

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 或 REALSENSE 相机图像的方法
- 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：Ubuntu20.04、ROS-noetic、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、摄像头设置方法（代码框架中的变量设置）

对于摄像头，可通过修改附录中给出的参考代码中的 `state` 变量来切换使用不同的摄像头：COMPUTER，ZED，REALSENSE。

若使用电脑摄像头，则将 `state` 变量修改为 COMPUTER，对应 `capture.open(0)`；

若使用 ZED 摄像头，则将 `state` 变量修改为 ZED，对应 `capture.open(4)`；（此处可能为 2、3、4，需做调整）；

若使用 RealSense 摄像头，将 `state` 变量修改 REALSENSE，对应

为在代码中订阅 ros image 话题/camera/color/image_raw;

具体可查看 demo 程序或者附录中的代码框架。

重要提示：机器人上的白色 USB 线，请插到笔记本右侧的 USB 口。

5、实验内容

5.1 直方图均衡化及控制机器人运动实验

(1) 算法实现

在 exp1.cpp 中添加如下功能程序：

- 1) 统计每个灰度下的像素个数并绘制出灰度直方图；
- 2) 计算图像灰度的概率密度分布；
- 3) 重新计算均衡化后的灰度值，四舍五入。参考公式： $(N-1)*T+0.5$ 。

直方图均衡化，更新原图每个点的像素值。

(2) 控制机器人运动

在 exp1.cpp 中添加代码，实现：查找到摄像头之后，让机器人动起来。可以参考 demo 程序（见附录），在给定的程序框架中添加让机器人原地旋转的代码。

(3) 实验步骤

- 1) 打开终端：ctrl+alt+t
- 2) 创建 dip_ws 工作空间：

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

- 3) 创建新的功能包、编译、添加环境变量

```
cd ~/dip_ws/src
catkin_create_pkg exp1
cd ..
catkin_make
source devel/setup.bash
```

- 4) 新建实验一的 cpp 文件

```
cd ~/dip_ws/src/exp1
mkdir src && cd src
```

```
gedit exp1.cpp
```

(在 exp1.cpp 中补充代码，可以参考附录给出的代码框架)

5) 配置 CMakeLists.txt 文件

```
cd ~/dip_ws/src/exp1
```

```
gedit CMakeLists.txt
```

修改文件如下：

```
cmake_minimum_required(VERSION 3.0.2)
project(exp1)
find_package(catkin REQUIRED COMPONENTS
  roscpp
  std_msgs
  geometry_msgs
  sensor_msgs
  cv_bridge
)
find_package(OpenCV REQUIRED)
catkin_package()
include_directories(
  ${catkin_INCLUDE_DIRS}
  ${OpenCV_INCLUDE_DIRS}
)
add_executable(exp1 src/exp1.cpp)
target_link_libraries(exp1
  ${catkin_LIBRARIES}
  ${OpenCV_LIBRARIES}
)
```

6) 在终端进行编译

```
cd ~/dip_ws
```

```
catkin_make
```

7) 测试（使用电脑摄像头）

打开一个终端

```
roscore
```

打开另外一个终端

```
source ~/dip_ws/devel/setup.bash  
roslaunch dashgo_driver driver_imu.launch
```

8) 使用机器人摄像头完成实验内容

A、使用 ZED 摄像头

打开一个新终端，启动小车底盘

```
roslaunch dashgo_driver driver_imu.launch
```

查看是否出现“publish odometry”消息等提示。

打开一个新终端

```
source ~/dip_ws/devel/setup.bash  
roslaunch exp1 exp1
```

B、使用 RealSense 摄像头

打开一个新终端，启动小车底盘

```
roslaunch dashgo_driver driver_imu.launch
```

查看是否出现“publish odometry”消息等提示。

打开一个新终端，运行 REALSENSE 官方功能包

```
roslaunch realsense2_camera rs_rgbd.launch
```

打开一个新终端

```
source ~/dip_ws/devel/setup.bash  
roslaunch exp1 exp1
```

具体实现可以参考以下 [github](https://github.com/HITSZ-NRSL/HITSZ-AutoCourses/tree/master/digitalImageProcessing) 代码

