用U-Boot构建IXP2350目标系统的引导程序

徐亚鹞[1,2],谢凯年[1]
(1.上海交通大学 微电子学院,上海 200030;
2.上海贝尔阿尔卡特(法国)股份有限公司,上海 201206)

摘 要:U-Boot是当前比较流行、功能强大且源码公开的一种通用引导程序,它可以支持x86,PowerPC,ARM,MIPS,NIOS,Microblaze等多种体系结构处理器。IXP2350是Intel公司生产的一款基于XScale内核的高性能网络处理器,广泛应用于宽带接入设备、无线基础设备系统、路由器和多服务交换系统等通信设备中。利用U-Boot来构建嵌入式处理器的引导程序,具有成本低廉、开发周期短等优点。文中对IXP2350的结构作简要介绍,并根据U-Boot的结构、功能和特点,介绍如何利用U-Boot构建IXP2350处理器目标系统引导程序的过程。

关键词:U-Boot;IXP2350;XScale;嵌入式系统;引导程序
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2007)05-0010-05

Build BootLoader of IXP2350 Target System with U-Boot

XU Ya-kun[1,2],XIE Kai-nian[1]
(1.School of Microelectronics,Shanghai Jiaotong University,Shanghai 200030,China;
2.Alcatel Shanghai Bell Co. Ltd,Shanghai 201206,China)

Abstract:U-Boot is a universal and open-source bootloader. It supports many types of microcontrollers. IXP2350 is an Intel powerful network processor basing on XScale core. Its high performance makes it ideal for a wide variety of applications. Using U-Boot to build IXP23xx bootloader will benefit to cutting down cost and saving developing time. This paper is to give some introduction on the feature of U-Boot. And the step-by-step migration of U-Boot to IXP2350 target system is also described in detail.

Key words:U-Boot;IXP2350;XScale;embedded system;bootloader

0 引 言

构建嵌入式系统的引导程序是嵌入式系统开发的一个重要环节,是嵌入式系统的硬件调试和后续软件的开发的基础。如何降低嵌入式系统的开发难度和开发成本通常是嵌入式系统开发者所要面临的问题。U-Boot是当前比较流行、功能强大的通用引导程序,利用U-Boot构建嵌入式处理器的引导程序,具有开发难度小、成本低廉和开发周期短的优点。文中将以构建IXP2350目标系统的引导程序为例,介绍U-Boot在ARM目标系统上的移植过程。

1 IXP2350简介

IXP2350是Intel公司IXP23xx系列网络处理器之一,图1为该处理器的基本结构[1]。该处理器在单颗芯片上集成了4个多线程微引擎ME(Micro-engine)、2个网络处理器引擎NPE(Network Processor Engine)以及一颗32位基于ARMV5内核的XScale处理器。XScale内核主频最高达1.2GHz,包括32k的高速数据缓存和32k的高速指令缓存,512k的二级缓存。4个ME为高性能32位可编程网络处理引擎,每个ME能够同时处理8个线程,可提供高性能的网络数据包处理。两个NPE为200MHz的网络处理器引擎,能够提供多T1/E1、单个从UTOPIA L2、ATM TC over T1/E1、HDLC的处理,其中NPE1可配置为两个10/100M网口,可把其中之一作为调试用网口。该处理器上还集成了灵活丰富的外部接口,包括命令推挽CPP(Command Push-Pull)DDR SDRAM接口、XScale系统互联XSI(Xscale System Interconnect)DDR SDRAM接口、一个兼容32/64位PCI2.2的PCI接口、一个32位的QDR SRAM接口和一个MSF接口等。MSF(Media and Switch Fabric Interface)为网络设备接口,可灵活配置为UTOPIA、POS-2或SPI-3。

收稿日期:2006-07-10
作者简介:徐亚鹞(1977-),男,云南宣威人,工程师,硕士研究生,研究方向为软件工程集成电路设计;谢凯年,博士,研究方向为嵌入式操作系统、软硬件协同设计、片上系统。

由于 IXP2350 强大的数据处理能力和它高性能、高可靠性、灵活扩展的硬件特性,以及丰富的流分类、拥塞管理、队列调度及优秀的 QoS 功能,使得它在通信领域,尤其网络通信领域有着广泛的应用[1]。

