return mediaCapture.VideoDeviceController; }
        }
        public async Task<MediaCapture> Initialize(CaptureUse primaryUse =
CaptureUse.Photo)
        {
            mediaCapture = new MediaCapture();


            ////////////////////////////////////////////////////////
            var devices = await DeviceInformation.FindAllAsync(DeviceClass.VideoCapture);
            var deviceInfo = devices[0]; //grab first result

            foreach (var device in devices)
            {
                if (device.Name.ToLowerInvariant().Contains("back"))
                {
                    deviceInfo = device;
                }
            }

            var mediaSettings = new MediaCaptureInitializationSettings
            {
                MediaCategory = MediaCategory.Communications,
                StreamingCaptureMode = StreamingCaptureMode.AudioAndVideo,
                VideoDeviceId = deviceInfo.Id
            };

//////////////////////////////////////////////////////////////////////////////
            await mediaCapture.InitializeAsync(mediaSettings);
```

```
            mediaCapture.SetPreviewRotation(VideoRotation.Clockwise90Degrees);

            //var ISOControl = mediaCapture.VideoDeviceController.IsoSpeedControl;
            //await ISOControl.SetValueAsync(100);

            //mediaCapture.VideoDeviceController.Focus.TrySetAuto(true);
           // mediaCapture.VideoDeviceController.FlashControl.

            var FocusControl = mediaCapture.VideoDeviceController.FocusControl;
            await FocusControl.SetValueAsync(130);
            //await FocusControl.FocusAsync();
            //FocusControl=AutoFocusRange.Normal;
            //await FocusControl.FocusAsync();
            //FocusControl.

            mediaCapture.VideoDeviceController.PrimaryUse = primaryUse;


            // Create photo encoding properties as JPEG and set the size that should be
used for photo capturing
            //imgEncodingProperties =
ImageEncodingProperties.CreateUncompressed(MediaPixelFormat.Bgra8);
            imgEncodingProperties = ImageEncodingProperties.CreateJpeg();
            //imgEncodingProperties.Width = 640;
            //imgEncodingProperties.Height = 480;

            return mediaCapture;
        }
        public async Task<StorageFile> CapturePhoto(string desiredName =
"livermancer.jpg")
        {
            // Create new unique file in the pictures library and capture photo into it

            //StorageFolder test = Windows.Storage.ApplicationData.Current.LocalFolder;
            //var localFolder = await test.CreateFolderAsync("test",
CreationCollisionOption.OpenIfExists);

            var photoStorageFile = await
ApplicationData.Current.LocalFolder.CreateFileAsync(desiredName,
CreationCollisionOption.GenerateUniqueName);
            await mediaCapture.CapturePhotoToStorageFileAsync(imgEncodingProperties,
photoStorageFile);

            return photoStorageFile;
        }
        public async Task StartPreview()
        {
            // Start Preview stream
            await mediaCapture.StartPreviewAsync();
        }
        public async Task StartPreview(IMediaExtension previewSink, double
```

```
desiredPreviewArea)
        {
            // List of supported video preview formats to be used by the default preview
format selector.
            var supportedVideoFormats = new List<string> { "nv12", "rgb32" };
            // Find the supported preview size that's closest to the desired size
            var availableMediaStreamProperties =

mediaCapture.VideoDeviceController.GetAvailableMediaStreamProperties(MediaStreamType.Vide
oPreview)
                    .OfType<VideoEncodingProperties>()
                    .Where(p => p != null && !String.IsNullOrEmpty(p.Subtype) &&
supportedVideoFormats.Contains(p.Subtype.ToLower()))
                    .OrderBy(p => Math.Abs(p.Height * p.Width - desiredPreviewArea))
                    .ToList();
            var previewFormat = availableMediaStreamProperties.FirstOrDefault();
            // Start Preview stream
            await
mediaCapture.VideoDeviceController.SetMediaStreamPropertiesAsync(MediaStreamType.VideoPre
view, previewFormat);
            await mediaCapture.StartPreviewToCustomSinkAsync(new MediaEncodingProfile
{ Video = previewFormat }, previewSink);
        }
        public async Task StopPreview()
        {
            await mediaCapture.StopPreviewAsync();
        }
        public void Dispose()
        {
            if (mediaCapture != null)
            {
                mediaCapture.Dispose();
                mediaCapture = null;
            }
        }
    }
}
```

表 4-23 Windows Phone 版本结果显示界面代码

```
public sealed partial class TestPage : Page
    {
        StreamSocketListener listener1;
        StreamSocketListener listener2;
        StreamSocketListener listener3;
        StreamSocketListener listenerResult;
        StreamSocket client;
        string[] One;
        string[] Two;
        string[] Three;
        string[] Four;
```

```
        string[] Five;
        string[] Six;
        string[] Seven;
        string[] Eight;
        string[] Nine;
        string[] Ten;
        string[] Eleven;
        public TestPage()
        {
            this.InitializeComponent();
            One = new string[4]{"-","10(0.56)+","20(1.14++)","40(2.28)+++"};
            Two = new string[5]{"-","±¨¤","+","++","+++"};
            Three = new string[6]{"-","5(0.5)","15(1.5)","40(4.0)","80(8.0)","160(16)"};
            Four = new string[5] { "-","15±¨¤", "70+","125++","500+++" };
            Five = new string[3] { "-","-", "+" };
            Six = new string[6] {"0.2(3.5)","1(17)", "2(35)", "4(70)", "8(140)",
"12(200)" };
            Seven = new string[6] {"-", "15(0.15)", "30(0.3)+", "100(1.0)++",
"300(3.0)+++", "2000(20)++++" };
            Eight = new string[4] {"-", "1(17)+", "2(35)++", "4(70)+++" };
            Nine = new string[6] { "-","100(5)", "250(15)", "500(30)", "1000(60)",
">2000(110)" };
            Ten = new string[7] { "-","1.005", "1.010", "1.015", "1.020", "1.025",
"1.030" };
            Eleven = new string[7] { "-","6.0", "6.5", "7.0", "7.5", "8.0", "9.0" };
        }

        /// <summary>
        /// Invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached.
        /// This parameter is typically used to configure the page.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }

        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            //progressbar1.Visibility = Visibility.Visible;
            //progressbar1.Value = 0;

            client = new StreamSocket();

            HostName server = null;
            server = new HostName("192.168.191.1");

            string connectError = "";
            try
            {
                await client.ConnectAsync(server, "748");
            }
```

```
            catch(Exception ex)
            {
                connectError = ex.Message;
            }
            if (connectError != "")
            {
                await new MessageDialog("No internet connection.").ShowAsync();
                return;
            }

            string connectError1 = "";
            try
            {
                listener1 = new StreamSocketListener();
                listener1.ConnectionReceived += OnConnection;
                await listener1.BindServiceNameAsync("111");

                listener2 = new StreamSocketListener();
                listener2.ConnectionReceived += OnConnection_two;
                await listener2.BindServiceNameAsync("112");

                listener3 = new StreamSocketListener();
                listener3.ConnectionReceived += OnConnection_three;
                await listener3.BindServiceNameAsync("114");

                listenerResult = new StreamSocketListener();
                listenerResult.ConnectionReceived += OnConnectionResult;
                await listenerResult.BindServiceNameAsync("113");

                string name1 = PhotoCapturePage.picName1;
                string name2 = PhotoCapturePage.picName2;
                string name3 = PhotoCapturePage.picName3;

                string request1 = "<protocol><file name=\"" + name1 + "\" mode=\"send\"
port=\"111\" /></protocol>";
                //string request2 = "<protocol><file name=\"" + name2 + "\" mode=\"send\"
port=\"112\" /></protocol>";
                //string request3 = "<protocol><file name=\"" + name3 + "\" mode=\"send\"
port=\"114\" /></protocol>";
                DataWriter writer = new DataWriter(client.OutputStream);
                writer.WriteString(request1);
                await writer.StoreAsync();

