文章编号:1671-4598(2018)03-0124-05 DOI:10.16526/j.cnki.11—4762/tp.2018.03.032 中图分类号:TP302.1 文献标识码:A

基于 Linux 内核 framebuffer 的图像采集与显示系统设计

李 艳,吴 浩,刘政科
(东华大学 机械工程学院,上海 201620)

摘要:针对工业控制领域的自动化生产需要,设计了一种基于 Linux 内核驱动 framebuffer 和 UVC 类型摄像头的图像采集与显示系统;以三星公司的 S5PV210 微控制器为控制单元,基于 Linux 内核移植了 framebuffer 驱动,调用 Linux 内核提供的 V4L2 编程接口,基于 Epoll 架构进行多路图像采集;对采集到 YUVU 格式图像进行编码,转化成 BMP 和 JPEG 保存,以便后续对图片做相应的应用开发;最后调用 libjpeg 库函数对 jpeg 图片进行解码,并显示在 TFTLCD 屏幕上;实验结果显示:能够采集到图片数据,并能成功在 LCD 屏幕上显示,framebuffer 驱动能正常工作,满足工业自动化生产图象处理的需求。

关键词:S5PV210;framebuffer;V4L2;Epoll;libjpeg;图像采集与显示

Design of Image Acquisition and Display System Based on Linux Kernel Framebuffer

Li Yan, Wu Hao, Liu Zhengke
(College of Mechanical Engineering, Donghua University, Shanghai 201620, China)

Abstract: According to the need of automatic production in the field of industrial control, an image acquisition and display system based on embedded Linux kernel framebuffer and USB Video Class type camera is designed. The system uses Samsung S5Pv210 as core controller, and transplated framebuffer driver based on Linux kernel . Calling V4L2 programming interface provided by the Linux kernel, using the UVC types of USB camera multi-channel images acquisition based on Epoll architecture. The image data collected YUVU format is encoded, saved as BMP and JPEG file formats for subsequent development and application. Libjpeg library called by the system to decode JPEG images, and make image displayed in the TFT LCD screen. The result of experiments show that the images can be collected , saved and displayed in the LCD screen, and framebuffer driver can work so that can meet the needs of image process in automatic production.

Keywords: S5PV210; framebuffer ; V4L2; Epoll; Libjpeg; image acquisition and display

0 引言

手机屏幕作为最重要的信息输出部件,承担着人机交互的功能,它的质量好坏直接影响着用户体验,因此,在手机的生产过程中,屏幕的质量检测至关重要。目前国内手机屏幕检测的方法仍然是依靠人工检测,这种方式耗时耗力,并且不能够形成规模化、标准化。

本系统以手机屏幕缺陷采集为应用背景,旨在研究一种小型的低成本的能够实时的适用于手机液晶屏幕流水线生产的图像采集与图象处理的系统装置。基于嵌入式 Linux 平台,以 S5PV210 微控制器为主控单元,7 寸 LCD 液晶屏为图像显示单元;UVC (USB video class) 视频设备协议的 usb 摄像头为系统的图像采集单元,基于最优秀多路 IO 复用机制 Epoll 框架利用 V4L2 编程接口,进行多路图像采集,图像编解码显示,并为后续的图像缺陷检测做准备工作。

收稿日期:2017-10-13; **修回日期**:2017-10-20。

作者简介:李 艳(1978-),女,上海人,副教授,主要从事嵌入式系统,DSP,工业机器人方向的研究。

通讯作者:吴 浩(1991-),男,湖北咸宁人,硕士,主要从事嵌入式 Linux 系统,图象处理,人工智能方向的研究。

1 系统环境的搭建

1.1 系统的硬件平台

为了得到满足图像采集与显示系统需要的最小硬件平台。对 S5PV210 开发板的硬件资源进行裁剪,系统通过摄像头进行图像采集,并将采集数据送至 ARM 处理器进行图像处理,配载 512M 的 SDRAM 作为系统内存和 1G 的 Nand Flash 为系统存储器以挂载嵌入式 Linux 操作系统和运行图形用户应用程序,期间通过 LCD 显示屏将采集图像实时显示给用户,处理结果既可以保存于掉电数据不丢失的 Flash 存储器中。为了调试程序、烧写系统以及与 PC 机的通信,保留了 USB 转串口和网线接口以连接到上位机 PC,最后加入电源管理,组成了整个硬件开发平台。

