目　录

摘　要

本系统目的是分析并读取出 Jpeg 格式照片中的额外信息，如照片的拍摄信息和 GPS 信息，这些信息可以在照片中的 Exif 格式里面分析并提取出来。之后通过以文件夹的形式来展现分类后的照片。在分类的过程中使用了多线程的技术来提升分类的速度。同时对 Exif 格式有了自己的见解。

本论文结合市面上的照片管理程序和系统背景，描述了一个使用.Net 4.7 平台下的 C#开发语言和使用 WPF 框架设计 UI 界面的系统。实现了本次选题的目的。

关键词：照片，C#，WPF

作　　者：徐福扬

指导老师：姚望舒

Abstract

The purpose of this system is to analyze and read additional information in Jpeg format photos, such as photo capture information and GPS information, which can be analyzed and extracted in the Exif format of the photo. The classified photos are then presented in the form of a folder. Multi-threaded techniques are used in the classification process to speed up the classification. At the same time,otherwise i have my own opinion on the Exif format.

This paper combines the photo management program and system background on the market to describe a system that uses the C# development language under the .Net 4.7 platform and the UI interface using the WPF framework. The purpose of this topic was achieved.

Keywords: Photo, C#, WPF

Written by Fuyang Xfu
Supervised by Wangshu Yao

前　言

伴随着现代科技的高速发展，手机扮演了在现在生活中一种无可替代的角色，与此同时手机的像素比上个世纪得到了一个巨大的提升，虽然目前还是不能和专业的相机相比。但是手机相机拥有着的比专业的相机的便利性。人们可以享受手机其他功能的同时，也用手机的拍照功能。人们几乎时时刻刻都可以照相，但是随着照片的增多，照片的管理便成为了一个难题。不能够精确的查找某张照片或者是某时某刻在哪拍的。

本毕业设计就是根据基于上述问题而产生的一个解决方案。于是照片管理程序就由此而来，本系统是以 Winform 形式来管理本地的某个文件夹下面的所有受支持的图片格式。该系统由.NET4.7 平台下的 C#语言开发的。

第一章 概述

1.1 开发背景

随着科技的飞速发展，人们的经纪水平比之前有了一个质的飞跃，现如今人们更加注重生活的质量，旅游成为了人们生活中必不可少的一部分，每当人们走过一个景点或者是路过一个比较值得留念的地方是，人们总会情不自禁的拍张照片将这美好的一瞬间永久的保存下来，存到手机或者是存放至电脑中，但是随着时间的流逝，照片拍的也越来越多，这些美好的一瞬间会变的越来越杂乱无章，不方便查找也不方便欣赏。于是乎，照片管理软件也就因此应运而生。这款软件可以轻松的按照照片拍摄的日期和拍摄所在地进行快速的分类。本系统的目的是根据一定的规则分类一些照片。

1.2 照片管理程序的现状

以下选取了三种软件进行分析，如 Windows 照片浏览程序（win10 平台下）、ACDSEE 看如软件（版本是 7.0）和毕业于奥地利一所大学一名学生开发的迷你却五脏俱全的看图软件。

1.2.1 Windows 照片浏览程序

在 pc 端人们最经常用的照片管理程序就是 Windows 自带的照片浏览程序，严格意思上并不是一款专业的照片管理程序，而是一块照片的浏览程序，并没有根据时间或者是根据照片的拍摄地点分类这个功能。手机端的虽然功能比较全，但是手机的容量毕竟是有限的，而且屏幕没有电脑端的大，浏览起来不是特别的方便，最重要的是没有分类这一个功能。

1.2.2 ACDSee 看图软件

ACDSee 是一款看图软件，这款软件是最早可以解码 jpeg 的软件之一，这款软件拥有标准版和专业版，标准版本是专业版本的一个阉割版本，对于普通用户而言标准版本已经

足够使用。专业版本拥有许多加强的功能同时有些功能相当的实用，但是这也意味着要向用户索取费用，这款软件有着如下几种主要功能；1、日常照片的浏览缩略图形式展现 2、常见照片文件格式的转换 3、可以通过幻灯片的形式放映图片 4、可以通过电脑的光驱刻录照片。以上就是这款看图软件的优势。这款软件的不足之处在于 ACDSEE 支持的文件格式越来越多，导致 ACDSEE 消耗的系统资源也越来越多，这也进一步的导致图片的读取速度变的更慢。

1.2.3 IrfanView 图片浏览器

这款图片浏览器是运行再 windows 操作系统下面众多照片管理程序中的一个，它拥有着图片的浏览、可以进行简单的编辑图片和可以转换主流图片的格式，它和 ACDSEE 一样有着商业版本和非商业版本，这意味着用户如果使用商业版本需要付费，同时使用非商业版本是免费的，但是功能相比专业版而言要少许多。这款软件的特点就是体积比其他的都小的多，基本安装后只占用 1.6MB 的磁盘空间，而完整安装则仅仅需要 10MB 的磁盘空间这比其他的要少许多。这是这款软件占用磁盘空间大小的优势所在。但是这款软件的操作语言在操作系统下表现欠佳，在部分非英文文件夹下的图片浏览时选择下一张图片时会产生一个错误。

1.2.4 Picasa 相册

Picasa 是一套应用软件，用于整理和编辑数字照片，起先由 Idealab 所创。Picasa 原为一家位于美国加利福尼亚州帕萨迪纳的公司，提供数字照片自动化管理服务，并发行同名软件。2004 年 7 月，Google 收购 Picasa 并开始提供免费下载。此外，该公司生产的即时通信工具 Hello 也一并归 Google 所有。 2016 年 2 月，Google 宣布自 2016 年 3 月 16 日起停止对 Picasa 的更新与支持，同时在 5 月 1 日终止 Picasa 网络相册（Picasa Web Albums）照片共享服务，此外还将停止对部分 Picasa API 的支持。

Picasa 亦允许用户为相册编制自己的插件或程序。当前官网有三个示范，分别是 Picnik、Booksmart 和 ShoZu。Picnik 可以使用户透过浏览器编辑照片，Booksmart 能将相册中的照片编成一本书，ShoZu 能使用户透过手机上传图片至 Picasa 相册。

Picasa 还可以直接显示编辑 RAW 格式文件。类似的软件包括，苹果的 iPhoto 以及 Adobe 的 Photoshop Elements。

1.3 本论文主要的功能模块

首先第一章主要就是介绍了本系统的开发背景，其次就是选取了三种不同的看图软件进行了对比，对比它们之间的优缺点。

1）第二章内容简要描述

在第二章的第一小节中介绍了.NET 框架，这款框架是由世界软件巨头微软公司研发出来的一款跨平台具有的框架目前仅仅局限于 Windows 平台下。之后便介绍了.NET 框架下的一个很重要的一个名词【公共语言运行环境】这个是.NET 框架的核心之一。在这个小节中同时也介绍了其它的术语。