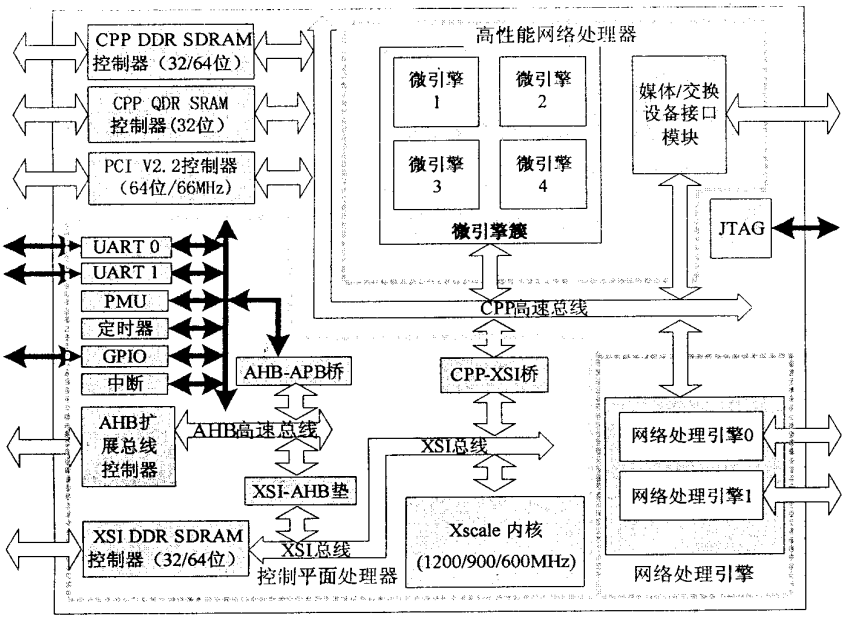


图 1　IXP2350 的基本结构

图中,AHB: Advanced High－performance Bus,高级高性能总线;APB: Advanced Peripheral Bus,高级外设总线;PMU: Performance Monitoring Unit,性能监控单元。

2　U－Boot 的移植

2.1　U－Boot 介绍

2.1.1　U－Boot 的代码结构

U－Boot 是由德国的工程师 Wolfgang Denk 从 PPCBoot 代码发展而来的一种通用启动引导程序,能够引导 VxWorks, Linux, pSOS 等操作系统,它支持 x86, PowerPC, ARM, MIPS, NIOS, Microblaze 等多种处理器[2]。

图 2 是 U－Boot 代码的基本结构[3],整个 U－Boot 目录主要由 CPU 相关代码、板级相关代码、CPU 相关库、通用驱动程序和网络相关代码以及 U－Boot 命令管理组成。CPU 相关代码,U－Boot 所支持的 CPU 相关程序,都是以 CPU 名字命名的目录,比如子目录 arm720t, arm920t, mpc8xx, mips, mpc8260 和 nios 等,每个特定的子目录中都包括 cpu.c 和 interrupt.c, start.S。其中 cpu.c 初始化 CPU、设置指令 Cache 和数据 Cache 等;interrupt.c 设置系统的各种中断和异常,比如快速中断、开关中断、时钟中断、软件中断和未定义指令等;start.S 是 U－Boot 启动时执行的第一个文件,它主要是初始化 CPU 存储器接口,设置系统堆栈和系统工作方式,为进入 C 程序奠定基础。

板级相关文件(Board dependent files)是和一些已有开发板硬件有关的文件,比如具体开发板的硬件地址的分配以及程序链接文件 u－boot.lds,具体开发上硬件设备如 flash 读写操作的驱动程序等,这些程序存放在 board 目录下。

CPU 相关库是一些与 CPU 体系结构相关的代码,也是以 CPU 体系库命名的子目录,例如 lib_arm, ib_i386, lib_mips, lib_nios, lib_ppc,这些目录分别对应于 ARM, I386, MIPS, NIOS, PowerPC 体系结构相关的代码。

U－Boot 提供在线读写 flash, IDE, I^2C, EEROM, RTC 等设备的能力,在 drivers 目录下存放的就是提供这些支持的通用驱动程序。在该目录下,还有各种网卡、支持 CFI 的 flash、串口和 USB 总线的驱动程序。

U－Boot 提供以太网支持,可以从 BOOTP/TFTP 引导操作系统,具有 IP、MAC 预置能力。在 net 目录下存放着这些网络相关的代码,如 BOOTP 协议、TFTP 协议、RARP 协议和 NFS 文件系统的实现等。