                //writer.WriteString(request2);
                //await writer.StoreAsync();

                //writer.WriteString(request3);
                //await writer.StoreAsync();

                writer.DetachStream();
                writer.Dispose();
```

```
            }
            catch(Exception ex1)
            {
                connectError1 = ex1.Message;
            }
            if(connectError1!="")
            {
                await new MessageDialog("Internet connection has lost.").ShowAsync();
                return;
            }

        }

        private async void OnConnection_three(StreamSocketListener sender,
StreamSocketListenerConnectionReceivedEventArgs args)
        {
            string error = "";
            try
            {
                StorageFile imageFile = await
ApplicationData.Current.LocalFolder.GetFileAsync(PhotoCapturePage.picName3);
                DataWriter writer = new DataWriter(args.Socket.OutputStream);
                Stream file = await imageFile.OpenStreamForReadAsync();
                byte[] buffer = new byte[1024];
                int bytesRead;
                int totalBytes = 0;
                do
                {
                    bytesRead = file.Read(buffer, 0, buffer.Length);
                    writer.WriteBytes(buffer);
                    totalBytes += bytesRead;
                    await writer.StoreAsync();
                    await writer.FlushAsync();
                } while (bytesRead > 0);

                writer.DetachStream();
                writer.Dispose();
                file.Dispose();
                args.Socket.Dispose();


                string request = "<protocol><file name=\"livermancer.jpg\"
mode=\"receive\" port=\"113\" /></protocol>";
                DataWriter writer1 = new DataWriter(client.OutputStream);
                writer1.WriteString(request);
                await writer1.StoreAsync();
                writer1.DetachStream();
                writer1.Dispose();


            }
```

```
            catch(Exception ex)
            {
                error = ex.Message;
            }
            if(error!="")
            {
                await new MessageDialog("Internet connection has lost.").ShowAsync();
                return;
            }
        }

        private async void OnConnection_two(StreamSocketListener sender,
StreamSocketListenerConnectionReceivedEventArgs args)
        {
            string error = "";
            try
            {
                StorageFile imageFile = await
ApplicationData.Current.LocalFolder.GetFileAsync(PhotoCapturePage.picName2);
                DataWriter writer = new DataWriter(args.Socket.OutputStream);
                Stream file = await imageFile.OpenStreamForReadAsync();
                byte[] buffer = new byte[1024];
                int bytesRead;
                int totalBytes = 0;
                do
                {
                    bytesRead = file.Read(buffer, 0, buffer.Length);
                    writer.WriteBytes(buffer);
                    totalBytes += bytesRead;
                    await writer.StoreAsync();
                    await writer.FlushAsync();
                } while (bytesRead > 0);

                writer.DetachStream();
                writer.Dispose();
                file.Dispose();
                //listener.Dispose();
                args.Socket.Dispose();
                //listener.Dispose();
                //client.Dispose();
                /*string request = "<protocol><file name=\"livermancer.jpg\"
mode=\"receive\" port=\"113\" /></protocol>";
                DataWriter writer1 = new DataWriter(client.OutputStream);
                writer1.WriteString(request);
                await writer1.StoreAsync();
                writer1.DetachStream();
                writer1.Dispose();*/

            }
            catch(Exception ex)
            {
```

```
                error = ex.Message;
            }
            if(error!="")
            {
                await new MessageDialog("Internet connection has lost.").ShowAsync();
                return;
            }
        }

        private async void OnConnection(StreamSocketListener sender,
StreamSocketListenerConnectionReceivedEventArgs args)
        {
            string error = "";
            try
            {
                StorageFile imageFile = await
ApplicationData.Current.LocalFolder.GetFileAsync(PhotoCapturePage.picName1);
                DataWriter writer = new DataWriter(args.Socket.OutputStream);
                Stream file = await imageFile.OpenStreamForReadAsync();
                byte[] buffer = new byte[1024];
                int bytesRead;
                int totalBytes = 0;
                do
                {
                    bytesRead = file.Read(buffer, 0, buffer.Length);
                    writer.WriteBytes(buffer);
                    totalBytes += bytesRead;
                    await writer.StoreAsync();
                    await writer.FlushAsync();
                } while (bytesRead > 0);

                writer.DetachStream();
                writer.Dispose();
                file.Dispose();
                args.Socket.Dispose();

                string request = "<protocol><file name=\"livermancer.jpg\"
mode=\"receive\" port=\"113\" /></protocol>";
                DataWriter writer1 = new DataWriter(client.OutputStream);
                writer1.WriteString(request);
                await writer1.StoreAsync();
                writer1.DetachStream();
                writer1.Dispose();
                //client.Dispose();
            }
            catch(Exception ex)
            {
                error = ex.Message;
            }
            if(error!="")
            {
```

```
                await new MessageDialog("Internet connection has lost.").ShowAsync();
                return;
            }
        }

