实验一　图像获取与直方图均衡化

1　实验目的

① 掌握 ROS 下通过 OpenCV 读取 ZED 相机的图像的方法

② 掌握通过 topic 控制机器人运动的方法

③ 掌握图像的直方图计算及其均衡化原理，并设计算法

2　实验仪器

① 机器人硬件：移动机器人、笔记本

② 系统和软件：Ubuntu16.04、ROS-kinetic-full、OpenCV（仅用于实现图像的读取操作）

3　实验原理

3.1 图像直方图计算

图像直方图使用以表示数字图像中亮度分布的直方图，标绘了图像中每个亮度值的像素数，如图 1 所示。将直方图除以面积进行归一化就成了概率密度函数，图像中灰度值为 k 的像素得到概率密度满足：

$$p_k = \frac{n_k}{N}$$

其中 n_k 为灰度值为 k 的像素数，N 为图像中的像素总数。

Fig 1　图像及其直方图

3.2 直方图均衡化

直方图均衡化广泛应用于图像增强处理。图像的像素灰度变化是随机的，直方图的图形高低不齐，把原始图像的灰度直方图从比较集中的某个灰度区间变成在全部灰度范围内的均匀分布。直方图均衡化就是对图像进行非线性拉伸，重新分配图像像素值，使一定灰度范围内的像素数量大致相同。直方图均衡化就是把给定图像的直方图分布改变成“均匀”分布直方图分布。

设 r 和 s 分别表示原图像灰度级和经过直方图均衡化的图像灰度级，并将其归一化。设连续图像的概率分布为$P_r(r)$和$p_s(w)$，假定直方图均衡化变换函数为$s=T(r)$，则变换前后有：

$$\int_r^{r+\Delta r} p_r(w)dw=\int_s^{s+\Delta s} p_s(w)dw$$

设$C(r)$为累积分布函数，一般地有：

$$\int_{s\min}^{s} p_s(w)dw=\int_{r\min}^{r} p_r(w)dw=C(r)$$

若期望变换后输出图像的概率密度均匀分布，即：

$$p_s(s)=\frac{1}{s\max-s\min},$$

则累积分布函数满足：

$$\begin{aligned}C(r)&=\int_{s\min}^{s}\frac{1}{s\max-s\min}dw\\&=\frac{1}{s\max-s\min}(s-s\min)\end{aligned}$$

因此，像素值经过直方图均衡化后满足：

$$s=[s_{\max}-s_{\min}]C(r)+s_{\min}$$

4 实验内容

4.1 创建实验程序工程

我们要通过程序对图像进行数字处理，需要在程序中实现对图像的操作。

① 打开新的终端，并新建工作空间

```
$ cd
$ mkdir -p dip_ws/src
$ cd dip_ws/
$ catkin_make
```

② 创建新的功能包并编译

```
$ cd src/
$ catkin_create_pkg image_pkg
$ cd ../
$ catkin_make
$ source devel/setup.bash
```

③ 新建读取图像的 cpp 文件

进入功能包

```
$ cd src/image_pkg/
```

创建 src 目录和 cpp 文件

```
$ mkdir src && cd src
$ gedit exptl_grayProcessing.cpp
```

在 exptl_grayProcessing.cpp 撰写摄像头读取图像的代码(参考附录的代码)

④ 配置 CMakeList.txt:清除 image_pkg/CMakeList.txt 原文件内容后，在文件中增加如下几行代码：

```
cmake_minimum_required(VERSION 2.8.3)
project(image_pkg)

#以下语句用于解决无法找到 opencv 配置文件问题
set(OpenCV_DIR  /opt/ros/kinetic/share/OpenCV-3.3.1-dev)
find_package(catkin REQUIRED roscpp OpenCV)
catkin_package()
include_directories(${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS})
add_executable(exptl_grayProcessing src/exptl_grayProcessing.cpp)
target_link_libraries(exptl_grayProcessing    ${catkin_LIBRARIES}
${OpenCV_LIBS})
```