1.2 系统的软件框架

软件平台采用 Linux,内核版本 2.6.35,为针对目标平台 S5PV210 的硬件特性,首先将启动介质 Uboot 下载到了 nand-flash 中,利用 Busy box 制作了 ext3 格式的根文件系统。由于程序开发的需要,本系统在 uboot 的命令行界面中设置了启动内核与挂载根文件的系统的方式。设置启动命令 bootcmd,使开发板通过 tftp 下载镜像 (zImage)。设置启动参数 bootargs,使开发板从 nfs 去挂载 rootfs (内核配置应使能 nfs 形式的 rootfs)。考虑到内核版本与硬件的不兼容问题,本系统基于 linux 内核驱动框架,移植了 framebuffer 驱动程序。如图 1 所

示，是系统的软件框架图。

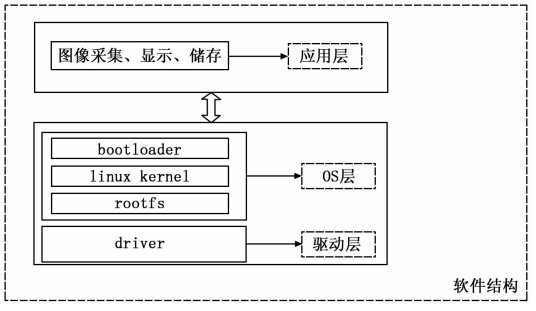

图 1　系统软件框架图

2　Framebuffer 原理及驱动移植

framebuffer 帧缓冲（简称 fb）是 Linux 内核中虚拟出的一个设备，提供给用户态进程一个统一标准接口实现直接写屏操作。从驱动来看，framebuffer 是一个典型的字符设备，而且创建了一个类/sys/class/graphics，用户可以把 Framebuffer 当作是显示内存的一个虚拟映像设备，将 framebuffer 映射到进程地址空间之后，就可以实现对 LCD 屏幕的直接读写操作，而这种写操作可以立刻反应在屏幕上。这种操作是统一的、抽象的，用户完全不必关心物理显存的位置、工作原理、刷新页面等具体细节，这些都是由 Framebuffer 设备驱动来完成。

2.1　驱动框架部分

驱动框架部分，是内核开发人员编写，此部分不涉及具体硬件操作，主要包含 4 个文件。

1）fbmem.c 此文件主要任务是：调用 fbmem _ init (void) 此函数通过 register _ chrdev 接口向系统注册一个 framebuffer 主设备号 29 即 fb 字符设备驱动，同时通过 class _ create 创建 graphics 类，配合 mdev 机制生成一个供给用户访问的设备文件（/dev 目录下）。linux 驱动设备中，所有的缓存显示设备都是由 framebuffer 子系统来管理，即 linux 设备驱动框架只需要知道一个主设备号为 29 的 framebuffer 设备即可。应用层如果要访问 framebuffer 均会被推送给 fb _ fops，统一由此结构体进行分发操作。register _ framebuffer（struct fb _ info * fb _ info）此函数提供 register _ framebuffer 接口给具体 framebuffer 驱动编写着来注册 fb 设备。

单独的显示缓存设备都被视作一个 framebuffer 从虚拟设备，必须要在驱动加载初始化时，通过 register _ framebuffer 接口向 framebuffer 子系统注册自己，才能使自己能够被驱动程序接口调用。如此，当应用程序要访问该 fb 从设备时，才能方便 framebuffer 子系统进行有序操作和管理分发。

2）fbsys.c 此文件中的函数将会被 register _ frame buffer 接口调用，用来实现 frame buffer 在/sys 目录下的一些属性文件 bits _ per _ pixel、modes、stride、rotate…等。

3）modedb.c 即显示模式库文件，此文件是管理显示模式，如分辨率，刷新率，VGA、720P 等。

4）fb _ notify.c 其具体作用：被系统调用，它管理了一个链表，提供给 linux 内核管理，主要用来做一个反向唤醒机制。当注册了一个 framebuffer 驱动之后，系统将会通知这个链表中所有的驱动。

2.2　驱动部分

本系统中 framebuffer 驱动部分是被实现成为了 platform 平台总线驱动，这部分由驱动开发人员来编写，主要做一些硬件相关的工作，硬件初始化，初始化时钟，寄存器，GPIO，中断等等，包括如下 4 个文件。