        private async void OnConnectionResult(StreamSocketListener sender,
StreamSocketListenerConnectionReceivedEventArgs args)
        {
            string error = "";
            DataReader reader = new DataReader(args.Socket.InputStream);
            try
            {
                await reader.LoadAsync(11);
                string msg = reader.ReadString(11);
                int[] result = new int[11] { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
                char[] temp = new char[11];
                temp = msg.ToCharArray();
                for (int i = 0; i < 11; i++)
                {
                    result[i] = temp[i] - '1';
                }
                await this.Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () => {
                    //progressbar1.Visibility = Visibility.Collapsed;
                    one.Header = One[result[0]];
                    two.Header = Two[result[1]];
                    three.Header = Three[result[2]];
                    four.Header = Four[result[3]];
                    five.Header = Five[result[4]];
                    six.Header = Six[result[5]];
                    seven.Header = Seven[result[6]];
                    eight.Header = Eight[result[7]];
                    nine.Header = Nine[result[8]];
                    ten.Header = Ten[result[9]];
                    eleven.Header = Eleven[result[10]];
                });
            }
            catch (Exception ex)
            {
                error = ex.Message;
            }
            reader.DetachStream();
            reader.Dispose();
            listenerResult.Dispose();
            args.Socket.Dispose();
            client.Dispose();
            if(error!="") {
                await new MessageDialog("Internet connection has lost.").ShowAsync();
                return;