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

5.2 程序框架附录：exp1.cpp

```
#include <stdlib.h>
```

```
#include "opencv2/highgui/highgui.hpp"
```

```
#include <opencv2/opencv.hpp>
```

```
#include <opencv2/core/core.hpp>
```

```
#include "ros/ros.h"
```

```
#include "iostream"
```

```
#include <cv_bridge/cv_bridge.h>
```

```

#include "geometry_msgs/Twist.h"

enum CameraState
{
    COMPUTER = 0,
    ZED,
    REALSENSE
};
CameraState state = REALSENSE;

#define N 255 //灰度 level
using namespace std;
using namespace cv;

//getHistImage()--画图像直方图
Mat getHistImage( Mat hist)
{
    Scalar color(172, 172, 100); //划线颜色
    Scalar Background(255,255,255); //背景颜色
    int thickness = 2; //划线宽度
    int histss[256] = {0};

    /** 第一步：下面计算不同灰度值的像素分布 */

    int histSize = 500;
    Mat histImage(histSize, histSize, CV_8UC3, Background ); //绘制背景

    for (int h = 0; h < 256; h++) {

        /** 第二步：画出像素的直方图分布 */

    }
    return histImage;
}

Mat frame_msg;
void rcvCameraCallBack(const sensor_msgs::Image::ConstPtr& img)
{
    cv_bridge::CvImageConstPtr cv_ptr;
    cv_ptr = cv_bridge::toCvShare(img, sensor_msgs::image_encodings::BGR8);

```

```
    frame_msg = cv_ptr->image;
}
```

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, "exp1_node"); // 初始化 ROS 节点
    ros::NodeHandle n;
    ros::Subscriber camera_sub;
    VideoCapture capture;

    if(state == COMPUTER)
    {
        capture.open(0);
        if (!capture.isOpened())
        {
            printf("电脑摄像头没有正常打开\n");
            return 0;
        }
        waitKey(1000);
    }
    else if(state == ZED)
    {
        capture.open(4);
        if (!capture.isOpened())
        {
            printf("ZED 摄像头没有正常打开\n");
            return 0;
        }
        waitKey(1000);
    }
    else if(state == REALSENSE)
    {
        camera_sub = n.subscribe("/camera/color/image_raw",1,rcvCameraCallBack);
        waitKey(1000);
    }

    Mat frame;//当前帧图片

    int Grayscale[N];//灰度级
    int Grayscale2[N];//均衡化以后的灰度级
    float Gray_f[N];//频率
    int Gray_c[N];//累计密度
    ros::Rate loop_rate(10); // 设置循环频率为 10Hz
```

```

while (ros::ok())
{
if(state == COMPUTER)
{
capture.read(frame);
if (frame.empty())
{
printf("没有获取到电脑图像\n");
continue;
}
}
else if(state == ZED)
{
capture.read(frame);
if (frame.empty())
{
printf("没有获取到 ZED 图像\n");
continue;
}
frame = frame(cv::Rect(0,0,frame.cols/2,frame.rows)); //截取 zed 的左目图片
}
else if(state == REALSENSE)
{
if(frame_msg.cols == 0)
{
printf("没有获取到 realsense 图像\n");
ros::spinOnce();
continue;
}

frame = frame_msg;
}
}

```

```

Mat frIn = frame;
Mat New;
cvtColor(frIn,frIn,COLOR_RGB2GRAY,0);

```

/** 第三步：直方图均衡化处理 */

```

Mat last = getHistImage(New);
Mat origi= getHistImage(frIn);
imshow("his",last); //均衡化后直方图

```

```

imshow("origi",origi);//原直方图
imshow("Histed",New);//均衡化后图像
imshow("Origin",frIn);//原图像

```