1）s3cfb.c 驱动主体。属于一个 platform _ driver，通过此文件中的 s3cfb _ probe（）函数，调用操作硬件的函数接口，实现对硬件的操作算法。主要做的工作是：ioremap 实现了寄存器地址的动态映射，分配 frame buffer 的缓存…. 等

2）s3cfb _ fimd6x.c 文件中定义了所有 LCD 硬件寄存器操作的函数，负责对硬件设备做一些具体的初始化，配置时钟，内存映射等。

3）mach－x210.c 负责提供 platform _ device，platformdevices 是在开机时，自动被注册，当 platformdriver 注册之后，二者将会配对。

4）devs.c 为 platform _ device 负责提供具体用到的硬件描述信息即 resource 资源数据，资源数据主要包括 LCD 有关的寄存器地址、IO 资源、中断号等。

2.3　具体移植操作

开发 Framebuffer 设备驱动，存在几个重要结构体，需要重点研究，根据 Framebuffer 设备驱动程序的结构，驱动主要跟 struct fb _ info 此结构体有关，此结构体记录了 Framebuffer 设备的几乎全部信息，具体包括设备的参数设置、状态信息和底层硬件操作的函数指针等。在 Linuxkernel 驱动中，每一个 Framebuffer 设备都对应一个 fb _ info 才能完成驱动框架部分的接口函数。

1）struct fb _ info 此结构体定封装了 frame buffer 驱动的所有有效信息。

2）struct fb _ ops，是 fb _ info 成员结构体，fb _ ops 硬件操作接口集包含很多接口，如设置可变参数、设置颜色寄存器、清屏接口、画位图接口、内存映射等。

3）fb _ fix _ screeninf 此结构体是 fb _ info 成员结构体，填充用户不可修改的参数，包如显示内存的物理地址和长度等。

4）struct fb _ var _ screeninfo 是 fb _ info 成员结构体，填充用户可以修改的参数，包括屏幕分辨率、每个像素比特数等。

5）struct fb _ fops，是 Framebuffer 属于字符设备，用户通过 fb _ fops 结构中定义的文件操作接口函数可以操作 Framebuffer 设备。移植需要的具体操作如下：

1）打开\kernel\drivers\video\Kconfig 文件，添加代码：config FB_S3C_EK070TN93

```
bool "EK070TN93"
depends on MACH_SMDKV210 || MACH_SMDKC110
select BACKLIGHT_PWM
```

2）初始化 LCD 控制器。包括时钟信号的配置，信号的极性设置，VSYNC、HSYNC 时序配置，像素配置，使能通道，LCD 屏幕坐标设置，framebuffer 始末位置设置，frame buffer 在虚拟屏幕中的偏移量设置，输出模式等。

3）添加信息到设备 devs.c 中实现的加入控制器信息到 device 设备，如此，驱动框架才会认识新加入的设备名字，通过设备名字去查找相应的 probe 函数。

4）在系统初始化中增加对 lcd 的初始。即在系统初始化文件中添加：.init _ machine = x210 _ lcd _ init；5）make me-

nuconfig 配置，在 console 选项中还要加入 frambuffer support 选项。

6）最后 make 编译并执行，移植步骤完毕，如果移植完毕后，检测没有移植成功，可以在如下 3 个文中件排查问题，：menuconfig、Makefiel、Kconfig。

经过如上步骤，已经完成了 framebuffer 驱动在 linux kernel 中的移植工作，已经与 S5PV210 开发板的硬件完美兼容，接下来的只需要编写应用程序调用 framebuffer 驱动，将采集到的图像显示在开发板的 LCD 屏幕上。

3 基于 Epoll 框架的图像采集

Epoll 是 Linux 内核为处理大批量文件描述符而作了改进的 poll，是 Linux 下多路复用 IO 接口 select/poll 的增强版本，其工作原理如图 2 所示：在大量并发连接中只有少量活跃的情况下，能显著提高程序系统 CPU 利用率。获取事件的时，无须遍历整个被监听描述符集，只要遍历被内核 IO 事件异步唤醒而加入 Ready 队列的描述符集合。