            }
        }
}
```

5. 尿常规检测系统的验证分析与性能测试

5.1 系统运行界面

5.1.1 检测系统的后台启动与运行

本系统的后台系统环境为 XAMPP 集成环境，在使用尿常规检测系统提供的服务之前应该先启动后台服务器待其启动成功之后再进行后续操作。

检验服务器是否成功启动的操作是在本地浏览器中输入 localhost，会出现如下图 5-1 所示的界面，代表服务器和数据库正常启动。

图 5-1 XAMPP 启动界面

在本文中实现了两个平台的客户端——iOS 平台和 Windows Phone 平台，由于时间关系，两个客户端所实现的功能不完全一致。

iOS 客户端实现了用户登录和注册、存储用户个人信息、收藏健康贴士、采集用户反应试纸的图像数据，将采集的图像数据上传并从后台获得图像分析结果、将检测结果按照用户 ID 和检测时间存储于后台和客户端、系统使用说明、从后台更新健康贴士、根据定位查看周边的医院并提供从用户所在位置到选择的目标医院的公交、骑行、驾乘和步行四种路径导航方式等功能。

而 Windows Phone 版本客户端功能较 iOS 版本少，但是仍然保有系统使用说明、采集用户反应试纸图像信息、从后台获取试纸图像分析结果并显示、用户注册和登录等核心功能。

下面根据客户端操作系统的不同，针对各个不同的功能点分别展示对应的界面。

5.1.2 iOS 版本界面

a) APP 欢迎界面

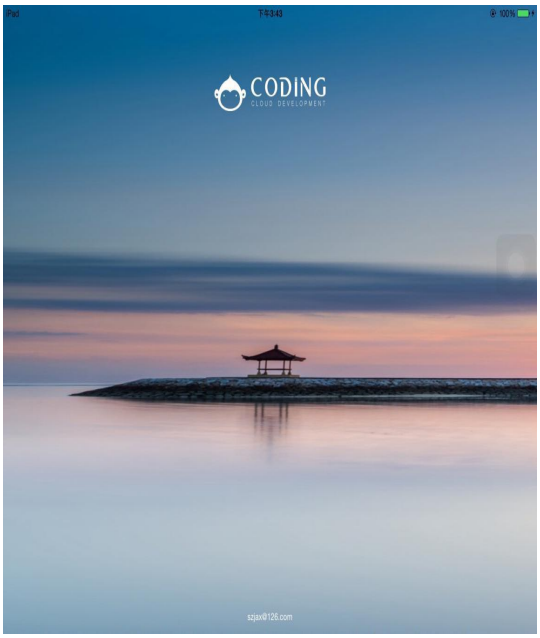

图 5-2 APP 欢迎界面

b) 系统操作使用说明

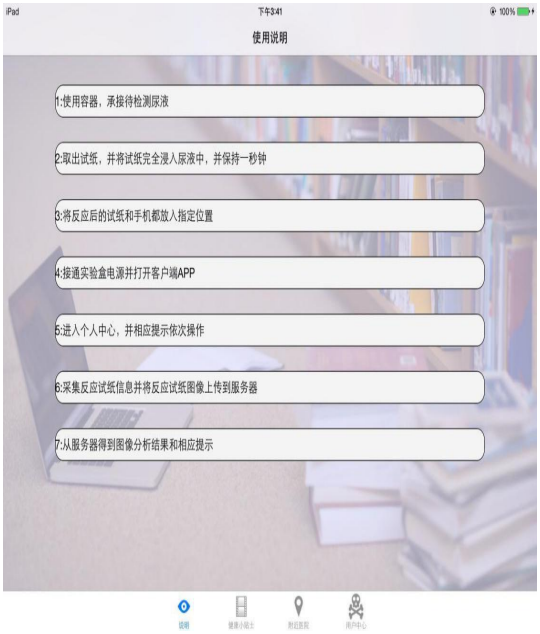

图 5-3 使用说明界面

c) 基于 GPS 搜索附近的医院

图 5-4 搜索附近医院界面

d) 搜索公交导航信息

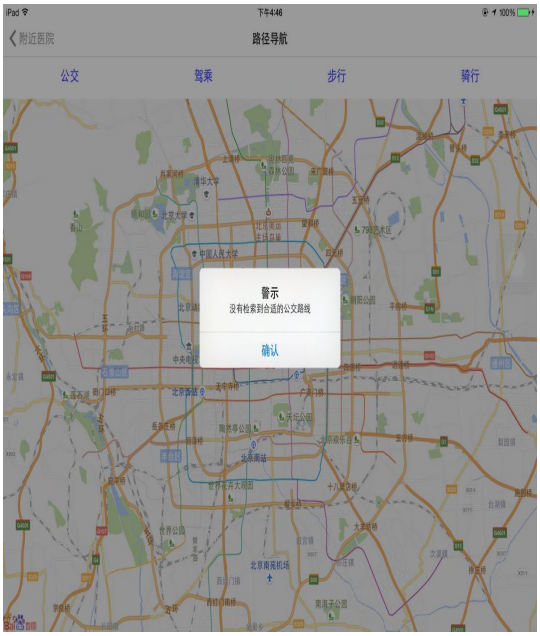

图 5-5 公交搜索界面

e) 搜索驾乘导航信息

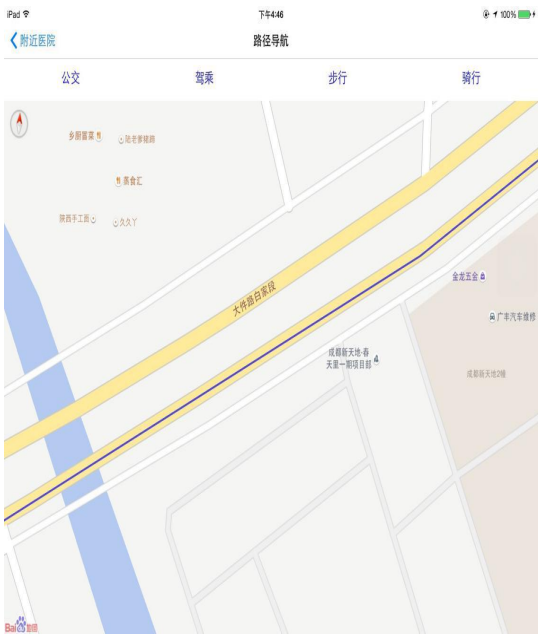

图 5-6　驾乘搜索界面

f) 搜索骑行或步行导航信息

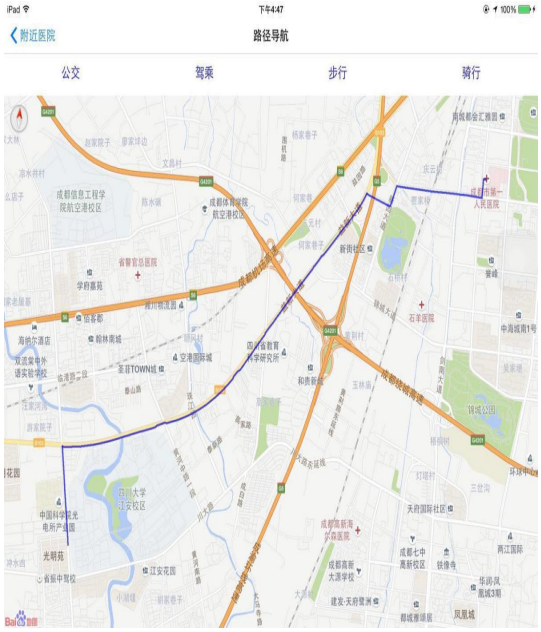

图 5-7　骑行/步行搜索界面

g)用户登录界面

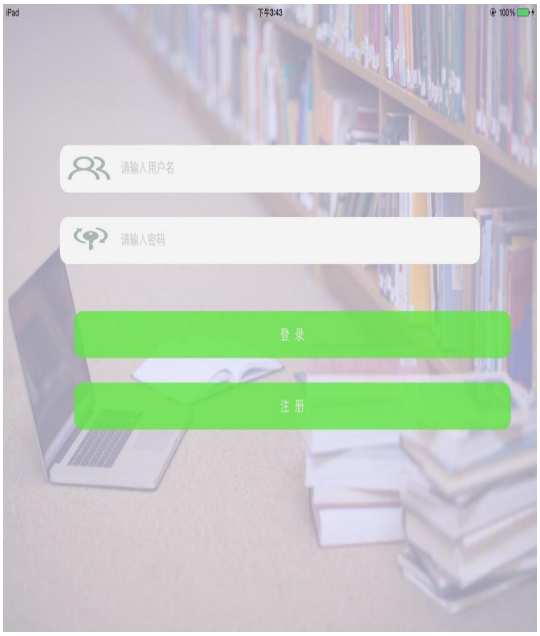

图 5-7　登录界面

h)用户注册界面(未填写信息)

图 5-8 注册界面(未填写信息)

i)用户注册界面(已填写信息)

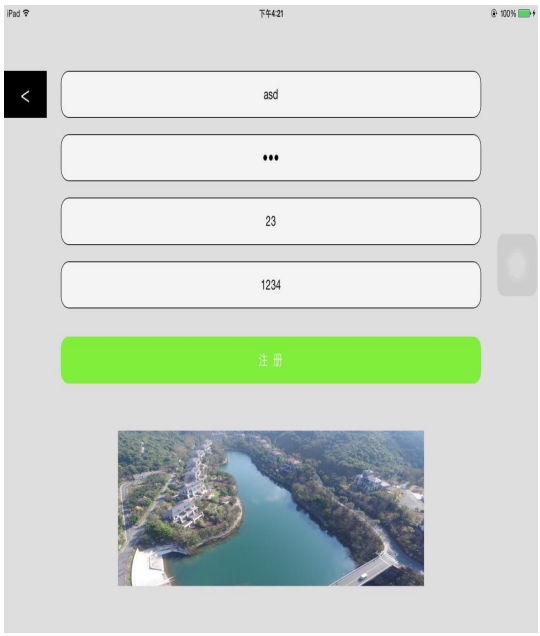

图 5-9 注册信息已填界面

j)用户个人主页界面

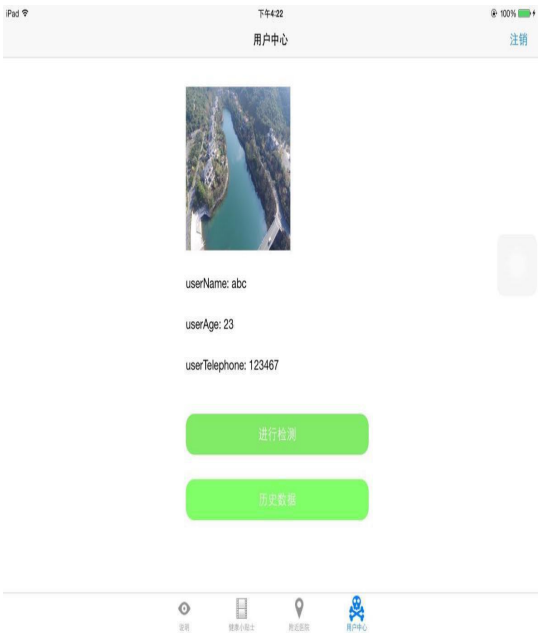

图 5-10 用户中心界面

k)采集图像界面

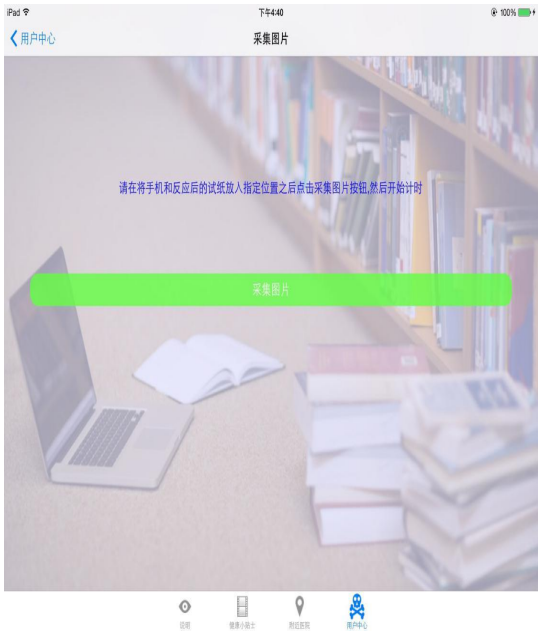

图 5-11 图像采集界面

l)确认图像上传界面

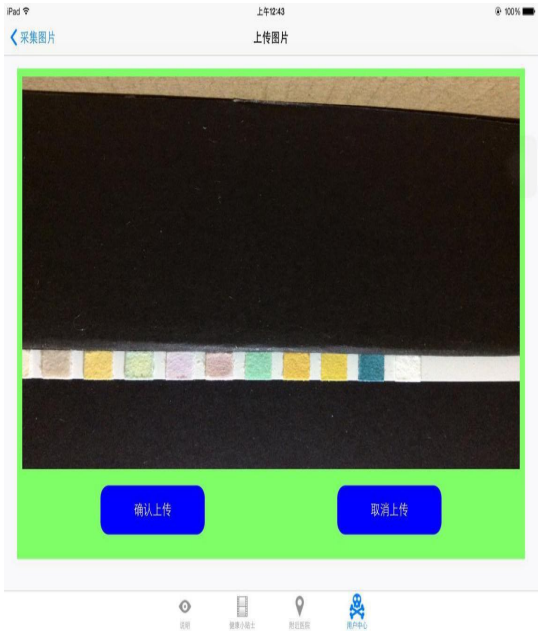

图 5-12 图像确认上传界面

m)图像正在上传界面(显示上传百分比)

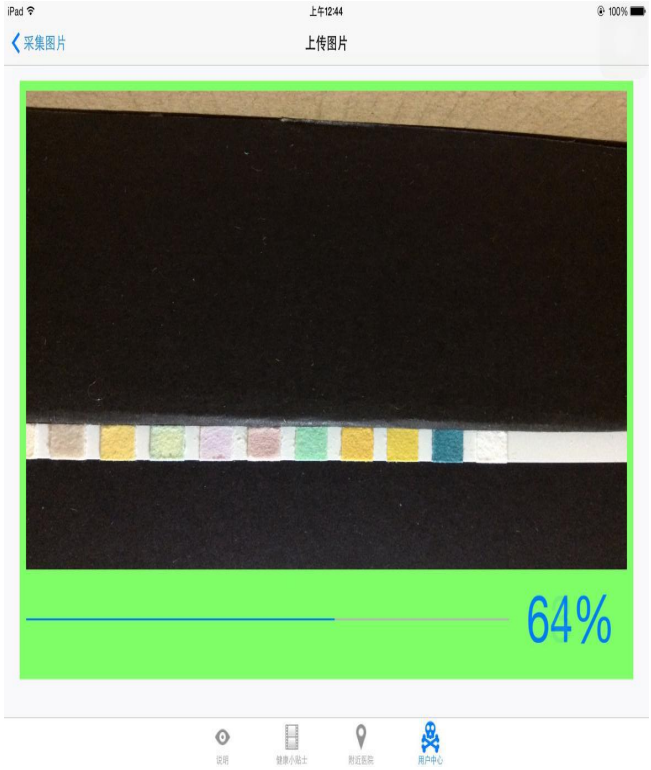

图 5-13 正在上传界面

n)查看检测历史列表界面

图 5-14 历史检测数据首页界面

o)显示分析结果界面

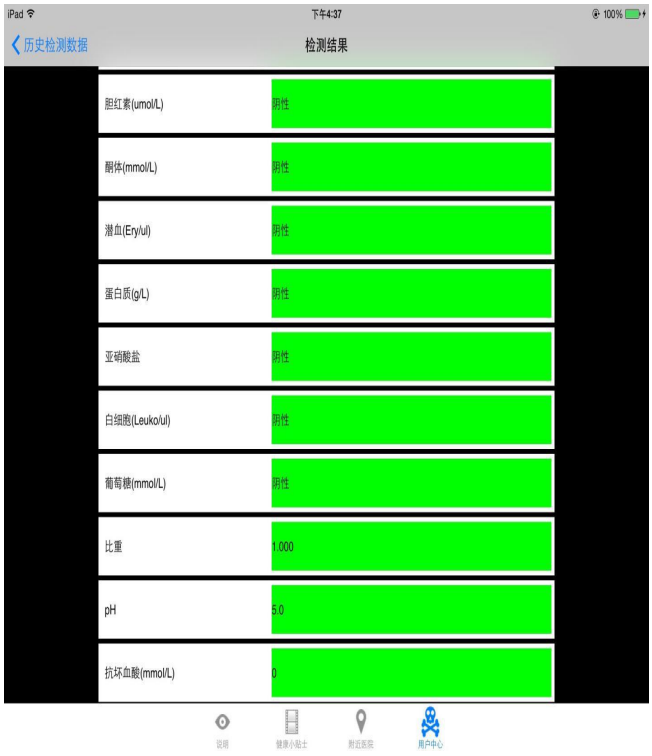

图 5-15 检测结果界面

5.1.3 Windows Phone 版本界面

由于 Windows Phone 版本是把所有功能全都糅合到一个大的界面中，所以在此不做功能分开的展示，而是将所有界面截图统一展示。如下图 5-16 到图 5-23 所示。

图 5-16

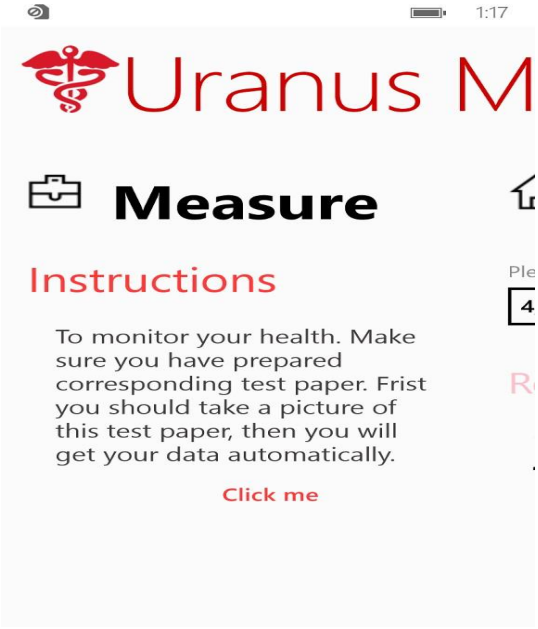

图 5-17

图 5-18

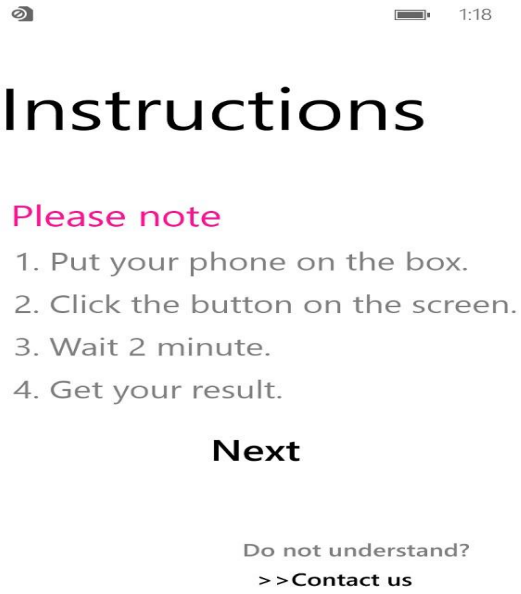

图 5-19

图 5-20

图 5-21

图 5-22

