使用摄像头识别颜色让机器人去执行一个动作

1、在 package 包的 src 目录下新建 color_test.cpp 文件

代码开头导入我们要用到的头文件

注意 opencv 的引入需要在 CMakelists.txt 里面单独添加：

find_package

find_package(OpenCV REQUIRED)

指定编译使用了 opencv 库的 cpp 文件时，也要添加 opencv 的库链接

比如：

```
add_executable  (color_test  src/color_test.cpp)
target_link_libraries  (color_test  ${catkin_LIBRARIES} ${OpenCV_LIBS})
```

部分参考代码：

```
#include "ros/ros.h"
#include <iostream>
#include <opencv2/opencv.hpp>
#include "geometry_msgs/Twist.h"
#include <boost/thread.hpp>
#include <csignal>  //导入信号库，程序中用来监听 ctrl+c 信号

using namespace std;
using namespace cv;
ros::Publisher cmd_pub;  //速度发布器
int colorType = 0;  //摄像头看到的颜色类型，  1 红 2 绿 3 黄 4 蓝
```

2、实现方法举例

可以将图片转换成 HSV 颜色模式，再计算图片各像素点是什么颜色，来计算不同颜色在图片中所占百分比，也可以用其他方法。

OpenCV 中 HSV 颜色模型及颜色分量范围，可查看如下网址：

https://www.cnblogs.com/wangyblzu/p/5710715.html

参考代码：

```
void hsv_percentage(Mat hsv_image) {
  float red_per = getRedPer(); //计算红色的百分比,自行实现
  float green_per = getGreenPer();  //计算绿色的百分比,自行实现
  float yellow_per = getYellowPer();    //计算黄色的百分比,自行实现
  float blue_per = getBluePer();    //计算蓝色的百分比,自行实现
  ROS_WARN("%f,%f,%f,%f",red_per, green_per, yellow_per, blue_per);  //打印颜色百分比，可以注释掉
  if (red_per > 0.2)
```

```
    {
        colorType = 1;
    }
    else if (green_per > 0.2)
    {
        colorType = 2;
    }
    else if (yellow_per > 0.2)
    {
        colorType = 3;
    }
    else if (blue_per > 0.2)
    {
        colorType = 4;
    }
    else
    {
        colorType = 0;
    }
}
//解析摄像头数据
void analyseColor()
{
    VideoCapture capture;
    capture.open(1);//打开  zed  相机
    if (!capture.isOpened())
    {
        printf("摄像头没有正常打开，请检查摄像头接线\n");
        return ;
    }
    Mat frame;//当前帧图片
    while (ros::ok())
    {
        capture.read(frame);
        if(frame.empty())
        {
            ROS_WARN("摄像头数据为空");
            break;
        }
        Mat hsv_image;
```

```
        cvtColor(frame, hsv_image, CV_BGR2HSV);
        hsv_percentage(hsv_image);
    }
}
//根据颜色类型，向底盘发送不同的消息
//红退绿进，黄左蓝右，请注意不要在红绿颜色服装同学旁边调试...
void sendCmd()
{
    ros::Rate loop_rate(5);    //频率为 5，每 200 毫秒发送一次速度
    geometry_msgs::Twist twist;
    twist.linear.y = 0;
    twist.linear.z = 0;
    twist.angular.x = 0;
    twist.angular.y = 0;
    //上面这 4 个值常为 0,修改线速度就修改 linear.x，修改角速度就修改 angular.z
    while (1) {
        switch(colorType){
            case 1:
                twist.linear.x = -0.1;    //线速度
                twist.angular.z = 0;    //角速度
                break;
            case 2:
                twist.linear.x = 0.1;
                twist.angular.z = 0;
                break;
            case 3:
                twist.linear.x = 0;
                twist.angular.z = 0.1;
                break;
            case 4:
                twist.linear.x = 0;
                twist.angular.z = -0.1;
                break;
        }
        if(colorType != 0) {
            cmd_pub.publish(twist);    //发送速度
        }
        loop_rate.sleep();        //根据频率进行延时
    }
```

```
}
//监听到 ctrl+c，退出程序
void sig_handler (int sig)
{
  if (sig == SIGINT)
  {
    exit(0); //关闭程序
  }
}
int main(int argc, char** argv)
{
  ros::init(argc , argv , "color_test");  //进行节点初始化
  ros::NodeHandle nh;  //创建节点句柄
  cmd_pub = nh.advertise<geometry_msgs::Twist>("cmd_vel", 10);  //创建速度发布器
  signal( SIGINT, sig_handler );  //由于开了线程，且线程里面写了无限循环，所以无法响应默认的 ctrl+c 关闭事件，需要自行监听处理。
  boost::thread analyseColor_thread(&analyseColor);  //新建一个线程，一直读摄像头数据并分析颜色
  boost::thread sendCmd_thread(&sendCmd);  //新建一个线程，一直判断颜色值做相应的动作
  analyseColor_thread.join();  //启动线程
  sendCmd_thread.join();  //启动线程
}
```

3、启动小车方法

（1）打开一个终端，先启动小车的驱动程序 roslaunch dashgo_driver driver_imu.launch

（2）再打开一个终端，启动上述识别颜色程序