U－Boot 能够提供单软件应用程序(standalone application)的运行环境,提供具有读写 I/O、内存、寄存器、外设测试等功能的命令集,提供串行口 kernit、S－record 下载代码,还提供脚本语言的支持。在 common 目录下就是命令处理和脚本处理的 C 代码,该目录下还包括一些与体系结构无关的文件。

U－Boot 提供简单 flash 文件

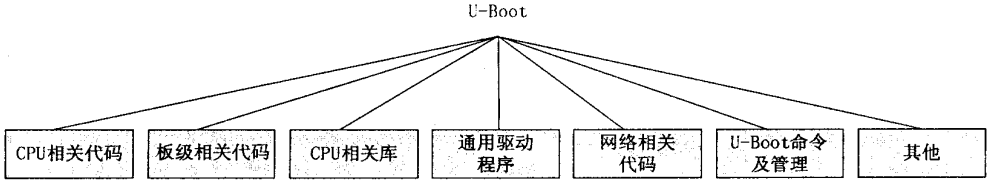


图 2　U－Boot 的代码结构

系统(File System)支持文件,支持 cramfs, ext2, fat, fdos, jffs2 和 reiserfs 文件系统,在 fs 这个目录下存放的就是文件系统 C 代码。此外,U-Boot 还提供数字温度传感器驱动程序、实时时钟驱动程序、上电自检(POST)程序,分别存放在 dtt, rtc 和 post 目录下。

U-Boot 是自由软件,其源代码开放于 sourceforge 网站的社区服务器中,Internet 上有一群自由开发人员对其进行维护和开发,它的项目主页是 http://sourceforge.net/projects/u-boot, U-Boot 的最新版本源代码可以从该站点免费获得[2]。目前的最新版本为 u-boot-1.1.4。

2.1.2 U-Boot 在 ARM 目标系统中的运行流程

图 3 是 U-Boot 在 ARM 系统中的运行流程,在 ARM 目标系统中,U-Boot 的二进制代码被存放在 flash 里,系统会首先在 flash 里面运行,在进行基本的初始化后,将 flash 中的代码复制到 RAM 中,然后从 RAM 中运行,下面具体介绍:

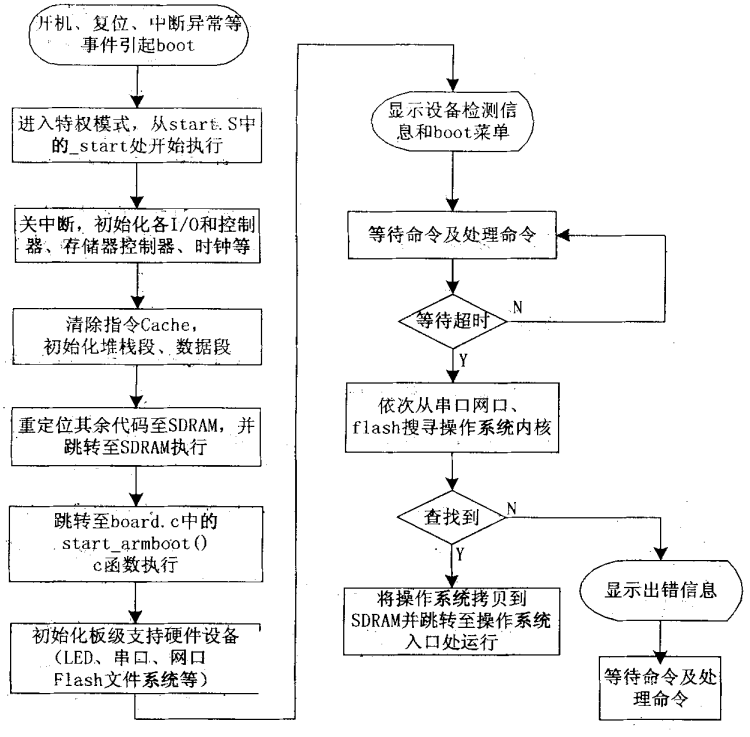


图 3 U-Boot 的运行流程

上电开机或复位之后,CPU 进入特权模式,U-Boot 从/cpu/目标处理器目录下的 start.S 的_start 标识处开始运行,再跳转至 reset 标示处执行,在完成 CPU 自身的初始化例如设置一些 CPU 内部寄存器,完成 SDRAM 以及堆栈的初始化之后,将余下代码从 flash 中复制到 SDRAM 中,跳转至 SDRAM 中运行。