在第二章的第二个小节中介绍了 WinForms 中的一款 UI 框架-WPF，这款框架是由微软在 2006 年 11 月份推出的，WPF 是 Windows Presentation Foundation 的简称，是.NET Framework3.0 以后版本及其后面版本的组成部分之一。可以看出微软公司还是很重视它的，它使用全新的一种基于 Xml 的标记语言称之为 XAML。

在第二章第三小节中介绍了开发本系统的开发语言 C#，C#语言是由安德斯·海尔斯伯格于主导开发的一种和面向对象的高级编程语言基于.NET 平台。微软公司在 2000 年便发布了 C#开发语言的第一个大版本，微软研发这中编程语言的目的就是希望它能够和当时所流行的 Java 语言同台竞技。就在微软发布 C#语言时，它已经成为了 Ecma 国际和国际标准组织的标准规范。

第四小节则介绍了开发本系统的集成开发环境 VS2017（Microsoft Visual Studio 2017），这款集成开发环境是微软公司开发工具包系列中的一个产品。这款集成开发环境是一个很完善的开发工具集，它包含了软件开发生命周期所涉及到的觉多大数软件，例如需求分析中的 UML 类图等。开发人员可以用它来编写微软旗下所有的产品包括网站，桌面应用程序等。与其他软件一样 VS 更加开发团队的规模大小分为许多个子版本。其中社区版是可以提供给至多 5 人小团队使用的，当然社区版是免费的，本系统使用的就是社区版。

第五小节介绍了本系统分类的核心技术点 Exif 全称是“可交换图像文件格式”，是专门为数码相机所设计的，目前也可以用于手机中的摄像机。

第六小节介绍了 MVVM 设计模式，这中设计模式是由马丁·福勒 PM（Presentation Model）设计模式的其中的一个变体，PM 是由 MVC 设计模式演变出来的，同理 MVVM 也是由 MVC 设计模式演变而来的。MVVM 用相同的方式把视图的行为和状态抽象出来。但是 PM 不是特别依赖于特定的用户界面。

2）第三章简要描述

在第三章节中，主要描述了系统的需求分析下面的总体需求分析和板块功能需求分析，在本章主要阐述需求分析的作用；在系统工程和软件工程中，需求分析是指在创建新系统

或更改现有系统或产品时，需要完成的所有工作，以确定新系统的用途，范围，定义和功能。 需求分析是软件工程中的关键过程。 在此过程中，系统分析员和软件工程师确定客户的需求。 只有在确定了这些需求之后，他们才能分析并寻求新系统的解决方案。

在软件工程的历史中，需求分析一直被认为是软件工程中最简单的一步，但在过去十年中越来越多的人意识到它是整个过程中最关键的过程。。 如果分析师在需求分析期间未能正确识别客户的需求，则最终软件实际上无法满足客户的需求，或者软件无法在指定时间内完成。

3）第四章简要描述

第四章主要介绍了系统的某个功能的具体实现其中包含分类的具体实现、读取照片中的元数据信息、根据照片中的经纬度转换成具体的地理位置（通过调用搞得地图的 WebAPI 接口实现的）、读取并展现图片和实现图片的切换功能。

4）第五章简要描述

第五章主要是讲解了在实现本系统的过程中所遇到的各种问题。另外对对此次毕业设计进行了一次总结。总结本系统的不足及其有待提升的地点。

第二章 开发环境

2.1 .NET 框架

.NET 框架是由微软公司（Microsoft）开发的一套运行平台，微软公司是一个致力于敏捷软件开发（agile software development）、快速应用开发（rapid application development）、与平台无关性（只能够在 Windows 和 Windows Phone 下）和网络透明化的软件框架。.NET 是微软为 2000 年代对服务器和桌面型软件工程迈出的第一步。.NET 里面拥有许多有利于互联网站和内部网应用系统迅捷开发的技术，图 2-1 是.Net 结构图。

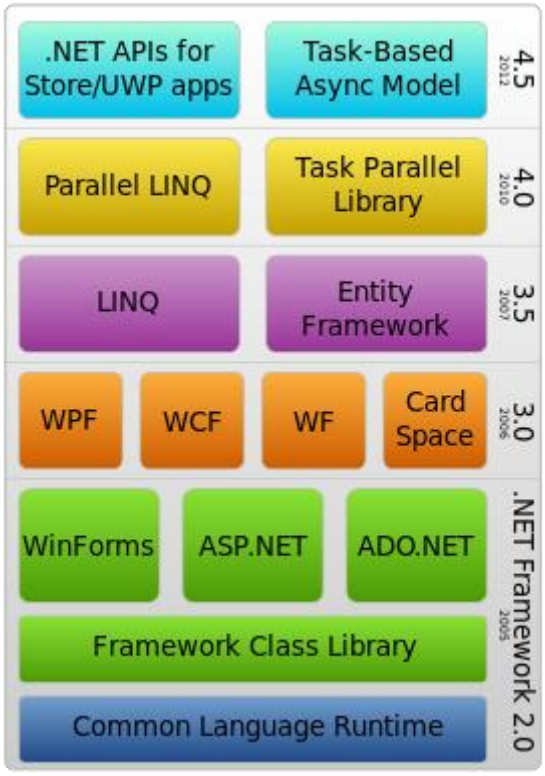

图 2-1 结构图

.NET 框架是微软公司在 Windows DNA 平台之后的开发的一套新开发平台。.NET 框架是用一种采用系统虚拟机（Vm）运行的编程平台，把通用语言运行库（Common Language Runtime）作为基础，同时支持多种面向对象的语言比如（C#、F#、VB.NET、C++、Python 等）。

.NET 也提供对应的应用程序接口（API）的更新。这些更新可以让程序开发任员可以开发 Windows 应用软件的同时也能够进行网络应用软件以及组件和服务（web 服务）的开发。.NET 平台也提供了一个新的反射机制的且面向对象编程编程接口。.NET 设计得足够通用化及拥有足够的灵活性，进而可以拥有不同高级语言都在面向.NET 下开发。

.NET Framework 包含公共语言运行时 (CLR) 和 .NET Framework 基础类库。 公共语言运行时（CLR）是 .NET Framework 框架的基础。 运行时可以被看作是一个在执行时代码的代理程序，它提供对内存的管理、线程的管理和远程的处理等核心服务，并且还强制实施严格的类型安全检查以及可提高类型的安全性和类型的可靠性的等其他形式的来提升代码准确性。 事实上，代码管理的概念是运行时的一个最基本概念和原则。 在运行时上运行的代码被称之为托管代码，而不在运行时为目标的代码被称作为非托管代码。 基础类库是一个综合性的面向对象的可重用类型集合，它可使用它来开发多种不同的应用，这些应用程序可以包括传统的命令行程序或图形用户界面 (GUI) 应用程序，也包括基于 ASP.NET 提供的最新网站的应用技术（如 Web 窗体和 MVC）。

.NET Framework 可由非托管组件承载，这些公用的组件将公共语言运行时加载到它们对用的进程中并且启动托管代码去执行它，从而创建一个同时利用托管和非托管功能的软件环境。 .NET Framework 不但可以提供若干个运行时主机，同时也提供对第三方运行时程序的开发应用。