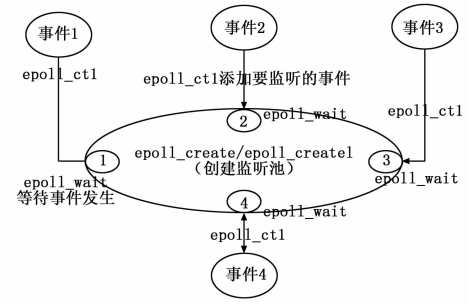

图 2 Epoll 机制工作原理

V4L2（Video For Linux Two）是 Linux 操作系统下用于采集图片、视频和音频数据的 API 接口，配合适当的视频采集设备和相应的驱动程序，可以实现图像采集。在 Linux 操作系统中一切皆文件，在本系统中，摄像头设备文件是" /dev/video0"。V4L2 支持两种方式来采集图像：内存映射方式（mmap）和直接读取方式（read）。V4L2 在 videodev. h 文件中定义了一些重要的数据结构，在采集图像的过程中，通过对这些数据的操作来获得最终的图像数据。如图 3 所示是 V4L2 在 linux 中的驱动框架如图 3 所示。

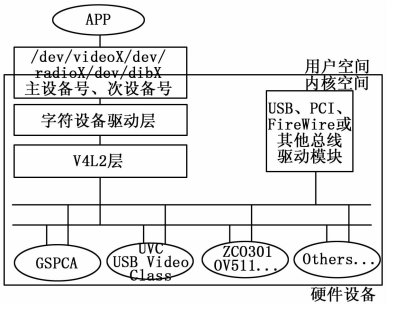

图 3 V4L2 驱动框架图

如图 4 所示是 epoll 框架的 V4L2 图像采集流程：

第一步，打开视频设备文件，通过 V4L2 接口进行视频采集的参数初始化。

第二步，申请若干视频采集的帧缓冲区，并将这些帧缓冲区从内核空间映射到用户空间，便于应用程序读取并处理图像数据。

第三步，将申请到的帧缓冲区在视频采集输入队列排队，并启动视频采集。

第四步，开始视频数据的采集，应用程序从视频采集输出队列取出帧缓冲区，处理完后，将帧缓冲区重新放入视频采集输入队列，循环往复采集连续的视频数据。

第五步，在事件处理函数 void cammer _ handler（int fd，void * arg）里面保存图像数据。

第六步，由于次摄像头只支持 YUVU 格式图片采集，需要调用函数对图片格进行转化，最后采集到的图像数据 image _ jpeg. jpg 和 image _ bmp. bmp。

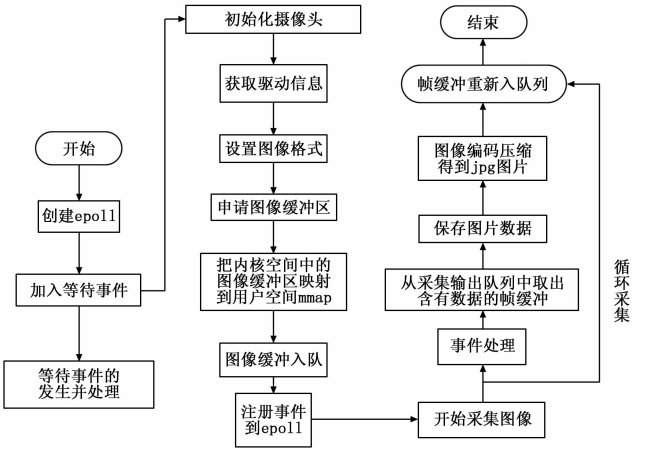

图 4 基于 Epoll 框架的图像采集软件流程图

4 基于 framebuffer 图像解码显示

4.1 framebuffer 图像显示

framebuffer 帧缓冲（简称 fb）是 Linux 内核中虚拟出的一个设备，它向应用层提供一个统一的标准编程接口，它向应用层屏蔽了驱动层的一些细节，方便用户进行应用编程。从驱动来看，fb 是一个典型的字符设备。如图 4 是 frame buffer 驱动框架图。本系统中基于 framebuffer 驱动，图像显示程序的关键如下：

（1）设备文件 fbfd = open（FBDEVICE，O_RDWR）；

（2）获取设备信息 vinfo. xres，vinfo. yres，vinfo. bits_per_pixel

（3）mmap 做映射 pfb = mmap（NULL，len，PROT _ READ | PROT_WRITE，MAP_SHARED，fbfd，0）；

（4）填充 framebuffervoid fb_draw2（const struct pic_info * pPic）

由此可以直接利用 Video4Linux 采集的图像数据，映射到 Framebuffer 的内存区域中，便可以直接显示到 LCD 上。也可将摄像头采集到的数据直接保存成 JPG 或 BMP 等格式的图像文件后，利用 jpeglib 解码，映射到 Framebuffer 的内存区域中，同样可以显示到 LCD 上。