在 SDRAM 中,从 lib-arm 目录下的 board.c 中的 start_armboot 函数入口开始执行,首先通过函数指针数组 init_fnc_ptr 调用一系列初始化函数,完成板级相关的基本初始化。调用的函数及其顺序为:

(1)调用 cpu.c 中的 cpu_init()函数,初始化 CPU 外围,I/O 端口,memory 控制器及复位 CPM 等操作。

(2)调用 board.c 中的 board_init()函数。board_init 函数会调用一系列函数对目标系统进行初始化,如获取系统时钟频率、初始化系统的波特率、初始化串口、初始化控制台和显示选项等操作。

(3)调用 interrupts.c 中的 interrupt_init()函数,初始化中断,初始化 CPU 中断异常处理,挂靠中断服务程序等。

(4)调用 env_init()初始化环境变量。

(5)调用 board.c 中的 init_baudrate()设置串口波特率。

(6)调用 serial.c 中的 serial_init()初始化串口。

(7)调用 console.c 中的 console_init_f()函数初始化控制台。

(9)调用 board.c 中的 display_banner()显示 uboot 启动信息。

(10)调用目标板支持代码中的 dram_init()配置 DRAM 控制器。调用 board.c 中 display_dram_config()函数显示 DRAM 配置信息。

接下来调用 flash.c 中的 flash_init()初始化 flash 文件系统,再调用 display_flash_config()显示 flash 配置信息。

然后再调用 net 目录下 eth.c 中的 eth_initialize()函数初始化以太网口,cpu\目标 cpu 子目录下的 interrupts.c 中的 enable_interrupts()函数使能中断,最后调用 common 目录下 main.c 中 main_loop()等待并处理终端输入命令。

2.2 U-Boot 的移植

2.2.1 开发工具链的建立

在对 U-Boot 代码进行移植之前需要建立开发工具链,以便对 U-Boot 代码进行编译和调试。可从 U-Boot 的官方站点获得 U-Boot 的开发工具的相关信息。U-Boot 官方站点为: http://www.denx.de。可选择 ELDK(Embedded Linux Development Kit)作为开发工具。ELDK 是一套运行在 Linux 操作系统之上的嵌入式 Linux 开发工具,ARM 版的 ELDK 可从站点 http://mirror.switch.ch/ftp/mirror/

eldk/eldk 下载获得[2]。下载之后,可直接进行安装。也可从 http://www.armgnu.com 或 http://www.uclinux.com 获得基于 ARM 的 GNU 工具链。

2.2.2　代码的移植

U-Boot 支持很多种目标系统板的类型,为了减小移植开发的工作量,可以选择一种和自己目标板相接近的类型,对其进行修改以构建自己的目标系统[4]。本例采用的目标板板上处理器为 IXP2350,还包括 8M flash、16Mx64 位纠错 ECC DDR SDRAM 和其他外设。在已有的系统板里,有 ixpd425 目标板,它采用 Intel 的另一款网络处理器 IXP425,该款处理器亦采用 XScale 内核处理器[5],与 IXP23xx 较为接近。下面根据 IXP2350 的目标系统对 ixpd425 目标板 U-Boot 包中的代码进行修改和移植,其过程为:

1)建立 CPU 的支持。到 CPU 目录下,新建 IPX2350 目录,并将与之相近的 IXP 目录下的 start.S, timer.c, pci.c, interrupt.c 复制到该目录,修改其中代码的头文件引用、宏定义等相应内容,将所有 IXP425 字段替换为 IXP2350。由于 IXP2350 只提供 DDR SDRAM 接口,且目标板上使用的是 CPP DDR SDRAM 口,因此需要在 start.S 修改 SDRAM 控制器初始化的代码段,依照 IXP2350 的编程手册,添加 ECC CPP DDR SDRAM 控制器的初始化汇编代码。