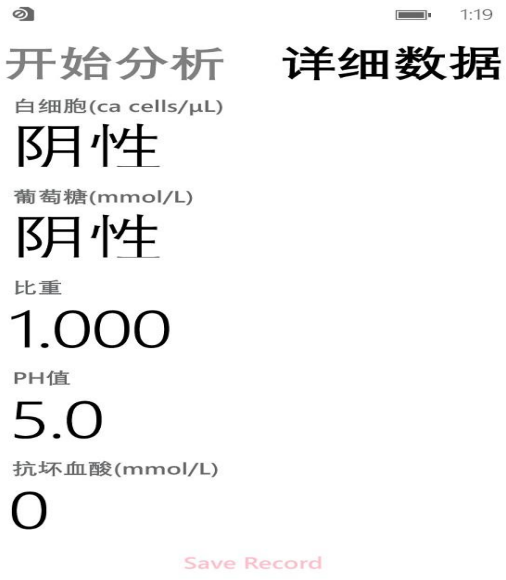

图 5-23

5.2 尿常规检测系统的验证和测试

5.2.1 实验方法

本文设计并实现的尿常规检测系统主要由实体和软件两部分组成。实体部分主要是为了提供稳定的物理环境，从而进一步地减少误差和提升检测精度同时还能方便用户操作。而软件部分功能较多，已经在前文叙述。

在使用本尿常规检测系统检测尿液时所用到的十一项检测试纸如图 5-23 所示，与其对应的比色卡如图 5-24 所示。尿常规的检测步骤如下：

（1）使用容器承接待测尿液；

（2）取出检测试纸，并将试纸完全进入尿液中，并保持一秒钟；

（3）待试纸从尿液中取出后使用滤纸滤去多余尿液；

（4）将反应后的试纸和手机都放入指定位置；

（5）接通实验盒电源并打开客户端 APP；

（6）点击采集图像按钮，等待系统自动采集图像数据并从后台获取分析数据；

（7）得到该次检测结果并存入历史数据中。

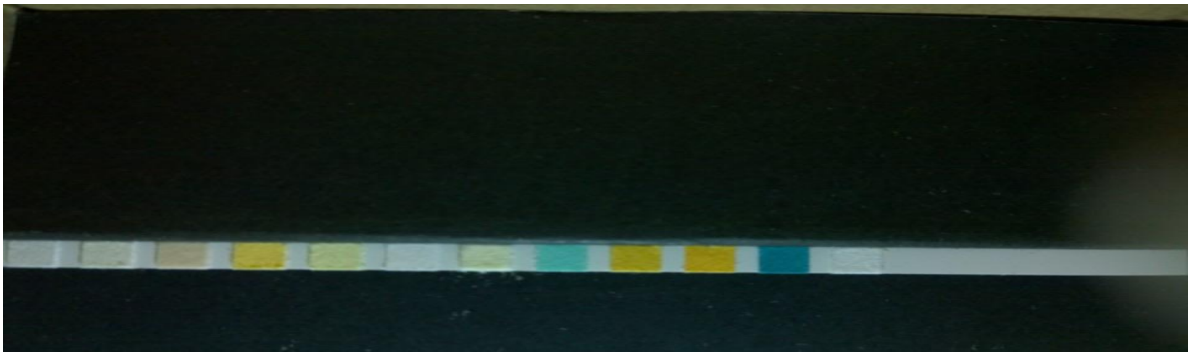

图 5-23 反应试纸图像

试纸对应的比色卡图像如下图 5-24 所示

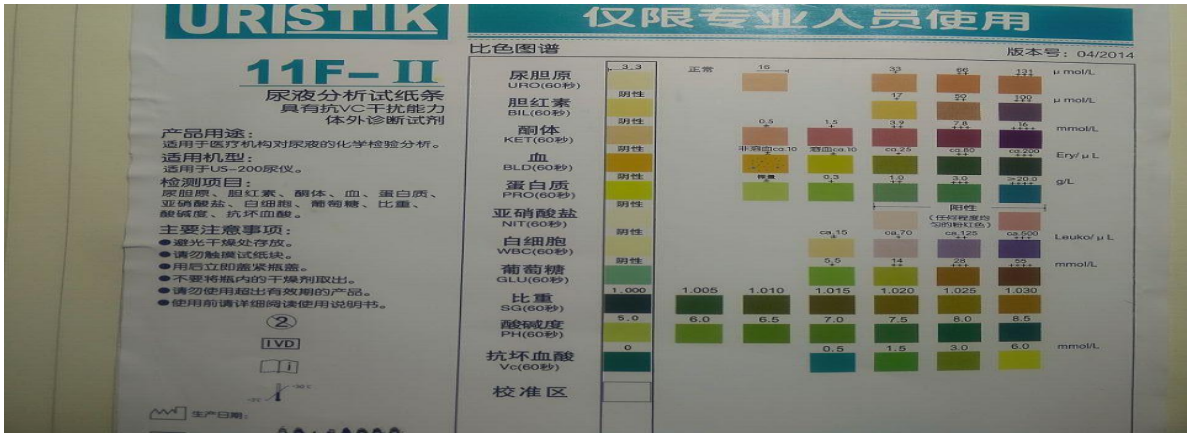

图 5-24 比色卡图像

反应后的试纸图像使用高斯滤波预处理后的试纸图像为下图 5-25 所示。

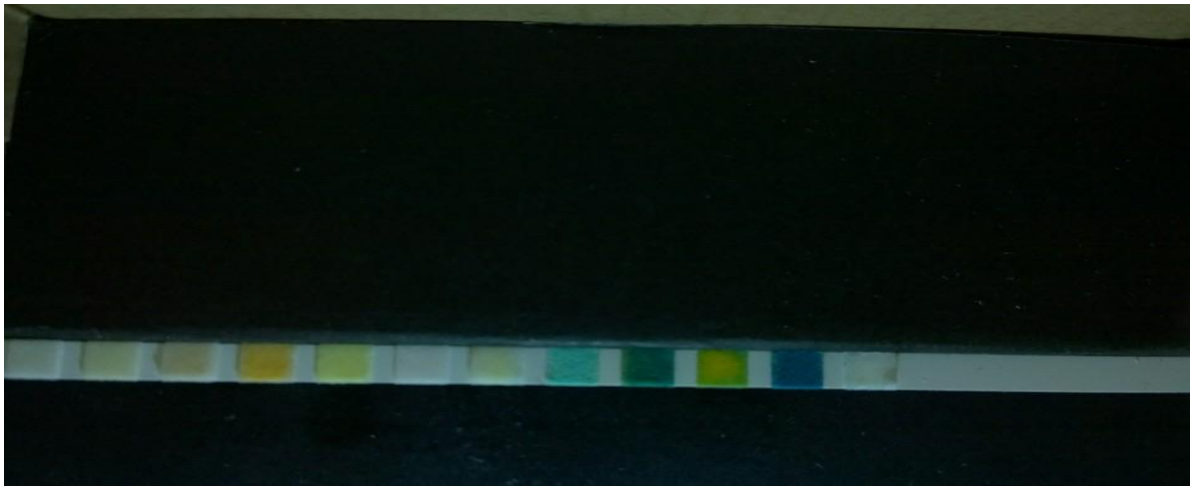

图 5-25 高斯滤波之后图像

接着下面是使用 Canny 算子来进行边缘检测后分割出的试纸条上的反应模块，其效果图如下图 5-26 所示。

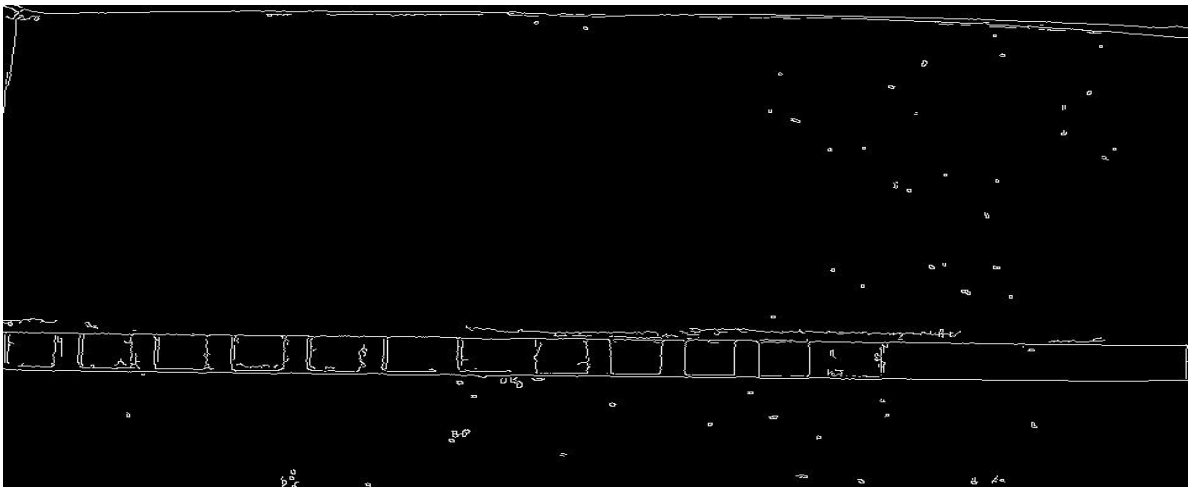

图 5-26 Canny 算子分割后图像

5.2.2 实验数据

为了验证本文设计的基于图像处理的尿常规检测系统的有效性，采用了使用传统尿常规检测仪器和本尿常规检测系统两种仪器对同一份待测样本进行检测然后对两份结果进行对比的办法。下面五次实验的医院方法所得的数据均为使用医院里正在使用的尿常规检测仪器检测的数据。在实验中，本文设计的尿常规检测系统所使用的相关参数如下所示。

实验共进行了五次，结果如下表 5-1 到表 5-5 所示。

表 5-1 实验结果对比(第一次)

检测项目	本文方法	医院方法
尿胆原	正常	正常
胆红素	阴性	阴性
酮体	阴性	阴性
潜血	阴性	阴性
蛋白质	微量	阴性
亚硝酸盐	阴性	阴性
白细胞	阴性	阴性
葡萄糖	阴性	阴性
比重	1.020	1.015
酸碱度	6.5	6.5
抗坏血酸	0	0.5

表 5-9 实验结果对比(第二次)

检测项目	本文方法	医院方法
尿胆原	正常	正常
胆红素	阴性	阴性
酮体	阴性	阴性
潜血	阴性	阴性
蛋白质	阴性	微量
亚硝酸盐	阴性	阴性
白细胞	阴性	阴性
葡萄糖	阴性	阴性
比重	1.025	1.020
酸碱度	7.0	7.5
抗坏血酸	0.5	0

表 5-10 实验结果对比(第三次)

检测项目	本文方法	医院方法
尿胆原	3.3~16umol/L	正常
胆红素	阴性	阴性
酮体	阴性	阴性
潜血	阴性	阴性
蛋白质	微量	微量
亚硝酸盐	阴性	阴性
白细胞	阴性	阴性
葡萄糖	阴性	阴性
比重	1.015	1.020
酸碱度	6.5	7.0
抗坏血酸	0	0

表 5-11 实验结果对比(第四次)

检测项目	本文方法	医院方法
尿胆原	正常	正常
胆红素	阴性	阴性
酮体	阴性	阴性
潜血	阴性	阴性
蛋白质	阴性	阴性
亚硝酸盐	阴性	阴性
白细胞	阴性	阴性
葡萄糖	+	阴性
比重	1.015	1.015
酸碱度	6.5	6.0
抗坏血酸	0	0

表 5-12 实验结果对比(第五次)

检测项目	本文方法	医院方法
尿胆原	正常	正常
胆红素	阴性	阴性
酮体	阴性	阴性
潜血	阴性	阴性
蛋白质	阴性	阴性
亚硝酸盐	阴性	阴性
白细胞	阴性	阴性
葡萄糖	阴性	阴性
比重	1.020	1.025
酸碱度	6.5	6.0
抗坏血酸	0	0

5.2.3 实验结果说明