1) 公共语言运行环境（CLR）

公共语言运行时（CLR）是用来线程执行、管理内存、代码安全验证、编译代码、执行以及其他提供对系统的服务。图 2-2 是 CLR 编译代码过程。

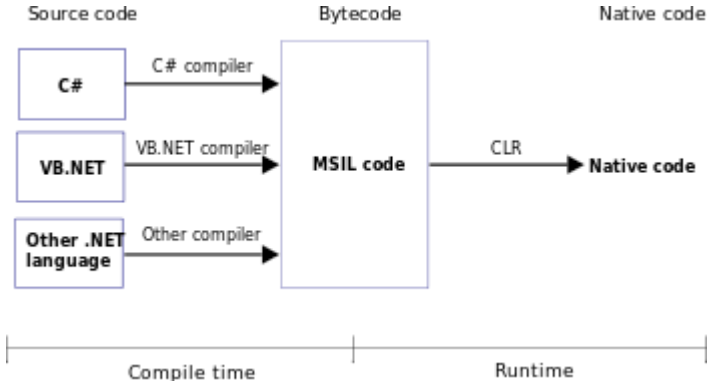

图 2-2 CLR 图示

CLR 环境的安全性，是依赖于托管组件的来源（例如 InterNET、企业网络、本地计算机以及本地网络中）在内的一些因素。 这意味着即使用在同一环境应用中，托管组件既意义执行文件访问操作的同时还可以对注册表访问操作。

运行时还能够通过一种实现称为常规类型系统 (CTS) 的严格数据类型验证和代码的种种检查基础结构进而增强代码可靠性。 CTS 可以确保所有托管代码都是可以通过元数据进行自我描述的。 对于网络上的各种 Microsoft 的 编译器或者是第三方语言编译器都可生成符合 CTS 的规范的托管代码。因此这也意味着托管代码可在严格实施类型保证类型安全的同时使用可以托管类型和实例。

此外，运行时的托管环境还抵消了软件环境中的常见问题。 例如，运行时会自动处理

对象布局同时管理对对象的引用，如果不存在引用时，便会释放掉。 这种自动内存管理系统可以解决了两个最常见最基本的应用程序错误：无效内存引用和内存泄漏（C/C++控制不好尤其容易产生的一种应用程序的错误）。

运行时还可以提高了开发人员的工作效率。 例如，程序员用他们选择的他们最擅长的开发语言编写应用程序，同时却仍然能充分利用其他开发人员用其他开发语言编写的运行时、类库和组件例如（动态链接库）。 任何选择以运行时为目标的编译器供应商都可以这样做。

尽管运行时拥有了高可用的前瞻性同时是为未来的软件设计的，与此同时它也能够支持现有的和以前的软件。 托管和非托管代码之间的互操作性使.NET 开发人员能够通过这种特性继续使用原来所需的 COM 组件和 DLL（由其他托管语言编写而成或者是面向.NET 平台的托管语言）。

运行时的目的是增强应用程序性能。 尽管公共语言运行时提供许多标准运行时服务，但是它从不解释托管代码。 而是一种被称为即时编译器(JIT) 编译的功能使所有托管代码可以以它在其上执行的系统（不同的 CPU 架构下）的本机语言运行。 同时，内存管理器排除出现零碎内存的可能性，并扩大了内存引用区域可以进一步提高应用程序的性能。

最后，运行时可由高性能的服务器端应用程序（如 Microsoft SQL Server 和 InterNET Information Services (IIS)）承载。 这些基础结构可以使您在享受支持运行时承载的行业最佳企业服务器的优越性能的同时，也能够使用托管代码来编写新的业务逻辑。

2.2 WPF 简介

WPF 的核心是一个与分辨率没有任何关系且基于矢量图形的呈现试图引擎，旨在发挥出现代图形硬件的最大效果。 WPF 通过一套完善的应用程序开发功能对该核心功能进行了扩展，这些功能包括了可扩展应用程序标记语言 (XAML，基于 xml)、数据绑定、二维和三维图形、布局、模板、文档、动画、控件、模板、文档、样式、媒体、文本和版式。.NET Framework 包含了 WPF，因此你也可以生成整合其他 .NET Framework 类库元素的应用程序。图 2-3 是 WPF 的结构。

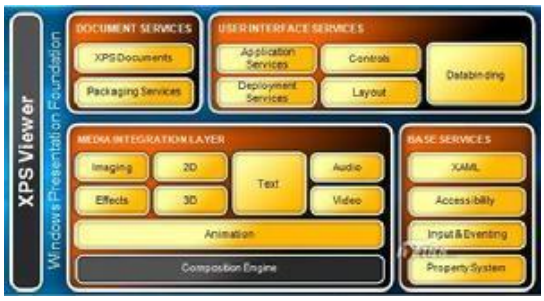

图 2-3 WPF 结构图

通过 WPF，可以使用标记和代码隐藏开发应用程序，这是 ASP.NET 开发人员已经熟悉的体验。通常使用 XAML 标记语言来实现应用程序的外观即 UI 层，同时使用面向.NET 平台下的托管编程语言（代码后置）来实现应用程序的种种行为。 这种外观和行为的分离具有以下优点：

降低了项目的开发和维护成本，因为特定于应用程序的外观的标记与其特定于行为的代码是分开的不是有着很强的耦合性。

通过这种方式开发效率得到了提高，因为设计人员在实现和设计应用程序外观的同时，其他的开发人员可以实现应用程序的种种行为。

2.3 C#简介

C＃是一种基于 Microsoft 引入的.NET 框架的高级面向对象编程语言。 C＃基于.NET Framework 类库，具有类似于 Visual Basic 的快速开发功能能力。C＃是由 Anders Hellsberg 语言学家开发的，他在 2000 年发布了该语言并希望用这种语言取代 Java。C＃已成为 Ecma 国际和国际标准组织的标准规范。示意图如图 2-4 所示。

图 2-4 C#简介示意图

ECMA 标准列出的 C#设计目标：

C#旨在设计成为一种“简单、现代、通用”，以及面向对象的程序设计语言

在这种语言的实现，应该提供对于以下软件工程 5 种要素提供必要的支持：强类型检查、数组维度检查、未初始化的变量引用检测、自动垃圾收集（Garbage Collection，指一种存储器自动释放技术）。软件必须做到强大、持久，并具有较强的编程生产力。

这种语言为在分布式环境中的开发提供适用的组件开发应用。

为使开发人员能够容易迁移到这种语言，因此源代码的可移植性尤为重要，特别是对于那些已熟悉 C 和 C++的程序员而言。

对尽可能支持国际化。

C#可以给独立和嵌入式的系统编写程序，从使用复杂操作系统的大型系统到特定应用的小型系统均适用。

虽然 C#程序在存储和操作能力需求方面具备经济性，但此种语言在某些情况下并不能在性能和程序大小方面与 C 语言相抗衡。

2.4 Visual Studio 2017 集成开发环境

Visual Studio 集成开发环境是一种创新启动板，可用于编辑、调试并生成代码，然后发布应用。 集成开发环境 (IDE) 是一个功能丰富的程序，可用于软件开发的许多方面。 除了大多数 IDE 提供的标准编辑器和调试器之外，Visual Studio 还包括编译器、代码完成工具、图形设计器和许多其他功能，以简化软件开发过程。图 2-5 是 Vs2017 的主窗口图。