4.2 libjpeg 库的移植及 JPEG 图片解码

JPEG 静态图片压缩标准是一种被广泛认可的图像格式标准，JPEG 标准是在变换编码的基础上，综合应用了 DCT 和哈夫曼编码两种手段达到了很好的图像压缩效果，基于 D C T 的编码方法是 JPEG 算法的核心内容。JPEG 算法的实现流程主要包括图像分割、颜色空间转换、DCT（Discrete cosine transform）、Quantization（数据量化）、Huffman coding（数据进行编码）。如图 5 是 DCT 基压缩解压框图。

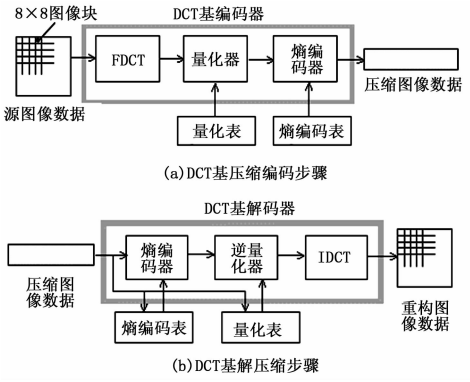

图 5　DCT 压缩解压框图

如图 5 所示：图像压缩大致可以概括成 3 个步骤，原始图像经映射变换后的数据，经量化器和熵编码器后以码流的形式输出。

1）映射变换。通过映射改变图像像素的特性，更加有利于对图像进行压缩编码。

2）量化器。对映射后的数据进行量化，可分为标准量化和矢量量化。对映射后的数据进行逐个的量化，称标准量化，若对映射后的数据成组的进行量化，称矢量量化。量化必然会造成图像的某些信息丢失，导致失真，即量化失真或量化噪声。量化的精细程度与压缩比例始终是对立的，所以应选用适当的量化级数和量化曲线形状来平衡这对矛盾。量化器的使用是图像编码产生失真的根本原因，如果使用了量化器，不可能保证复原图像与原始图完全一致。

3）熵编码器。其作用是用来消除符号编码冗余度的，一般不导致失真，理论上，编出的码流的平均码长应该等于量化后数据的信息熵。

DCT 变换就是利用傅立叶变换的特性. 采用图像边界褶翻将像变换为偶函数形式，接着对图像数据进行二维傅立叶变换，变后就只包含余弦项。所以称之为离散余弦变换。

首先，将图像分割成 8 * 8 的小块，在 JPEG 压缩算法中，通常情况是将颜色空间转换成 YCbCr 空间，Y 表示亮度，Cb 和 Cr 分别表示蓝色和红色的色差值，其转换公式为：

$$Y = 0.289R + 0.589G + 0.124B$$

$$Cr = (0.501R - 0.4186G - 0.0823B) + 128$$

$$Cb = (-0.1687R + 0.3313G + 0.500B) + 128$$

DCT 变换的本质就是将原来的小块图像投影到新的空间中，经过 DCT 变换后，原本杂乱的数据将变得工整。如下是 DCT 变换的公式：

元素公式：$F(u,v) = c(u)c(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\cos[\frac{(i+0.5)\pi}{N}u]\cos[\frac{(j+0.5)\pi}{N}v]$ 矩阵公式：

$$F = AfA^T$$

$$A(i,j) = c(i)\cos[\frac{(j+0.5)\pi}{N}i]$$

上面两式中，$C(u)$，$C(v) = 1/\sqrt{2}$，当 u，v=0，f 表示像素值的数组，$f(i,j)$ 表示 i 行 j 列的值，则离散余弦变换后定义一个新的数组 $F(u,v)$，表示 u 行 v 列的值。

在 DCT 变换之后，图像的数据信息并没有丢失，需要对其进行量化处理 Quantization，根据图像中的数据元素的使用频率，调整元素的编码长度，以获最优压缩比，进行哈弗曼编码压缩，至此在保证不丢失信息的前提下，则可以实现数据的无损压缩。本系统通过上面的基于 epoll 框架的图片采集可以得到的是经过编码压缩后的 jpg 格式的图片，jpg 格式的图片数据量比 bmp 格式图片小很多，有利于图片传输与保存。