由上述五张表所展示的数据内容显示，本尿常规检测系统在五次尿液检测中检测较为准确，特别是尿胆原、胆红素、酮体、潜血、蛋白质、亚硝酸盐白细胞等指标。这说明本文设计并实现的尿常规检测系统在一定程度上具有有效性。不过对另外一些指标，如 pH 值、抗坏血酸，系统检测的精度相较不是太高，这可能是因为比色板上的颜色信息跟试纸反应后的真实颜色有偏差，进而造成误差。而其他指标大多是阴性，比色板颜色同试纸颜色相差较小，因此误差较小。不过这种误差在使用正确配置的待测物溶液滴定并重新制作样本比色卡之后可以进一步消除。

5.3 尿常规检测系统误差分析

如前文所述，在检测正常用户的尿液待测物方面，基于图像分析的尿常规检测分析系统的检测准确度已经较高。但是本文设计的系统的检测精度平均值仍只有 80%左右，并且在某些情况的某些指标上准确度较低，造成这种结果的可能原因如下：

(1)实验操作引起的误差：

由于在实验时很多都是手动操作，如放置反应后的试纸和手机可能造成的移位，又如将尿常规检测试纸浸入待测样本中的浸入时间等等操作都会引入误差。

(2)图像采集设备造成的误差：

由于本系统的光学成像设备是中高端智能手机的后置摄像头，每个厂商都会根据某种需求(如缩小图像体积)在硬件上对所采集的图像信息进行加工处理，从而造成某一部分颜色信息失真进而导致误差的产生。

(3)外界环境对化学反应的影响：

虽然反应是在一个固定的实验盒中进行，但是其仍然可能受到周围自然环境的影响。如颜色反应是在空气中发生的，可能空气中的某些物质会和试纸中的检测物质发生反应，进而产生干扰。

(4)反应试纸引入的错误：

由于在普通医院中，尿常规检测就是一项半定量的生化检测，其检测精度本身就没有其它的定量检测那么高，所以相关厂商也没有对反应试纸上涂的化学反应物质有严格的测量，造成不同的试纸测同一份样本会有不同的结果的现象。

(5)测量次数的影响：

由于时间和实验条件有限，在实验中每种检测样本均只检测了十次。测试样本数不够，可能会引入随机误差，每次检测结果对准确度的影响都会比较大。所以应该尽可能多地进行实验，以便减少某次特别的误差对整个系统准确度的影响。

(6)图像处理算法选择引入的误差：

本系统是基于彩色图像处理算法而构造的，图像处理算法是整个检测系统的重中之重，因此本文所使用的图像处理算法会引入相应误差。虽然本文中设计的算法是根据拍摄的图像数据的特点选定的，能有效地进行降低此处的误差，但是由于算法本身具有的缺陷，如边缘检测模块对噪声比较敏感，无法保证区域内部颜色特征的一致性，而且不能产生连续闭合的轮廓，仍然会引入一定误差。

(7)蚁群算法产生聚类中心引入的误差：

蚁群算法是一种经典的进化算法，其最后的运行效果高度依赖最初设定的参数。由于在设计并实现本系统时，对进化算法方面并无太多研究，设置并最终得到的初始值可能效果并不是最为理想。而 FCM 亦为初始条件极为敏感的算法，因此相关参数的错误设置可能会为后面模糊聚类带来负面影响。

(8)实验所使用的光源可能引入的误差：

由于葡萄糖、蛋白质等待检测的生化指标的频率分布特性都不一样，所以各种物质对不同波长和不同频率的波的敏感程度也不一样，所以为了得到最好的检测效果，应该按照各种物质已经测得的频率分布特性来选择不同波长范围的光源，如葡萄糖的理想光源即为波长在 520-560nm 范围内的 LED。

5.4 尿常规检测系统的不足之处

如前文所述，尿常规检测分析仪正在向着高精度、高速度、小型化和低成本的方向发展，新的技术和新的器件将使干式生化分析技术具有非常好的应用前景。在本文中通过引入图像处理技术来降低仪器成本、增加仪器的便携性，但是从实验结果来看本文设计并实现的基于图像处理的干式生化分析技术的检测精度仍然不是特别理想，还存在着需要改进的地方。

（1）在图像处理方面，虽然本文使用改进了的模糊 C 均值算法，提高了模块的分割精度，减少了噪声干扰。但是随着数据量的增大，改进的 FCM 算法的运算速度也会相应的降低，进而降低尿常规检测的速度。

（2）在蚁群算法相关初始参数的训练并优化方面的工作不足。因为时间和经验的约束，在实现系统时，只能依据作者的常识和相关背景知识，并仅利用有限次数实验所得的图像数据分析得到算法最终的初始参数。而所得的参数在本系统中并不是最优的。

（3）在本系统中使用欧式距离来衡量颜色之间的差异，在某些情况下会有误差存在，因此在后续研究中将选择误差更小的颜色比较方法，定义更适合系统本身的特征距离。

（4）交互性不佳。在 Web 2.0 时代，APP 的设计原则之一就是交互性好，虽然本系统主要是对用户待测样本进行检测，交互需求不是很多，但是这仍然是本系统目前存在的一个不足之处，以后需要改进。

