ROS机械臂开发：从入门到实战

—— 第8讲：ROS机器视觉应用中的关键点

主讲人 胡春旭

机器人博客“古月居”博主
《ROS机器人开发实践》作者
武汉精锋微控科技有限公司 联合创始人
华中科技大学 自动化学院 硕士

目录

1. ROS图像接口

1. ROS图像接口

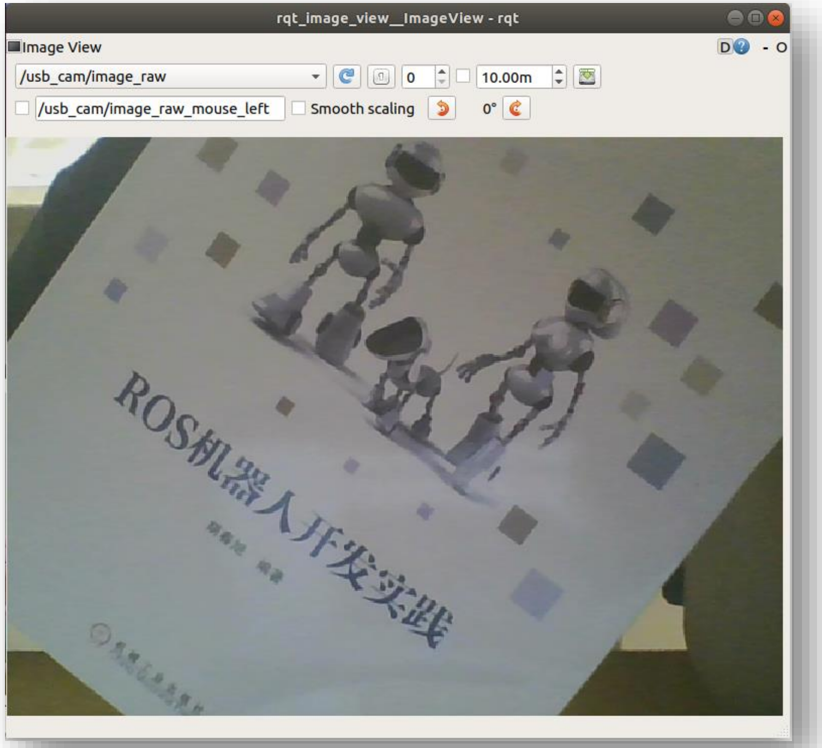
古月居 GYH.AI

摄像头驱动

$ sudo apt-get install ros-melodic-usb-cam

$ roslaunch probot_vision usb_cam.launch

$ rqt_image_view



usb_cam功能包中的话题

	名称	类型	描述
Topic发布	~<camera_name>/image	sensor_msgs/Image	发布图像数据

usb_cam功能包中的参数

参数	类型	默认值	描述
~video_device	string	"/dev/video0"	摄像头设备号
~image_width	int	640	图像横向分辨率
~image_height	int	480	图像纵向分辨率
~pixel_format	string	"mjpeg"	像素编码，可选值：mjpeg，yuyv，uyvy
~io_method	string	"mmap"	IO通道，可选值：mmap，read，userptr
~camera_frame_id	string	"head_camera"	摄像头坐标系
~framerate	int	30	帧率
~brightness	int	32	亮度，0~255
~saturation	int	32	饱和度，0~255
~contrast	int	32	对比度，0~255
~sharpness	int	22	清晰度，0~255
~autofocus	bool	false	自动对焦
~focus	int	51	焦点（非自动对焦状态下有效）
~camera_info_url	string	-	摄像头校准文件路径
~camera_name	string	"head_camera"	摄像头名称

1. ROS图像接口

- Header：消息头，包含消息序号，时间戳和绑定坐标系；
- height：图像的纵向分辨率；
- width：图像的横向分辨率；
- encoding：图像的编码格式，包含RGB、YUV等常用格式，不涉及图像压缩编码；
- is_bigendian：图像数据的大小端存储模式；
- step：一行图像数据的字节数量，作为数据的步长参数；
- data：存储图像数据的数组，大小为step*height个字节

```
➜  ~ rosmsg show sensor_msgs/Image
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

1080*720分辨率的摄像头产生一帧图像的数据大小是：3*1080*720=2764800字节，即2.7648MB

古月居 GYH.AI

压缩图像消息