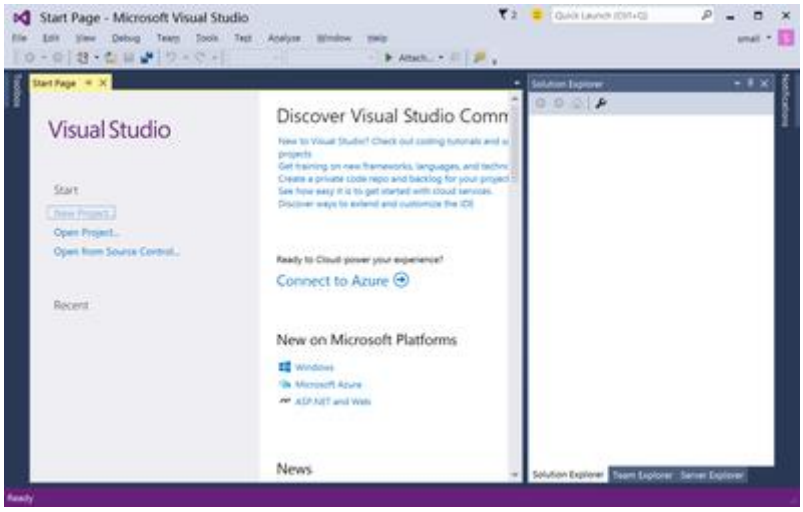

图 2-5 Vs2017 图示

Visual Studio 中的一些常用功能可帮助你在开发软件时提高工作效率，这些功能包括：

1) 波形曲线和快速操作

波形曲线是波浪形下划线，它可以在键入时对代码中的错误或潜在问题发出警报。 这些可视线索使你能立即修复问题，而无需等待在生成期间或运行程序时发现错误。 如果将鼠标悬停在波形曲线上，将看到关于此错误的其他信息。 左边距中也可能会出现一个灯泡，提供修复此错误的“快速操作”建议。

2) 代码清理

3) 重构

4) IntelliSense

5) 快速启动

6) 调用层次结构

7) CodeLens

8) 转到定义

9) 查看定义

2.5 Exif 格式信息

可交换图像文件格式（英语：Exchangeable image file format，官方简称 Exif），是一种专门为数码相机的照片格式设定的，它可以记录数码照片的各种属性信息、拍摄数据以及相机生产商的其它附加的数据。主要格式如下图 2-6 所示。

FFE1	APP1 Marker			
SSSS		APP1 Data Size		
45786966 0000		Exif Header		
49492A00 08000000		TIFF Header		
XXXX. . . .		IFD0 (main image)		Directory
LLLLLLLL				Link to IFD1
XXXX. . . .		Data area of IFD0		
XXXX. . . .		Exif SubIFD		Directory
00000000				End of Link
XXXX. . . .		Data area of Exif SubIFD		
XXXX. . . .	APP1 Data		Interoperability IFD	Directory
00000000				End of Link
XXXX. . . .			Data area of Interoperability IFD	
XXXX. . . .			Makernote IFD	Directory
00000000				End of Link
XXXX. . . .			Data area of Makernote IFD	
XXXX. . . .		IFD1(thumbnail image)		Directory
00000000				End of Link
XXXX. . . .		Data area of IFD1		
FFD8XXXX. . . XXXXFFD9		Thumbnail image		

图 2-6 EXIF 格式图

Exif 最早是由日本电子工业发展协会于 1996 年制定，版本为 1.0。1998 年，升级到 2.1，期间增加了对音频文件的支持。在 2002 年 3 月，发布了 2.2 版本。

Exif 可以附加于 TIFF、RIFF、JPEG 等文件里面，目的是增加有关数码相机在拍摄过程中的种种拍色信息的内容、缩略图或者是图像处理软件的版本信息等其他信息。

Windows 从 Windows 7 操作系统开始才具备对 Exif 的原生支持，用户可以通过鼠标右键点击图片打开图片的属性，点击属性 tab 并切换到详细信息标签下即可直接查看有关 Exif 信息如拍摄时间和相机的生产商等等。

Exif 信息是可以被任意软件编辑的，因此只有参考的功能不能用来决策。

Exif 信息以十六进制 0xFFE1 作为开头标记，之后的两个字节表示 Exif 的长度。因此可以得出 Exif 信息最大为 64 kB，但是而内部采用 TIFF 格式。

2.6 MVVM 设计模式

MVVM 它是一种软件架构的模式，它可以轻易的将开发用户界面的过程和开发后端业务逻辑的过程拆分出来，在 WPF 中是通过后置代码和 GUI 代码组成。MVVM 中的第一个 M 表示系统中真实存在的状态内容面向对象的领域模型，MVVM 的核心在于数据。第一个 V 代表的是和 MVC 中的 V 是一个含义，是用户可以在 UI 界面中看到的视图，及系统的外观。第二 V 代表的是视图模型是暴露在公共属性和命令视图中的一个抽象 MVVC 没有像 MVC 中控制器，最后一个 M 代表绑定器；数据一旦被改变了，那么对应的 UI 也跟着变化。设计模式示意图 2-7。