2)建立 CPU 依赖文件。到目录 include/asm-arm/目录下,新建目录 arch-ixp2350,并将 arch-ixp 目录下内容复制到该目录,修改 ixp425.h 和 ixp425pci.h 文件名为 ixp2350.h 和 ixp2350pci.h 文件,并对其中代码的头文件引用、宏定义等相应内容进行修改。参考 IXP2350 的编程手册,将 ixp2350.h 中 XScale 内核的控制寄存器和状态寄存器的地址映射的宏定义改为 IXP2350 的地址映射。

3)建立 board 支持。到目录 board 下,新建目标板目录 ixp2350_tbd。将 ixdp425 目录下的内容复制到该目录,修改 ixdp425.c 文件名为 ixp2350_tbd.c。到目录 include/configs 下,依照 ixdp425.h 建立文件 ixp2350_tbd.h。修改目标系统配置文件/include/configs/ixp2350_tbd.h。修改目标板定义文件/board/ixp2350_tbd/ixp2350_tbd.c 中的 I/O 端口 U-Boot 读、写和删除 Flash 设备的源代码文件。由于不同开发板中 Flash 存储器的种类各不相同,所以,需要参考相应的 Flash 芯片手册修改 flash.c 文件。修改的内容包括如下几个函数:

```
unsigned long flash_init (void ) /* Flash 初始化 */
void flash_print_info (flash_info_t *info) /* 打印 Flash 信息 */
int flash_erase (flash_info_t *info, int s_first, int s_last) /* Flash 擦除 */
volatile static int write_data (flash_info_t *info, ulong dest, ulong data) /* Flash 写入 */
int write_buff (flash_info_t *info, uchar *src, ulong addr, ulong cnt) /* 从内存复制数据 */
```

由于许多目标系统不使用 PCI,如果不需要 PCI 的支持,还需对 PCI 相关代码进行修改。具体为将 ixp2350_tbd.h 中的 # define CONFIG_PCI 的宏定义注释掉或改为 # undef CONFIG_PCI。将 cpu.c 中的 cpu-init 函数中的 pci_init()函数删除或注释掉。

由于 ixdp425 使用的网络设备为 Intel i82559,而 IXP2350 的调试用网口需要初始化 NPE1,该初始化需要操作系统的支持,要操作系统引导之后才能进行。因此,不使用网口进行调试,要去除对网络设备的支持。去除网络设备的支持,只需将 ixp2350_tbd.h 中 # define CONFIG_EEPRO100 的宏定义注释掉即可。

IXP425 和 IXP2350 都提供两个串口,它们的控制和状态寄存器地址映射都是一样的,因此串口的驱动可以重用。在 ixdp425 目标板默认的串口为 UART1,波特率 115200。如果使用 UART2 进行调试,波特率为 9600,可将 ixp2350_tbd.h 中的宏定义 # define CFG_IXP2350_CONSOLE IXP2350_UART1 和 # define CONFIG_BAUDRATE 115200 改为 # define CFG_IXP2350_CONSOLE IXP2350_UART2 和 # define CONFIG_BAUDRATE 9600。

4)修改 Makefile。到各新建目录下修改相应的 Makefile,修改和添加要编译的目标代码。

修改文件 u-boot.lds 中 cpu/ixp425/start.o 为 cpu/ixp2350/start.o。

在 Makefile 中添加如下行:

```
ixp2350_tbd_config:unconfig
@./mkconfig $(@:_config=) arm ixp2350 ixp2350_tbd
```

修改 config.mk 中的程序段定义:

由于 IXP2350 采用的 Xscale 处理器是 ARMV5T 架构的[6],所以还需将 cup/ixp2350 目录下 config.mk 中的编译选项 -march=armv4 -mtune=strongarm1100 改为 -march=armv5 -mtune=xscale。

2.3　编译和调试

执行完前面的修改之后,可执行如下操作对代码进行编译:

```
$ make distclean
$ make ixp2350_tbd_config
$ make
```

如若编译通过,可在 U-Boot 目录下面生成了 u-boot.bin 和 u-boot 两个可执行文件,可以将 u-boot.bin 文件烧录至 flash 中,用 U-Boot 文件制作反汇编文件进行调试。

由于 U-Boot 提供单应用程序(standalone application)的运行支持,当包括串口、SDRAM 及 flash 的最小系统建立后,可通过串口加载的单应用程序的方式对目标板的驱动程序进行调试。通过这种方式,将应用程序加载到 SDRAM 中的某一地址,通过 go 命令跳转至该地址执行应用程序,可极大地提高调试效率。