```
➜  ~ rosmsg show sensor_msgs/CompressedImage
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string format
uint8[] data
```

- format：图像的压缩编码格式（jpeg、png、bmp）
- data：存储图像数据数组

1. ROS图像接口

➢ 安装SDK（https://github.com/intel-ros/realsense/releases）

$ mkdir build

$ cd build

$ cmake ..

$ make

$ sudo make install

Intel RealSense

➢ 安装ROS驱动（https://github.com/IntelRealSense/librealsense/releases）

$ catkin_make -DCATKIN_ENABLE_TESTING=False -DCMAKE_BUILD_TYPE=Release

$ catkin_make install

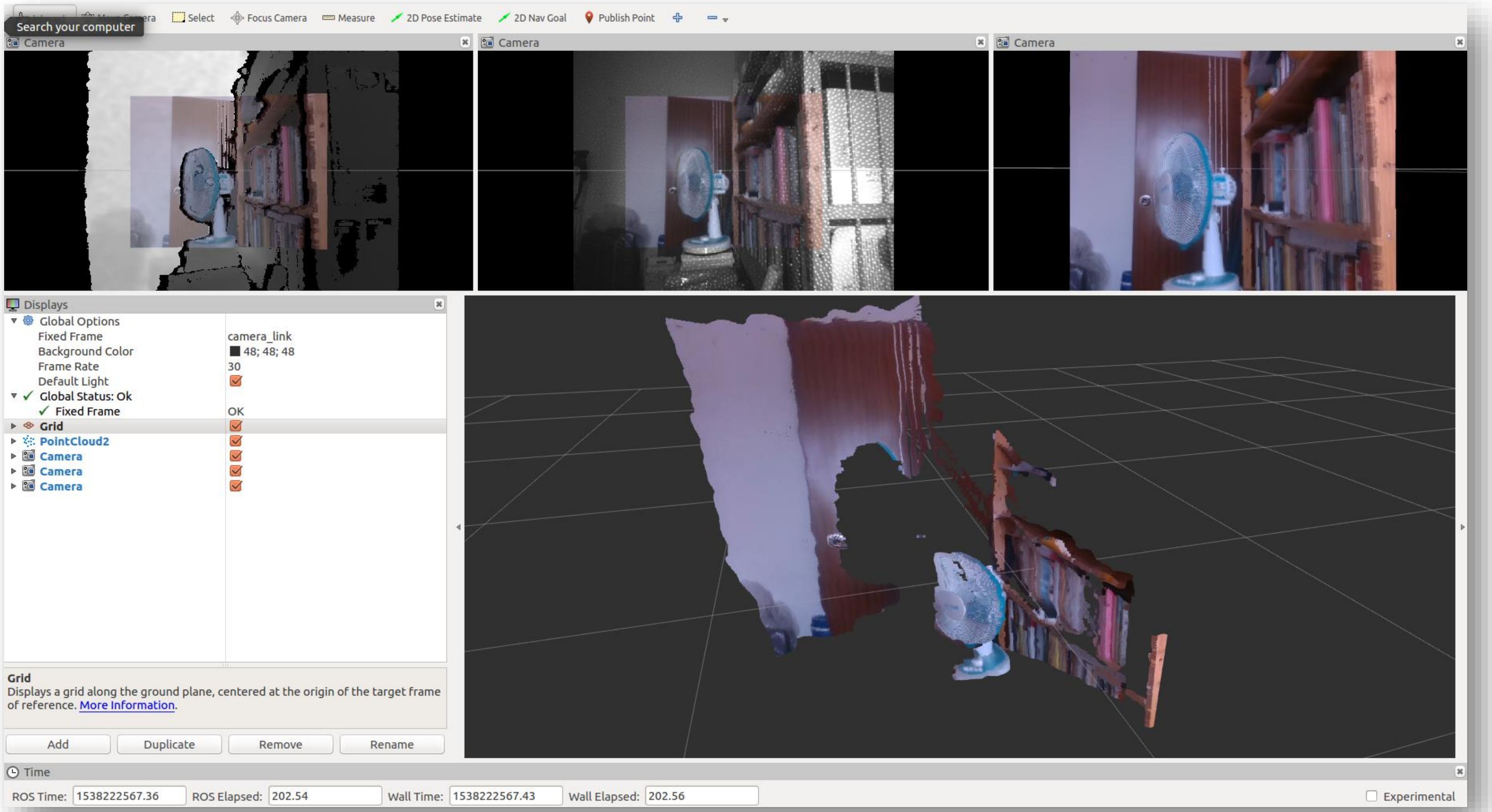
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrcsource ~/.bashrc

参考链接：

https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md

https://blog.csdn.net/u012926144/article/details/80761342

1. ROS图像接口



点云显示