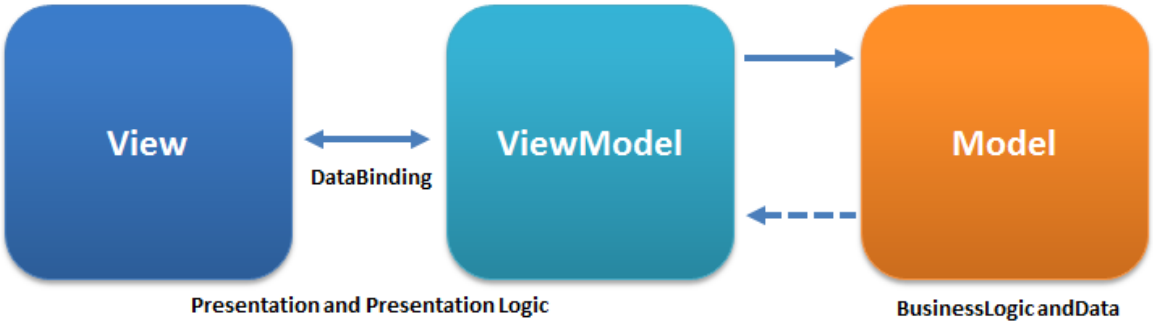

图 2-7 MVVM 设计模式示意图

第三章 系统分析与设计

3.1 系统需求分析

本系统具有浏览照片、缩放照片、读取照片拍摄信息、读取照片的地理位置（经纬度）和根据经纬度转换成对应的地点的文字信息。

3.1.1 总体需求

照片管理程序拥有如下功能，

1）用户输入一个文件夹后程序便读出这个文件夹及这个文件下的所有文件夹下面的照片。

2）先根据照片的拍摄时间和照片的拍摄地理位置进行分类，拍摄时间为一级分类，地理位置信息为第二级分类

3）根据图片的拍摄地点的经纬度信息转换成对应文字描述

4）分类具体以文件夹的信息呈现

系统的主要流程图 3-1 所示。

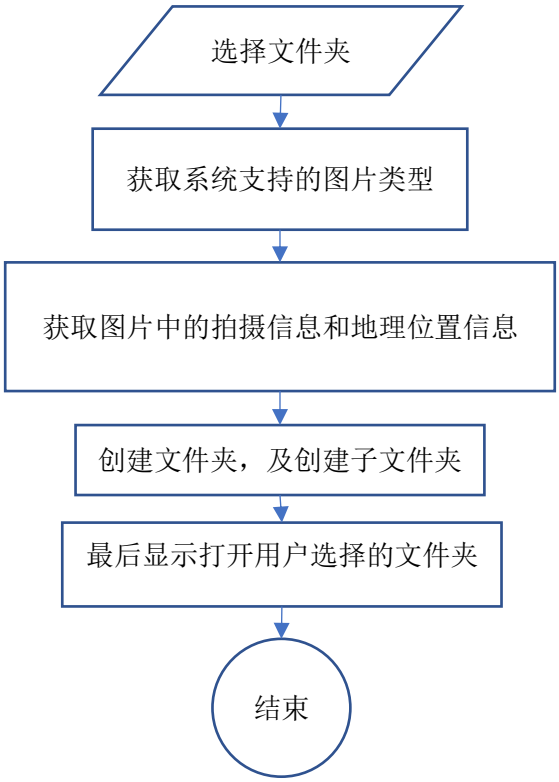

图 3-1 主要流程图

3.1.2 板块功能需求

由总体需求部分可以知道，系统分为浏览照片、读取照片的拍摄时间信息、照片的 gps 信息和根据这两种信息分类。

这个是系统的主页面，该界面由菜单栏、状态栏和照片预览窗口组成。这样设计是借鉴了其他照片管理程序的 UI 设计，与市面上的 UI 保存基本上的一致，不会出来太大的区别以免给用户带来使用上的不便。

1) 打开文件对话框；用户可以点击菜单栏的文件菜单项里面的打开子菜单项可以打开一个目录浏览的对话框，当用户选择了一个目录后，系统会读取下面的所有照片格式。
2) 读取目录下所有子目录的照片；用户给系统提供一个目录，系统会通过递归目录的形式读取每个目录及每个子目录下的照片信息。
3) 读取照片的拍摄时间信息；在照片的元数据中会有一个叫做 Exif SubIFD 目录下面的 TagType 为 36867 的 Tag 就是存放的照片的拍摄信息
4) 读取照片的拍摄地点信息；在照片的元数据中会有一个叫做 GPS 目录下面的就存放相关的经纬度信息。
5) 转换经纬度信息；由于经纬度不是对于人们而言不是那么的友好，因此要将经纬度信息转换成人们可以读取的信息。因此需要调用第三方的 Webapi 将经纬度信息转换成文字地理位置信息。
6) 分类功能；根据读取出的照片拍摄信息和地理位置信息进行分类，实现分类的做法就是再硬盘上以文件夹的信息呈现。

3.2 系统功能设计

综合上面的系统需求分析和板块功能需求分析，本系统包含以下功能，浏览照片、读取照片拍摄时间、读取照片拍摄地点信息和缩放照片信息 。下面分别描述系统功能。

1） 浏览照片功能

照片浏览功能，该功能主要用于浏览用户选择的具体的某个目录及目录下所对应的子目录下面的所有图片，流程图 3-2 所示。

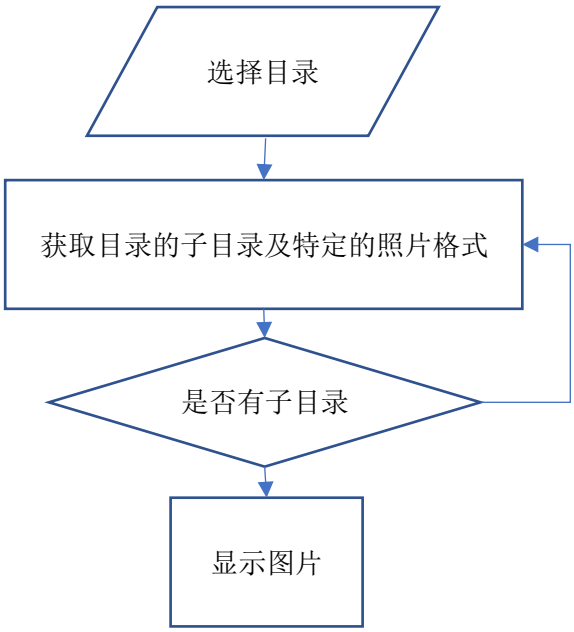

图 3-2 照片浏览

2） 读取照片的拍摄时间信息

读取照片中的拍摄信息；照片的拍摄时间信息再 Exif 中有描述的，但是有些照片可能会丢失了这部分的信息。因此如果读取不到照片的拍摄信息那么就读取照片的创建日期作为照片的拍摄时间。流程如图 3-3 所示。