4.3　libjpg 解码显示具体实现

libjpeg 是一个完全用 C 语言编写的库，被广泛应用于 JPEG 解码、JPEG 编码等应用中。首先需要移植 libjpeg 到系统中，编写 int is _ jpg（const char * path）函数，判断是否是 jpg 图片。static int jpg _ analyze（struct pic _ info * pPic）此函数功能：解码 jpg 图片，并将解码出来的数据存储；函数参数：pPic，记录源 jpg 图片，解码出来的图片宽高、图片数据缓冲区等信息。如图 6 所示，是本系统 jpg 图片解码显示软件流程图。具体实现如下：

第 1 步：错误处理函数部分的绑定，给解码器做必要的内存分配和数据结构的初始化。

```
cinfo.err = jpeg_std_error(&jerr.pub);jerr.pub.error_exit = my_error_exit;
jpeg_create_decompress(&cinfo);
```

第 2 步：将 fopen 打开的源 jpg 图片和解码器关联 jpeg _ stdio _ src（&cinfo，infile）；

第 3 步：读 jpg 文件头；

```
jpeg_read_header(&cinfo, TRUE);
```

第 4 步：启动解码器 jpeg _ start _ decompress（&cinfo）；

第 5 步：逐行解码并将解码出的数据丢到缓冲区 jpeg _ read _ scanlines（&cinfo，&buffer，1）；

第 6 步：解码完了，做清理工作 jpeg _ destroy _ decompress（&cinfo）；

第 7 步：调用函数，从图片存放路径，递归检索图片，显示到 LCD；

```
int display _ jpg (const char * pathname);
fb _ draw2 (&picture);
```

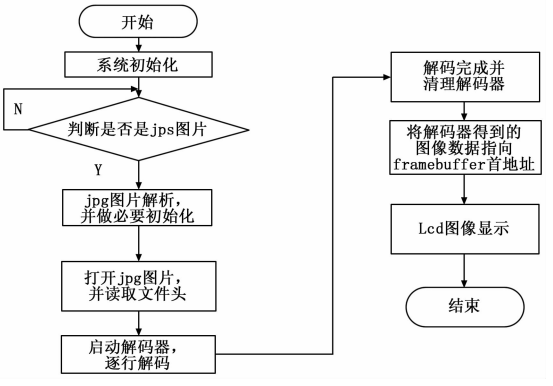

图 6　jpg 图片解码显示软件流程图

本程序能够实现递归检索文件夹，bmp、jpg 图片解码显示 LCD 屏幕上，同时还是实现了触摸屏播放显示。本系统选取了手机屏幕缺陷中的白屏黑点屏，黑屏亮点屏，绿屏黑点屏，花屏黑点屏进行实验测试，如图 7 所示，是 jpeg 解码显示测试图，本系统能够将摄像头采集到的手机液晶屏幕缺陷图像显示在开发板的 LCD 显示屏上。

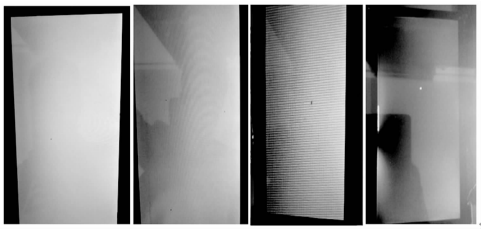

图 7 jpeg 解码显示测试图

5 结论

本文设计了基于嵌入式 Linux 的图像采集与显示系统，能够实现多路摄像头实时采集图像，并且能够实时显示在 LCD 屏幕上，还实现了图像的 BMP 格式与 JPEG 格式保存，有利于后续对图片做一些图像检测，例如 TFT 手机屏幕缺陷检测，基于深度学习模式识别等应用。在大型工厂车间多条流水线快速作业的手机屏幕生产线上，可以利用此系统装置实时快速的对手机液晶屏幕总成缺陷进行自动检测，代替人工肉眼检测的同时，又避免了耗费大量的人力物力财力。不仅减少了人工的重复劳动和生产成本，节约了时间和空间，提高了生产效率，而且提高了手机液晶屏生产线的自动化程度。本系统适合工业自动化领域，它成本低、功耗低、体积小，可以方便的应用到图像采集，图象处理，视频监控，安全防范等项目中。

参考文献：

[1] 刘峥嵘，张晓薇，俞 辉，等编著. 嵌入式 Linux 应用开发 [M]. 北京：机械工业出版社，2004

[2] 韦东山. 嵌入式 Linux 应用开发完全手册 [M]. 北京：人民邮电出版社，2008.