（5）操作不方便。本文设计的检测系统采用了多处控制用户的选择的方式来减少系统产生的误差，如固定反应试纸和手机的放置位置，固定光源的位置和波长、强度等因素，同时让反应在一个较为封闭的环境下进行。但是这样做会造成在现有条件下用户的操作不便。如用户不能随意放置待测试纸和手机，必须得小心翼翼的，用户体验还有待提高。

6．总结

慢性肾脏病是一种较为常见的危害性较大的慢性疾病。尿液常规检查是及早发现和诊断慢性肾病的一种重要的诊疗手段。然而，传统尿液检测存在的各种不便导致人们难以快捷方便的进行尿液检查。针对此问题，本文设计和实现了一种基于图像处理的尿液检测系统。

论文首先分析了现在慢性肾病在全世界的发病情况和人们及时发现和诊断自己的身体健康状态遇到的困难，简述了对新型尿液检测系统的需求。然后，对尿常规检测系统所使用的相关技术问题进行了介绍。最后，给出了基于图像处理技术的尿常规检测系统的详细设计和实验分析，并给出了相应的结论。

由于时间关系，本文设计和实现的尿液检测系统尚未达到实用的标准要求，还存在一些不足之处。例如，在数据量较大时运算速度较慢、颜色比较方法可能不够先进、交互性不够好、操作不够方便等等。此外，系统界面等也较为简陋。后期，可以在本文内容基础上进一步完善其功能，使其可以达到商业化实用产品的要求。

通过本系统的开发，让我对诸多方面有了一个更深的认识。从 iOS 客户端和 Windows Phone 客户端的开发到图像处理算法和 Opencv 开源视觉库，再到怎么制作实验盒，怎么焊接电路，怎么同人打交道等等方面都有了较大的提高。在实际开发过程中，学到了很多主流开发技术：iOS API、Windows Phone API、Opencv 视觉库、PHP 后台脚本、Linux 指令和 ThinkPHP 开源框架等。在开发过程中也意识到了自己的不足之处，比如在设计模式、架构设计、代码简洁性、高效性等方面可能做得还不够好。在今后的开发中定会以此开发经历为鉴，百尺竿头更进一步。

参考文献

[1]张萍. 基于彩色图像处理的干式生化分析技术的研究与应用[D]. 吉林：吉林大学电子科学与工程学院，2012.

[2]National Kidney Foundation.K/DOQI clinical practice guidelines for chronic kidney disease [EB/OL]. https://www.kidney.org/professionals/KDOQI/guidelines_ckd.2002

[3]KDIGO: Kidney Disease Improving Global Outcomes.KDIGO Clinical Practice Guideline for the Diagnosis, Evaluation, Prevention, and Treatment of Chronic Kidney Disease-Mineral and Bone Disorder (CKD-MBD) [EB/OL] . http://www.kdigo.org/clinical_practice_guidelines/pdf/CKD/KDIGO%20CKD-MBD%20GL%20KI%20Suppl%20113.pdf . August 2009

[4]A Martínez-Castelao, JL. Górriz, J Bover; et al. Consensus document for the detection and management of chronic kidney disease [DB/OL].http://www.revistanefrologia.com/es-linkresolver-X0211699514053919. Nefrologia 34 (2): 243－62. doi:10.3265/Nefrologia . 2014.02