```

/** 第四步：可以参考 demo 程序，添加让机器人原地旋转代码 */
geometry_msgs::Twist vel_msg;

```

```

ros::spinOnce(); // 处理回调函数
waitKey(5);
loop_rate.sleep(); // 控制循环速率

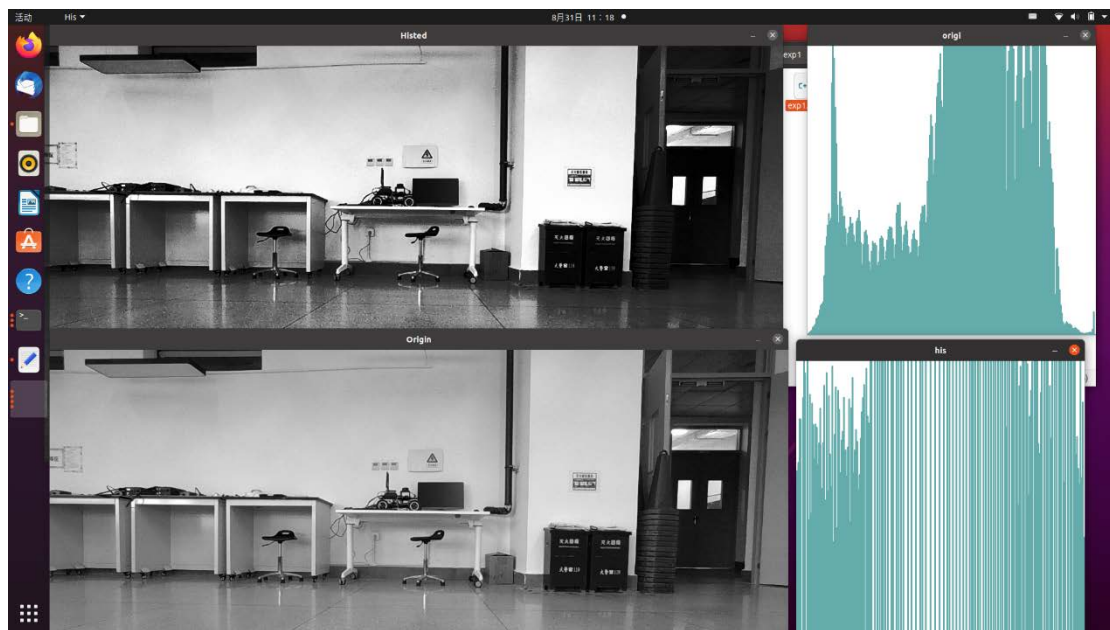
```

```

}
return 0;
}

```

参考运行程序结果：



5.3 demo 程序附录：

CMakeLists.txt

```

cmake_minimum_required(VERSION 3.0.2)
project(car_demo) # 对应功能包名字

find_package(catkin REQUIRED COMPONENTS
  roscpp
  std_msgs
  geometry_msgs

```

```

    sensor_msgs
    cv_bridge
)

find_package(OpenCV REQUIRED)

catkin_package()

include_directories(
    ${catkin_INCLUDE_DIRS}
    ${OpenCV_INCLUDE_DIRS}
)

add_executable(car_demo src/car_demo.cpp)

target_link_libraries(car_demo
    ${catkin_LIBRARIES}
    ${OpenCV_LIBRARIES}
)

```

car_demo.cpp

```

#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include <opencv2/opencv.hpp>
#include <cv_bridge/cv_bridge.h>

ros::Publisher vel_pub;

using namespace cv;

enum CameraState
{
    COMPUTER = 0,
    ZED,
    REALSENSE
};

CameraState state = REALSENSE;

Mat frame_msg;
void rcvCameraCallBack(const sensor_msgs::Image::ConstPtr& img)
{
    cv_bridge::CvImageConstPtr cv_ptr;
    cv_ptr = cv_bridge::toCvShare(img, sensor_msgs::image_encodings::BGR8);
    frame_msg = cv_ptr->image;
}

```

```

}

int main(int argc, char ** argv)
{
    ros::init(argc, argv, "car_demo");
    ros::NodeHandle n;
    ros::Subscriber camera_sub;
    VideoCapture capture;

    vel_pub = n.advertise<geometry_msgs::Twist>("/cmd_vel", 10);

    if(state == COMPUTER)
    {
        capture.open(0);
        if (!capture.isOpened())
        {
            printf("电脑摄像头没有正常打开\n");
            return 0;
        }
    }

```

```

waitKey(1000);
    }
    else if(state == ZED)
    {
        capture.open(4);
        if (!capture.isOpened())
        {
            printf("ZED 摄像头没有正常打开\n");
            return 0;
        }
        waitKey(1000);
    }
    else if(state == REALSENSE)
    {
        camera_sub = n.subscribe("/camera/color/image_raw",1,rcvCameraCallBack);
        waitKey(1000);
    }

    Mat frame;//当前帧图片
    ros::Rate loop_rate(10); // 设置循环频率为 10Hz
    geometry_msgs::Twist vel_msg;

    while (ros::ok())
    {

```

```

if(state == COMPUTER)
{
    capture.read(frame);
    if (frame.empty())
    {
        printf("没有获取到电脑图像\n");
        continue;
    }
}
else if(state == ZED)
{
    capture.read(frame);
    if (frame.empty())
    {
        printf("没有获取到 ZED 图像\n");
        continue;
    }
    frame = frame(cv::Rect(0,0,frame.cols/2,frame.rows)); //截取 zed 的左目图片

```

```

}
else if(state == REALSENSE)
{
    if(frame_msg.cols == 0)
    {
        printf("没有获取到 realsense 图像\n");
        ros::spinOnce();
        continue;
    }
    frame = frame_msg;
}

// 发布消息，让小车绕一小半径旋转
vel_msg.linear.x = 0.05;
vel_msg.linear.y = 0.;
vel_msg.angular.z = 0.5;
vel_pub.publish(vel_msg);

imshow("frame",frame);

ros::spinOnce(); // 处理回调函数
waitKey(5);
loop_rate.sleep(); // 控制循环速率

```

	<div data-bbox="263 185 287 235" data-label="Text">}</div> <div data-bbox="231 280 255 318" data-label="Text">}</div>
--	---