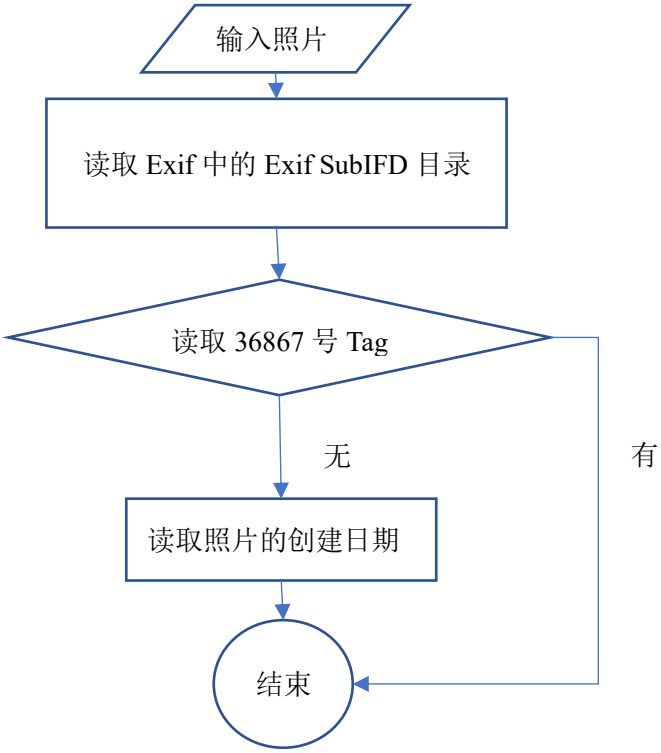

图 3-3 读取照片的拍摄时间信息

3 经纬度转换

通过读取 Exif 信息中 GPS 目录下面的信息，可以读取出经纬度信息和水平高度等信息。在此系统中只读取其中的经纬度信息。流程图如图 3-4 所示。

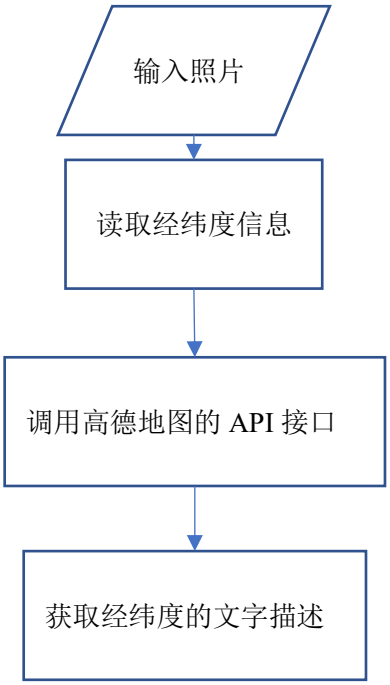

图 3-4 经纬度转换

4 分类功能

系统的分类功能会把照片的拍摄时间作为主分类，把拍摄地点作为次分类。具体表现在同一个时间的照片会在同一个目录下，在经过时间分类的文件夹下面会有地理位置文件夹。同一个拍摄地点会在同一个地理位置文件夹下。流程如下图 3-5 所示。

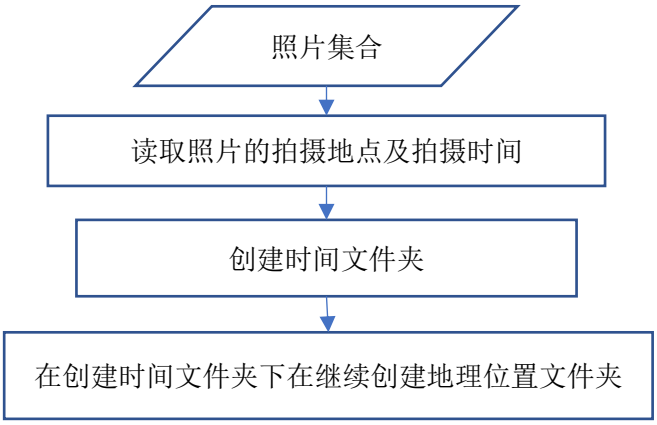

图 3-5 分类功能

第四章 系统实现

4.1 概述

通多第三章的总体需求分析及其系统功能的设计，再结合本人能力的实际情况下结合.NET 平台下的 C#开发语言实现了所有的功能开发。在本章节中，分别介绍了其中某几个关键功能的点实现。

4.2 功能模块

在此小节中主要描述了系统主要的界面。未选择文件夹时的默认界面，有两个向左和向右的箭头可以实现图片的浏览，下班由状态栏，显示用户所选择的某个文件夹，以及一共在这个文件夹下面一种由多少张相片，如图 4-1 所示的主界面。

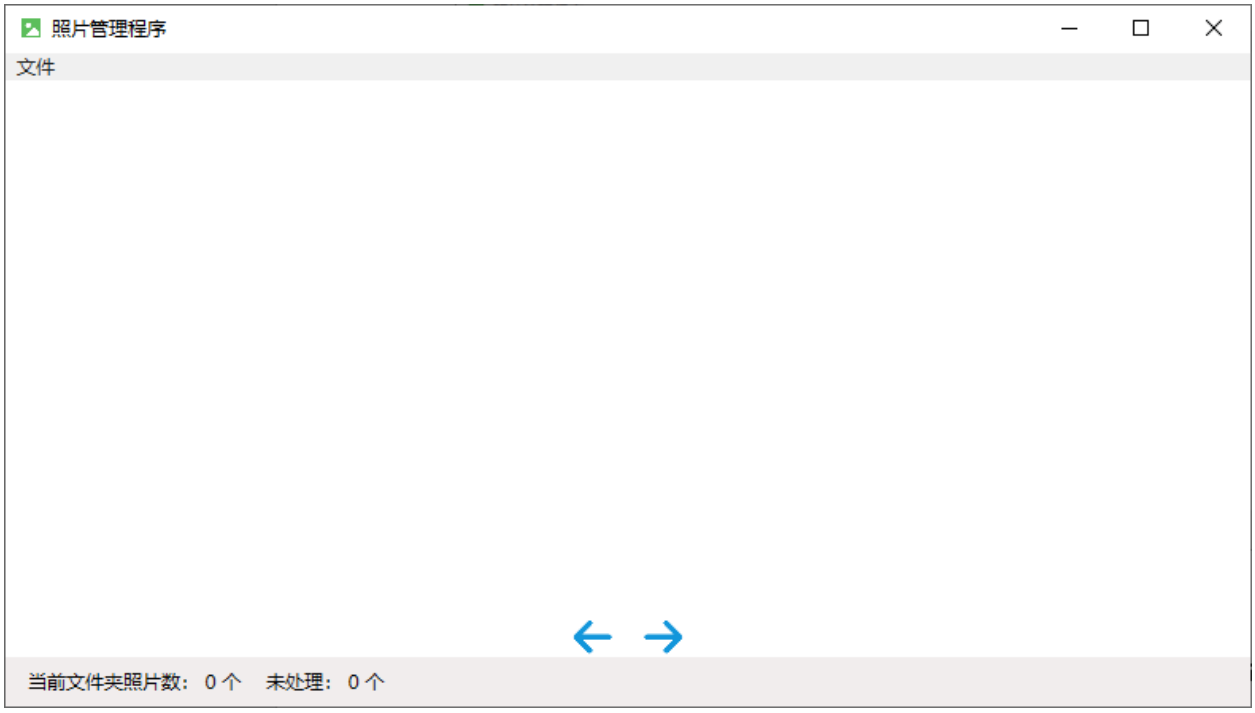

图 4-1 主界面图片

当用户选择某个文件时，状态栏下的文字会更新，例如图 4-2，用户选择的路径，显示出当前路径下一共由 31 张相片。如图 4-2 表示程序的下一个流程。

图 4-2 打开后图片

4.3 关键代码

本小节主要阐述了关键模块的代码实现逻辑，及对应的注释和对应的代码说明。主要描述了如何获取照片的拍摄时间和照片的拍摄 GPS 的信息，以及数据双向绑定在 WPF 中的体现。