[3] 刘 森. 嵌入式系统接口设计与 Linux 驱动程序开发 [M]. 北京：北京航空航天大学出版社，2006

[4] 于殿泓. 图像检测与处理技术 [M]. 西安：西安电子科技大学出版社，2006.

[5] 谭浩强. C 语言程序设计 [M]. 北京：清华大学出版社，1992.

[6] 许宏松. Linux 应用程序开发指南 [M]. 北京：机械工业出版社，2000.

[7] 杜春雷. ARM 体系结构与编程 [M]. 北京：清华大学出版社，2003.

[8] 宋宝华. Linux 设备驱动开发详解 [M]. 北京：人民邮电出版社，2008.

(上接第 123 页)

且基于 ROS 框架的自定位算法具有很好的可移植性与扩展性等优点，为下一步研究路径规划与自主导航等问题奠定了良好的基础。

参考文献：

[1] Wang Y, Wang D, Mihankhah E. Navigation of multiple mobile robots in unknown environments using a new decentralized navigation function [A]. Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on [C]. IEEE, 2016: 1-6.

[2] Patompak P, Jeong S, Chong N Y, et al. Mobile robot navigation for human-robot social interaction [A]. Control, Automation and Systems (ICCAS), 2016 16th International Conference on [C]. IEEE, 2016: 1298-1303.

[3] 刘开创，施家栋，王建中，等. 移动机器人自主返航控制系统设计与实验 [J]. 计算机测量与控制，2016，24 (3)：71-75.

[4] Handschin J E. Monte Carlo techniques for prediction and filtering of non-linear stochastic processes [J]. Automatica, 1970, 6 (4): 555-563.

[5] Bukhori I, Ismail Z H. Detection strategy for kidnapped robot problem in Monte Carlo Localization based on similarity measure of environment [A]. Underwater System Technology: Theory and Applications (USYS), IEEE International Conference on [C]. IEEE, 2016: 55-60.

[6] Kok M, Hol J D, Schon T B. Indoor positioning using ultrawideband and inertial measurements [J]. IEEE Transactions on Vehicular Technology, 2015, 64 (4): 1293-1303.

[7] Jeábek J, Zaplatílek L, Pola M. A proposal of radio ultra wide band systems for precision indoor localization [A]. Radioelektronika, 2015 25th International Conference [C]. IEEE, 2015: 355-358.

[8] Loevsky I, Shimshoni I. Reliable and efficient landmark-based localization for mobile robots [J]. Robotics and Autonomous Systems, 2010, 58 (5): 520-528.

[9] Iocchi L, Nardi D. Hough localization for mobile robots in polygonal environments [J]. Robotics and Autonomous Systems, 2002, 40 (1): 43-58.

[10] Xiong D, Xiao J, Lu H, et al. The design of an intelligent soccer-playing robot [J]. Industrial Robot: An International Journal, 2016, 43 (1): 91-102.

[11] Grisetti G, Stachniss C, Burgard W. Improved techniques for grid mapping with rao-blackwellized particle filters [J]. IEEE transactions on Robotics, 2007, 23 (1): 34-46.

[12] Zhang W, Li H, Yan X, et al. A Method of Recognizing Curve Direction Based on Hough Transform [A]. Computational Intelligence and Design (ISCID), 2016 9th International Symposium on [C]. IEEE, 2016, 2: 3-6.

[13] Lauer M, Lange S, Riedmiller M. Calculating the perfect match: an efficient and accurate approach for robot self-localization, RoboCup 2005 [A]. robot Soccer World Cup IX [C]. 2006.

[14] Riedmiller M, Braun H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm [A]. Neural Networks, 1993, IEEE International Conference on [C]. IEEE, 1993: 586-591.

[15] Martinez A, Fernández E. Learning ROS for robotics programming [M]. Packt Publishing Ltd, 2013.

[16] Hamzeh O, Elnagar A. A Kinect-based indoor mobile robot localization [A]. Mechatronics and its Applications (ISMA), 2015 10th International Symposium on [C]. IEEE, 2015: 1-6.

[17] Kamarudin K, Mamduh S M, Shakaff A Y M, et al. Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques [J]. Sensors, 2014, 14 (12): 23365-23387.