⑤ 在终端下编译

```
$cd ~/dip_ws
$catkin_make
```

⑥ 测试

笔记本电脑终端上启动 roscore

```
$ roscore
```

在另外一个终端界面，输入以下命令：

```
$ rosrun image_pkg exptl_grayProcessing
```

如果运行之后发现无法找到 expt1_grayProcessing，查看是否执行了 bash 文件的 source。

通过编写的 expt1_grayProcessing.cpp 所生成的可执行文件来查看图片，以后图像处理的代码均可在 expt1_grayProcessing.cpp 中完成。测试过程中，可以不用将笔记本连接到机器人，需将代码中 capture.open(1)中的参数 1 改为 0，程序将读取笔记本自带的摄像头，参数如果依然为 1，则读取机器人上 USB 摄像头。

4.2 直方图均衡化算法实现（按照要求自己编写程序）

在 expt1_grayProcessing.cpp 中添加如下功能程序：

① 统计每个灰度下的像素个数

② 统计灰度频率，并绘制出直方图（可以参考附录代码）

③ 计算累计密度

④ 重新计算均衡化后的灰度值，四舍五入。参考公式：(N-1)*T+0.5

⑤ 直方图均衡化,更新原图每个点的像素值

4.3 硬件连接

① 打开机器人上的主电源

② 将机器人 USB 集线器上的 USB 接口接入实验用笔记本电脑。注意，如果只是把摄像头的 USB 接口接入笔记本电脑，将无法控制机器人运动。

在**终端**输入以下命令，对机器人进行测试：

```
$ roscore
$ roslaunch dashgo_driver driver.launch
```

打开另一个**终端**，运行以下命令，实现键盘控制机器人移动：

```
$ rosrun dashgo_tools teleop_twist_keyboard.py
```

再打开另一个**终端**，运行以下命令，读取机器人上的双目摄像头 ZED 的图像：

```
$ rosrun image_pkg expt1_grayProcessing
```

附录：expt1_grayProcessing.cpp

具体实现可以参考以下 github 代码

https://github.com/HITSZ-NRSL/HITSZ-AutoCourses/tree/master/digitalImageProcessing

main 代码如下：

```
#include <stdlib.h>
#include <iostream>
#include <string>
#include <cv.h>
#include <highgui.h>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include "ros/ros.h"
#define READIMAGE_ONLY
#ifndef READIMAGE_ONLY
#include <geometry_msgs/Twist.h>
#endif

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{
	ROS_WARN("*****START*****");
	ros::init(argc, argv, "trafficLaneTrack"); //初始化 ROS 节点
	ros::NodeHandle n;

	//Before the use of camera, you can test ur program with images first: imread()
	VideoCapture capture;
	capture.open(0); //打开 zed 相机，如果要打开笔记本上的摄像头，需要改为 0
	waitKey(100);
	if (!capture.isOpened())
	{
		printf("摄像头没有正常打开，重新插拔工控机上当摄像头\n");
		return 0;
	}

#ifndef READIMAGE_ONLY
	ros::Publisher pub = n.advertise<geometry_msgs::Twist>("/smoother_cmd_vel", 5); //定义 dashgo 机
器人的速度发布器
#endif
	Mat src_frame;
```

```
    while (ros::ok())
    {
        capture.read(src_frame);
        if (src_frame.empty())
        {
            break;
        }
        imshow("src", src_frame);
        //  此处为实验部分，请自行增加直方图均衡化的代码

#ifndef READIMAGE_ONLY
        //以下代码可设置机器人的速度值，从而控制机器人运动
        geometry_msgs::Twist cmd_red;
        cmd_red.linear.x = 0;
        cmd_red.linear.y = 0;
        cmd_red.linear.z = 0;
        cmd_red.angular.x = 0;
        cmd_red.angular.y = 0;
        cmd_red.angular.z = 0.2;
        pub.publish(cmd_red);
#endif
        ros::spinOnce();
        waitKey(5);
    }
    return 0;
}
```