```
$ roslaunch realsense2_camera rs_rgbd.launch
$ rosrun rviz rviz
```

1. ROS图像接口

- height：点云图像的纵向分辨率；
- width：点云图像的横向分辨率；
- fields：每个点的数据类型；
- is_bigendian：数据的大小端存储模式；
- point_step：单点的数据字节步长；
- row_step：一列数据的字节步长；
- data：点云数据的存储数组，总字节大小为 row_step* height；
- is_dense：是否有无效点。

```
➜  ~ rosmsg show sensor_msgs/PointCloud2
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
sensor_msgs/PointField[] fields
  uint8 INT8=1
  uint8 UINT8=2
  uint8 INT16=3
  uint8 UINT16=4
  uint8 INT32=5
  uint8 UINT32=6
  uint8 FLOAT32=7
  uint8 FLOAT64=8
  string name
  uint32 offset
  uint8 datatype
  uint32 count
bool is_bigendian
uint32 point_step
uint32 row_step
uint8[] data
bool is_dense
```

点云单帧数据量也很大，如果使用分布式网络传输，需要考虑能否满足数据的传输要求，或者针对数据进行压缩。

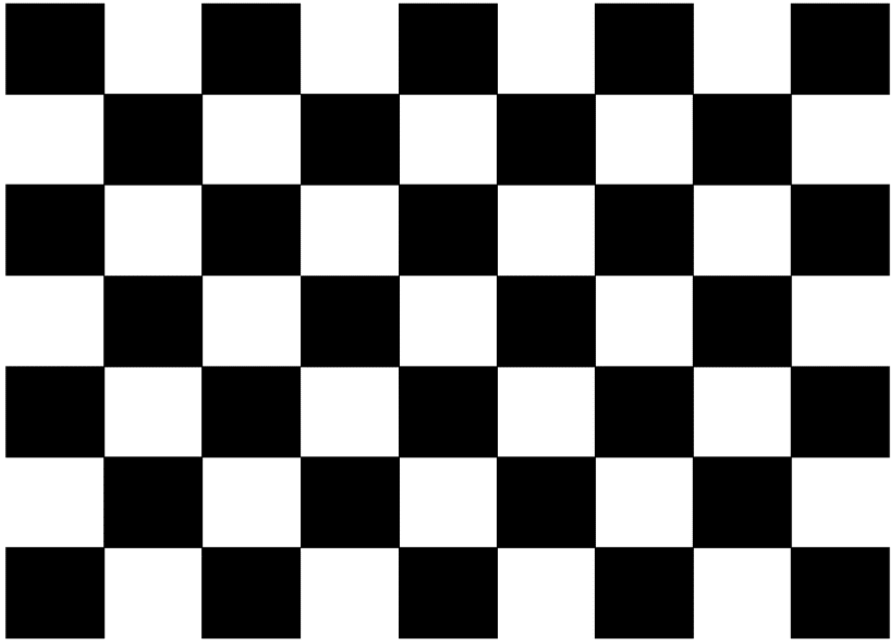
2. 摄像头内参标定

➢ 摄像头为什么要标定?

摄像头这种精密仪器对光学器件的要求较高，由于摄像头内部与外部的一些原因，生成的物体图像往往会发生畸变，为避免数据源造成的误差，需要针对摄像头的参数进行标定。

安装标定功能包