4.3.1 获取照片的拍摄日期及 GPS 信息

通过 ImageMetadataReader 的静态 ReadMetadata 方法可以读取出全部的 Exif 信息，该方法返回的是一个只读的集合对象。之后通过遍历只读的集合后根据 item 的 Name 值来判断是否是需要“目录”即 Exif 信息，“目录”为【Exif SubIFD】里面存放了拍摄时间和光圈等信息，本系统只取 Tag 值为 36867 的数据即是照片的拍摄时间。该 Tag 对应的时间格式为“yyyy:MM:dd HH:mm:ss”，但是 Directory 的实例方法【TryGetDateTime】已经帮处理了时间格式。如果该方法返回 False 则表示当前照片没有存放拍摄时间信息因此把照片的创建时间作为拍摄时间。程序 4.1 所示。

程序 4.1 读取拍摄时间信息

```
case "Exif SubIFD":
            if (item.TryGetDateTime(36867, out DateTime create))
            {
```

```
                model.PhotoDate = create;
            }
```

读取照片的 GPS 信息和读取照片的拍摄时间信息的过程是一致的，存储 GPS 信息的目录名称是“GPS”，这个目录下存放了方向、水平高度、经度值和纬度值。经度值和纬度值对应的 Tag 为 2 和 4，通过 Directory 的拓展方法 GetRationalArray 可以获取到经度或纬度的值，该方法返回一个 3 个长度数组，第一值存放的是度，第二个值存放的是分，最后一个值存放的是秒。这中类型的数据不是本系统最终需要，因此还需要通过调用 Rational 数组的拓展方法可以将这些转换成 Double 类型。该拓展方法的主要实现就是通过度分秒转换成小数。程序 4.2 所示。

程序 4.2 读取 GPS 信息

```
case "GPS":
                Rational[] la = item.GetRationalArray(2);//纬度
                model.Latitude = la.ToDouble();
                model.Longitude = item.GetRationalArray(4).ToDouble();
                AmapReturn reut = await amapHelper.Geocode_Regeo(model.Longitude.Value,
model.Latitude.Value);
                model.Address = reut.Regeocode.FormattedAddress;
                break;
```

在程序 4.2 后我们读取到的 GPS 信息是以小数的形式展现的，这个中形式的数据不利于系统之后的分类操作，因此需要一个将小数形式的经度和纬度转换成人类可以接受的一种表达式方式。在系统中是通过调用第三方的 API 接口（高德地图地址逆转换 API 接口）实现将小数形式的转换成文字信息形式如“苏州市人民路 123 号”。在调用 API 接口时使用的时微软推荐的异步编程模型中的 TPL 模型。通过向接口发送 GET 请求并传入约定号的参数后，该接口返回一个 JSON 格式的字符串类型的数据。之后需要通过调用 JsonConvert 的静态方法将字符串类型的 JSON 格式转换成与之对应的一个实体。如程序 4.3 所示。

程序 4.3 经纬度转换为地理信息

```
string result = await
_amapHttpClient.GetStringAsync($"{ApiVersion}/geocode/regeo/?location={lon},{lang}");
                AmapReturn amap =
Newtonsoft.Json.JsonConvert.DeserializeObject<AmapReturn>(result);
                return amap;
            }
```

由于一个文件夹存放照片的数量可能会很庞大，在分类的过程中会可能会由于这个问题会让程序出现卡死的现象。因此系统通过采用多线程的方式来提高系统的分类速度，在系统中开启了 3 个任务即 3 个线程来通知执行分类的方法。这种并行的分类会比常规的分类速度会有一个质的提升。如程序 4.4 所示。

程序 4.4 多线程分类操作。

```
List<Task> tasks = new List<Task>();
for (int i = 0; i < 3; i++)
{
        Task task = Task.Factory.StartNew(executePhoto, ImagePathQueue);
        tasks.Add(task);
}
```

系统会维护一个全局线程安全的队列，线程安全可以理解为在多线程环境下比不是线程安全的队列下，一是不用实现锁的机制，线程安全的队列已经帮我们全部都实现好了 Api 帮我们简化了许多操作；而是更加的完善。通过 ConcurrentQueue 的实例方法 TryDequeue 可以使队列中的一个元素出队。如果该方法返回 False 则表示该当前队列中的元素个数为 0 或者时出现了异常。之后便通过 ImageHelper 的静态方法 HanadleImage 处理当前出队的元素，该方法返回一个 ImageModel 对象，之后在创建问价夹的过程中是很有可能两个线程一起创建同一个名称的文件夹，因此在创建文件夹的过程中只能有一个线程创建，因此使用锁的的机制，保证同一时刻只能有一根线程执行。在创建完文件后系统会将当前元素复制到对应的文件夹内。同时通过改变全局状态的一个属性实现界面上通知用户系统当前已经分类几张照片和照片的总数。4.5、4.6 程序所示。

程序 4.5 分类具体逻辑

```
private async Task executePhoto(object _)
        {
            ConcurrentQueue<FileInfo> qu = _ as ConcurrentQueue<FileInfo>;
            while (qu.TryDequeue(out FileInfo fileInfo))
            {
                lock (Locker)
                {
```

程序 4.6 分类具体逻辑

```
try
                        {
                            if (!directoryInfo.Exists)
                            {
                                directoryInfo.Create();
                            }
                            string targetPath = directoryInfo.FullName;
                            if (!string.IsNullOrEmpty(result.Address))
                            {
                                DirectoryInfo sub = new
DirectoryInfo($"{directoryInfo.FullName}\\{result.Address}");
                                if (!sub.Exists)
                                {
                                    sub.Create();
                                }
                                targetPath = sub.FullName;
                            }
                            fileInfo.CopyTo($"{targetPath}\\{fileInfo.Name}");
                        }
```

4.3.2 数据的双向绑定

在 WPF 中数据的双向绑定主要是实现 INotifyPropertyChanged 接口，之后类下面的每个属性的 set 访问器中必须添加 PropertyChanged 方法，第一个参数表示的事件源，在本类中填写 this 表示事件源是当前类，第二个参数是一个继承 EventArgs 类的一个时间的参数，构造函数传入一个属性名称，表示这个属性被改变了。如程序 4.6 所示。

程序 4.7 当 Set 方法被调用时便通知 UI

```
public Stretch Stretch
```

```
        {
            get => _stretch;
            set
            {
                if (_stretch == value)
                {
                    return;
                }
                _stretch = value;
                this.PropertyChanged(this, new PropertyChangedEventArgs(nameof(Stretch)));
            }
        }
```

在 xaml 界面中中需要使用 xaml 的绑定语法 Source="{Binding Path=ImagePath, Converter={StaticResource ImageConverter}}"表示将数据源的 ImagePath 属性绑定至 Image 控件中的 Source 属性。在后台代码中需要为这个空间指定一个数据源。