3 结束语

构建 ARM 嵌入式系统的引导程序有许多方法,程序的通用性和可扩展性是引导程序开发所要考虑的重要因素。

笔者移植的 U-Boot 已经能够在目标板上稳定运行,还可以通过串口加载操作系统内核和文件系统。由于 IXP2350 采用的是 XScale 内核,而 XScale 是兼容 ARMV5T 架构的,因此可以在本移植基础之上稍作修改就能构建以 XScale 为内核的处理器系统乃至其他 ARM 内核的引导程序,这种移植具有一定的通用性和可扩展性,可为 ARM 嵌入式系统引导程序的开发提供一些参考。

参考文献:

[1] Intel. IXP2350 Network Processor Product Brief[EB/OL]. 2006. http://www.intel.com/design/network/prodbrf/303678.htm, Intel.

[2] DENX. The DENX U-Boot and Linux Guide (DULG) for TQM8xxL[EB/OL]. 2006. http://www.denx.de/wiki/DULG/Manual, DENX Software Engineering.

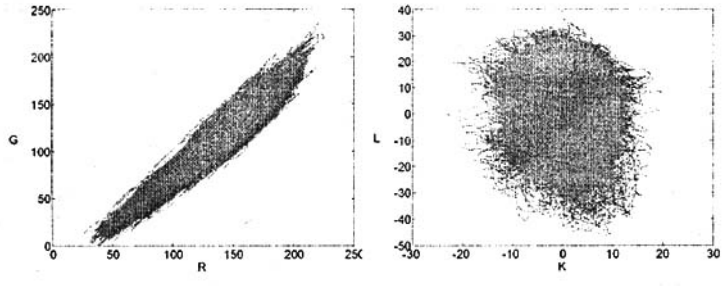
[3] DENX. U-Boot-1.1.4 README[EB/OL]. 2006. http://sourceforge.net/project/showfiles.php? group-id=65938, DENX Software Engineering.

[4] 曾宏安,齐 尧,焦振强,等.用 U-BOOT 构建嵌入式系统的引导装载程序[J].单片机与嵌入式系统应用,2005(2):82-87.

[5] Intel. Intel IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet[EB/OL]. 2005-03. http://www.intel.com/design/network/datashts/252479.htm, Intel.

[6] Intel. Intel XScale Core Developer's Manual[EB/OL]. 2004. http://www.intel.com/design/intelxscale/273473.htm, Intel.

(上接第 9 页)



(a) X 信号的 R-G 面投影图 (b) Z 信号的 K-L 面投影图

图 3 烟叶信号和 KL 变换后的烟叶信号示意图

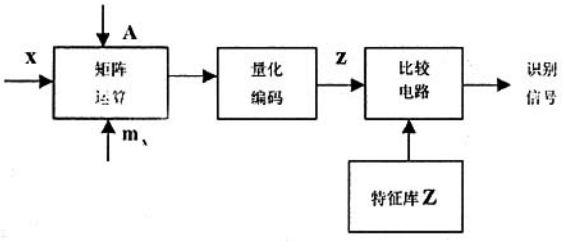


图 4 实时运行过程框图

4 结束语

提出了采用 KL 变换实现建立烟叶特征库的方法,有效地压缩了烟叶阈值空间,具有在线实时运行速度较快、误检率较小的优点。该方法目前已经被应用于杂质在线探测仪(国家九五攻关项目),使用效果良好,同时也验证了算法的有效性。

参考文献:

[1] 明 军,刘严岩,陈应兵.一种优化烟叶信号阈值的新方法[J].仪器仪表学报,2005,26(3):255-257.

[2] Castleman K R. 数字图像处理[M]. 朱志刚等译. 北京:电子工业出版社,2002:244-246.

[3] 孙 辉. 采用 KL-变换的三维谱像数据压缩方法[J]. 长春邮电学院学报,1999,17(1):5-12.

[4] 祁 雷,阎敬文,岳振贵,等. KL-变换 JPEG 和 JPEGKL-变换压缩性能分析[J]. 长春邮电学院学报,1997,15(2):5-9.

[5] 焦春林,高满屯. 基于离散 K-L 变换的彩色图像平滑滤波[J]. 计算机应用研究,2005(4):254-255.

[6] 史容昌. 矩阵分析[M]. 北京:北京理工大学出版社,2004:39-41.