```
$ sudo apt-get install ros-melodic-camera-calibration
```



棋盘格标定靶

古月居 GYH.AI

摄像头标定流程

➢ 启动摄像头

$ roslaunch probot_vision usb_cam.launch

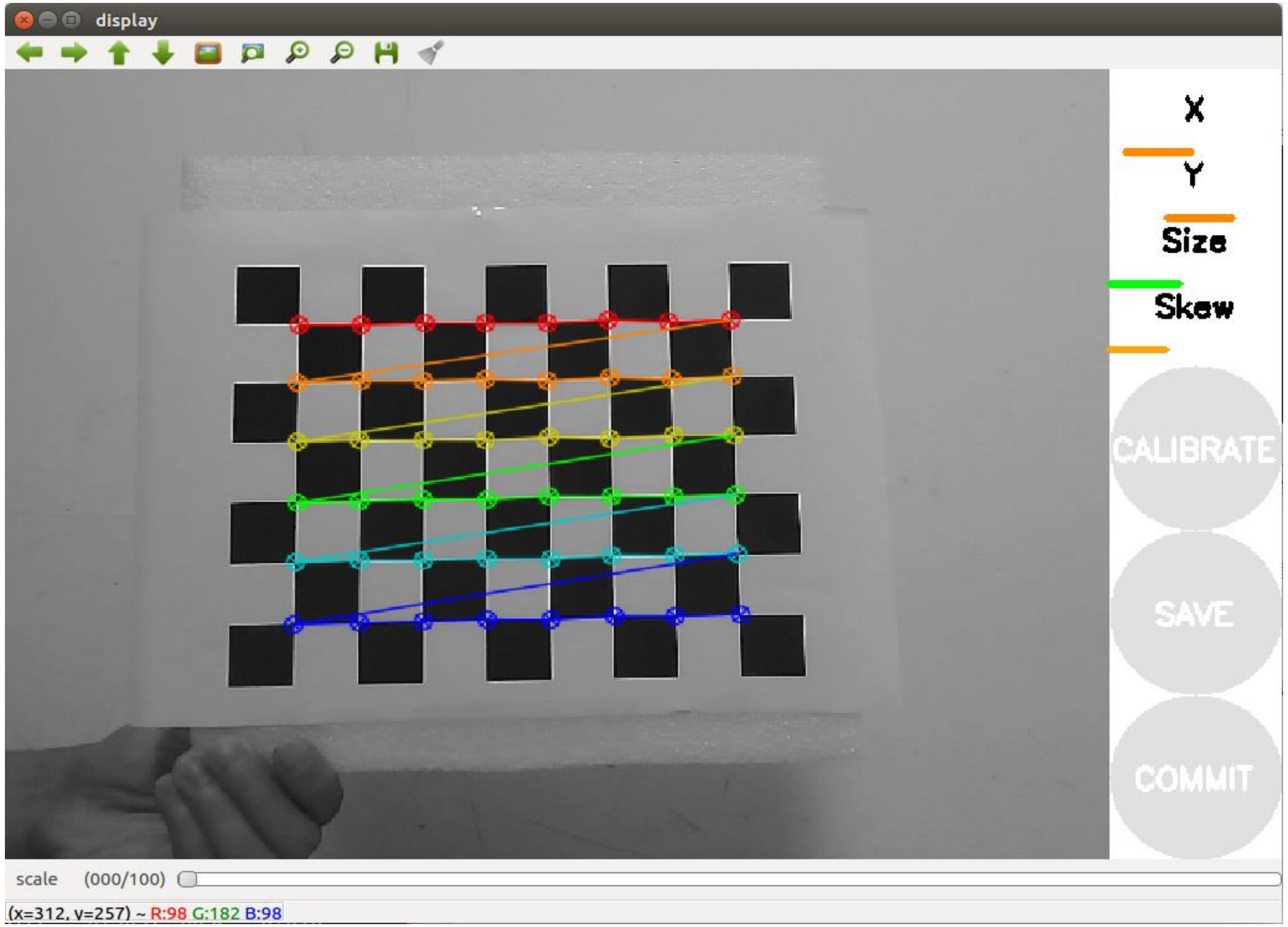
➢ 启动标定包

$ rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.024 image:=/usb_cam/image_raw camera:=/usb_cam

1. size：标定棋盘格的内部角点个数，这里使用的棋盘一共有六行，每行有8个内部角点；
2. square：这个参数对应每个棋盘格的边长，单位是米；
3. image和camera：设置摄像头发布的图像话题。

2. 摄像头内参标定

- X：标定靶在摄像头视野中的左右移动；
- Y：标定靶在摄像头视野中的上下移动；
- Size：标定靶在摄像头视野中的前后移动；
- Skew：标定靶在摄像头视野中的倾斜转动。



标定过程

摄像头如何使用标定文件?