4.4 Exif 分析

本系统的核心功能是读取相片中的 Exif 信息，这些信息在相片中可有可无，有照片会有，有的照片有但是可能会不全，如图 4-3 就是一张图片的 16 进制。

```
 Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: FF D8 FF E1 81 0A 45 78 69 66 00 00 4D 4D 00 2A    .X.a..Exif..MM.*
00000010: 00 00 00 08 00 05 01 10 00 02 00 00 00 08 00 00    ................
00000020: 00 4A 01 32 00 02 00 00 00 14 00 00 00 52 01 12    .J.2.........R..
00000030: 00 03 00 00 00 01 00 08 00 00 87 69 00 04 00 00    ...........i....
00000040: 00 01 00 00 00 66 88 25 00 04 00 00 00 01 00 00    .....f.%........
00000050: 01 CE 00 00 02 83 4E 6F 6B 69 61 20 36 00 32 30    .N....Nokia.6.20
00000060: 31 39 3A 30 32 3A 30 35 20 31 31 3A 35 33 3A 31    19:02:05.11:53:1
00000070: 39 00 00 11 82 9D 00 05 00 00 00 01 00 00 01 38    9..............8
00000080: 92 7C 00 07 00 00 00 46 00 00 01 40 82 9A 00 05    .|.....F...@....
```

图 4-3 Exif 格式

从图 4-3 可以清晰的看出一张 JPEG 相片的组成，十六进制 0xFFD8 开头的表示是一中 JPEG 文件，以 0xFFD9 结束。在 JPEG 数据中像 0xFF 表示的是一种标志如 FFD8 表示文件的开头。例如 0xFFE1 表示后面又 E1 个字节的数据换算层十进制表示有 225 个字节，0xFFE0 至 0xFFEF 之间的标识符可以称之为“应用标志”，在解码 JPEG 图像的过程有时候并不是必需使用的。虽然其中有些 Exif 信息存在应用标志当中,以 0xFFE1 作为开头标记，紧跟着后两个字节表示 Exif 信息的长度,内部采用 TIFF 格式来存储。

1 Bytes 0-1 Tag(用于标记这个 IFD 类别)

2 Bytes 2-3 Type(用于指定数据类型 GPS 的数据类似 roration)

3 Bytes 4-7 Count(用于指数据的数量，比如纬度就用度、分、秒三个数来描述)

4 Bytes 8-11 Value Offset(真实数据所在的偏移地址（相对于 File header），而且需要注意的是，这里记录的值小于 4 个字节，则数据左对齐。)

Exif 在定义规范时并没有硬性的规定必须包含哪些 IFD 及其每种 IFD 的顺序。但是，每个一个 Jpeg 文件的第一个 IFD 的位置是可以被准备定位的，之后变从第一个 IFD 后每隔十二字节又可以确定一个 IFD，再根据 Tag 确定 IFD 属于哪种类别，再然后依次类推找到对应 IFD。当然还有一种比较简单的（具有较强的目的性）,十六进制 0x8825 表示是 GPS IFD 入口，以 16 进制的方式打开一张 JPEG 文件后可以直接搜这个然后再根据 Tag 定位可以找到相片的 GPS 信息。比如寻找 GPS IFD Pointer(这个东西可以理解成我们电脑的磁盘,比如 D 盘下一共有 11 个文件夹，其中存在某个个文件夹就是用来存放 GPS 信息的，在 GPS 文件夹下又下面又存在子文件夹也就是子 Tag，比如说经纬度，水平高度，方向等等。)

4.5 加快分类的速度

由于存在许多照片，传统的编程方式可能会导致系统在分类众多照片中的速度极有可能会导致分类速度下降导致界面出现卡死的状态，这样用户的体验会极其不友好。因此在此系统中，采用了多线程的分类方式，在实验中发现单个线程分类 500 张照片的速度会非常的可观，因此可以实现动态的线程分类。在本系统中，采用的是 Task 编程方式，在分类完成后会弹出文件夹分类完所在的文件夹。如图所示 4-4 为分类完成的图。

图 4-4 分类完成后图片

从图 4-4 可以清晰的看出，本来是杂乱无章的照片；但是通过本系统整理照片的功能，每张照片都会处于其对应的目录。 通过读取改目录下的所有图片后获取对应的拍摄时间和拍摄地点，根据时间分组后，接着在根据地点分组便得到如图 4-4 所示的目录结构。

第五章 总结与展望

现在数码相机和手机的相机已经达到了一个比较成熟的阶段，但是由于数码相机相对于手机而言，数码相机缺少手机的便利。而且手机拥有定位的功能。可以通过写入 Exif 信息将相片的种种信息写入到相片中方便应用程序读取。如今人么越来越关注身边的各种美好的信息，更想着将这美好的一瞬间定格。随着相片增多相片的查找越来越困难，面对这些杂乱无章的相片，人们却无能为力。于是乎本系统就由此运用而生。为人们提供一个相片分类的解决方案。

本论文对现有的情况进行了简要的分析，同时介绍本毕业设计涉及到的各种技术如 C# 语言、Exif 信息分析和 WPF 数据的双向绑定。本系统的主要功能就是提取并分析相片中的各种 Exif 信息主要是相册的拍摄时间和拍摄所在的地点。

在本系统中，还是又许多值得去优化的地方。比如主界面设计的不是那么的人性化，缺少相信的提示，如果没有产品的使用文档的话，用户是不知道如何操作的。因此这是一个急切需要进行优化的地方；当用户打开一个很大的图片的同时，本软件就会出现很明显的卡顿和掉帧现象。原因就是文件大以至于 WPF 中的 Image 绘制不过来，因此才会导致本系统会出现卡顿的现象，这也是用户体验非常不好的一个地方。在分类中由于目前的分类方式比较死板，只能根据时间作为第一分类，地理位置信息作为第二分类。这个用户并不能自由的调整分类方式，这个一个不足点之一。

主要参考文献

[1] 微软文档官网 https://docs.microsoft.com/zh-cn/，2019 年 4 月 12 日

[2] Jeffrey Richter， via C#第四版，清华大学出版社，2014 年 4 月 1 日

[3] 刘铁猛，深入浅出 WPF，中国水利水电出版社，2010 年 7 月 1 日

[4] (美)内格尔 (Nagel,C.) 等著，C#高级编程(第 9 版)，清华大学出版社，2014 年 6 月 5 日

[5] 博客园，https://www.cnblogs.com/，2019 年 4 月 12 日

[6] StackOverFlow https://stackoverflow.com/，2019 年 4 月 12 日

[7] Exif 分析 https://www.jianshu.com/p/ae7b9ab20bca，2019 年 3 月 2 日

[8] Exif 简介 https://zh.wikipedia.org/wiki/Exif，2018 年 3 月 1 日

致谢

通过了这几个月的紧锣密鼓完成这个毕业设计的同时，最终我完成了毕业设计。与此同时，我有过至高考之后的轻松。同时却有那么的一种很大的失落感，是因为我的大学生活就这样的短暂的度过了。在整个大学期间我首先感谢的是各位老师的敦敦教诲，同学们和朋友们的关心以及室友们的相互包容，这与我美好的大学生活是分不开的。

在些论文和毕业设计的过程中，感谢班主任兼指导老师的诲人不倦的精神。让我有了信心完成整个毕业设计。同时感谢那些开源的工作者们贡献的程序和参考文档。感谢在 it 行业中热心分享自己的过程及其走过的路。