[5]KDIGO(Kidney Disease Improving Global Outcomes).KDIGO 2012 Clinical Practice Guideline for the Evaluation and Management of Chronic Kidney Disease. http://www.kdigo.org/clinical_practice_guidelines/pdf/CKD/KDIGO_2012_CKD_GL.pdf.Kidney Int Suppl 3 (1): 1－150. Retrieved 5 February 2016.

[6]闫靖波.基于 Opencv 的图像分割与目标跟踪算法的设计与实现[D].东北大学信息科学与工程学院，2011.

[7]蔡涛，徐国华，徐筱龙 . 基于模糊 C 均值与 Markov 随机场的图像分割 [J]. 计算机工程，2007，33(20)：36-39.

[8]Coloni A,Dorigo M , Maniezzo V.Ant system :Optimization by a colony of cooperating agent[J] .IEEE Trans on Systems, Man and Cybernetics-Part B :Cybernetics.1996 , 26(1):29 ～ 41.

[9]詹士昌,徐婕,吴俊.蚁群算法中有关算法参数的最优选择[J].科技通报.2003(05).

[10]杨欣斌,孙京诰,黄道.基于蚁群聚类算法的离群挖掘方法[J].计算机工程与应用.2003(09).

[11]OSChina 开源社区 .opencv.http://www.oschina.net/p/opencv.

[12]ThinkPHP 作者 .ThinkPHP3.2 完全开发手册 [EB/OL] . http://document.thinkphp.cn/manual_3_2.html.

[13]Github.AFNetworking.http://afnetworking.com.

[14]Github.Masonry.https://github.com/SnapKit/Masonry.

[15]Github.Realm.https://realm.io/.

[16] 百 度 . 百 度 地 图 iOS 地 图 SDK 概 述.http://lbsyun.baidu.com/index.php?title=iossdk

[17]Pal Nikhil R,Bezdek Janes C.On cluster validity for the fuzzy c-means model[J].IEEE Transactions on Fuzzy Systems,1995,3(3):370-379.

[18]张萍,王剑钢.结合空间信息的 FCM 聚类噪声图像分割方法[J].计算机与现代化.2012(03)

声　明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名）_____

论文指导教师（签名）_____

2016 年 05 月 30 日

致 谢

论文即将完成，值此论文完成之际，我首先要特别感谢我的毕业论文指导导师陈瑜老师。陈老师在我完成毕业设计的路上一直支持我，鼓励我并在毕业设计完成期间提出了很多建设性意见，这使我受益匪浅。正是因为陈老师对我的精心指导，我的毕业设计才能顺利完成，在此特别向陈老师表示衷心地感谢。

在整个论文的完成过程中，我遇到了一系列障碍，从论文选题，到程序实现到对尿常规检测系统相关技术的论证和分析，我都遇到了各种意想不到的问题。在遇到种种难解问题的时候，我都能得到及时的指引和帮助。可以说这一路上如果没有帮助过我的老师、同学对我的支持，我可能走不到现在。正是你们对我的支持、鼓励和指导，我才能走到现在。

同时感谢大学四年里的老师们，你们对我四年的悉心指导和教育才让我在这四年时间里进步飞快。你们不仅教会我各种各样的技术知识，还教导我“欲做事先做人”的道理。正是你们这四年里的指导和教育让我在学识、人格方面有了更深的沉淀，使我今后在走上社会后有知识和技能帮助我向目标奋斗。

同样感谢陪伴我度过大学四年生活的同学和朋友们，我十分真诚地感谢你们这四年里在生活和学习方面给予我的帮助和支持。

最后我还要感谢我的家人，他们在生活和精神上给我提供了莫大的支持，使我能全身心地投入到学业中，为我今后的成长和成才奠定扎实的基础。

在此由衷地感谢所有给予我关心和帮助的人！