```xml
<launch>

    <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
        <param name="video_device" value="/dev/video0" />
        <param name="image_width" value="1280" />
        <param name="image_height" value="720" />
        <param name="pixel_format" value="yuyv" />
        <param name="camera_frame_id" value="usb_cam" />
        <param name="io_method" value="mmap"/>

        <param name="camera_info_url" type="string" value="file://$(find probot_vision)/camera_calibration.yaml" />
    </node>

</launch>
```

probot_vision/launch/usb_cam_with_calibration.launch

3. ROS+OpenCV物体识别

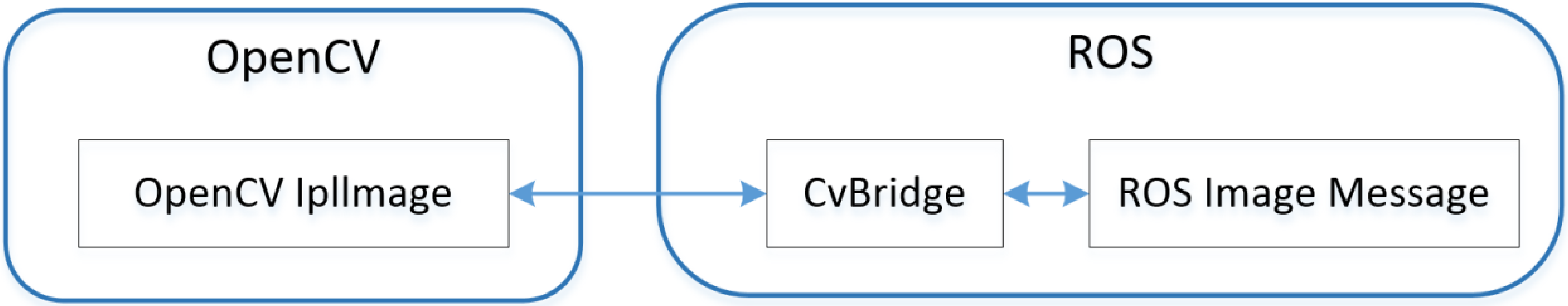
3. ROS+OpenCV物体识别

OpenCV是什么?

- Open Source Computer Vision Library;
- 基于BSD许可发行的跨平台开源计算机视觉库（Linux、Windows和Mac OS等）；
- 由一系列C函数和少量C++类构成，同时提供C++、Python、Ruby、MATLAB等语言的接口;
- 实现了图像处理和计算机视觉方面的很多通用算法，而且对非商业应用和商业应用都是免费的;
- 可以直接访问硬件摄像头，并且还提供了一个简单的 GUI系统——highgui。

安装OpenCV

```
$ sudo apt-get install ros-melodic-vision-opencv libopencv-dev python-opencv
```

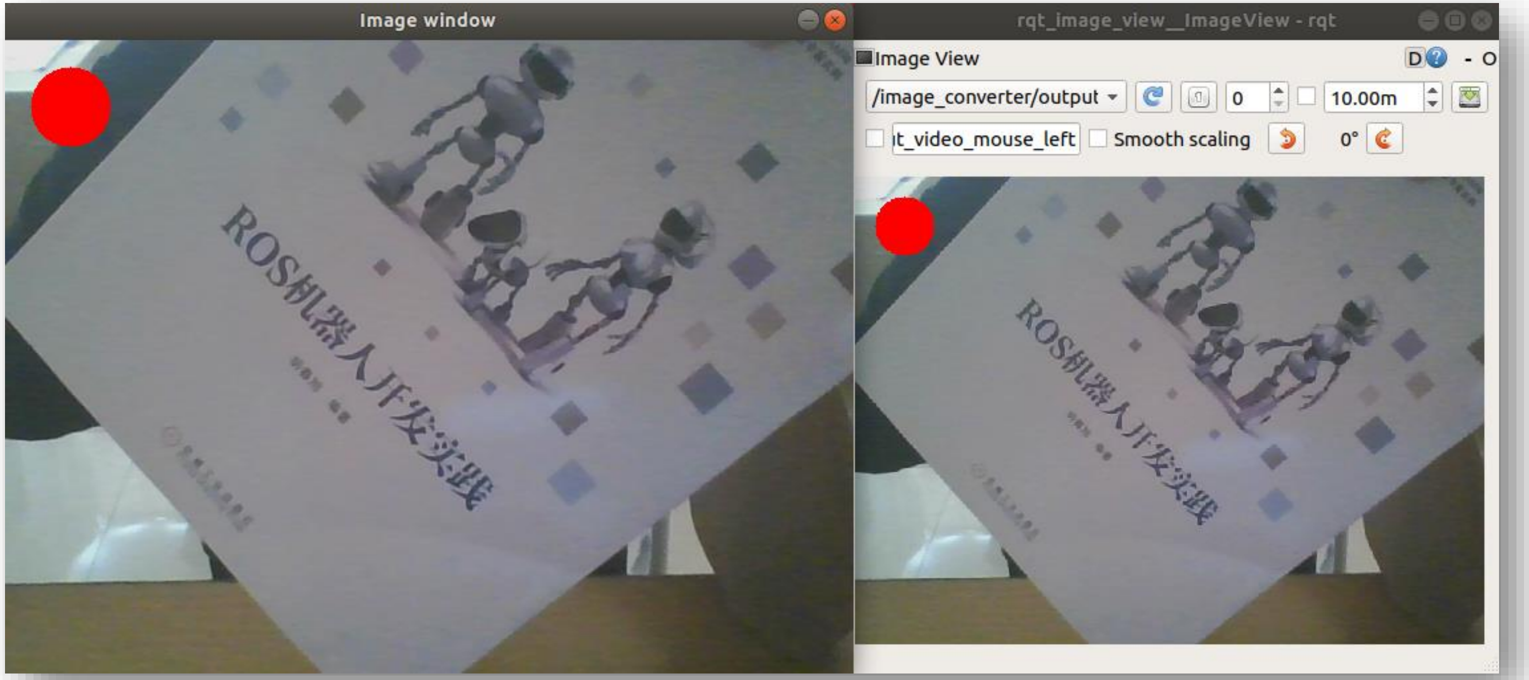


ROS与OpenCV的集成框架

3. ROS+OpenCV物体识别

测试例程

```
$ roslaunch probot_vision usb_cam.launch
$ rosrun probot_vision image_converter
$ rqt_image_view
```



3. ROS+OpenCV物体识别

```
ImageConverter()
  : it_(nh_)
{
  // Subscrive to input video feed and publish output video feed
  image_sub_ = it_.subscribe("/usb_cam/image_raw", 1, &ImageConverter::imageCb, this);
  image_pub_ = it_.advertise("/image_converter/output_video", 1);

  cv::namedWindow(OPENCV_WINDOW);
}

~ImageConverter()
{
  cv::destroyWindow(OPENCV_WINDOW);
}

void imageCb(const sensor_msgs::ImageConstPtr& msg)
{
  cv_bridge::CvImagePtr cv_ptr;
  try
  {
    cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
  }
  catch (cv_bridge::Exception& e)
  {
    ROS_ERROR("cv_bridge exception: %s", e.what());
    return;
  }

  // Draw an example circle on the video stream
  if (cv_ptr->image.rows > 60 && cv_ptr->image.cols > 60)
    cv::circle(cv_ptr->image, cv::Point(50, 50), 10, CV_RGB(255,0,0));

  // Update GUI Window
  cv::imshow(OPENCV_WINDOW, cv_ptr->image);
  cv::waitKey(3);

  // Output modified video stream
  image_pub_.publish(cv_ptr->toImageMsg());
}
```

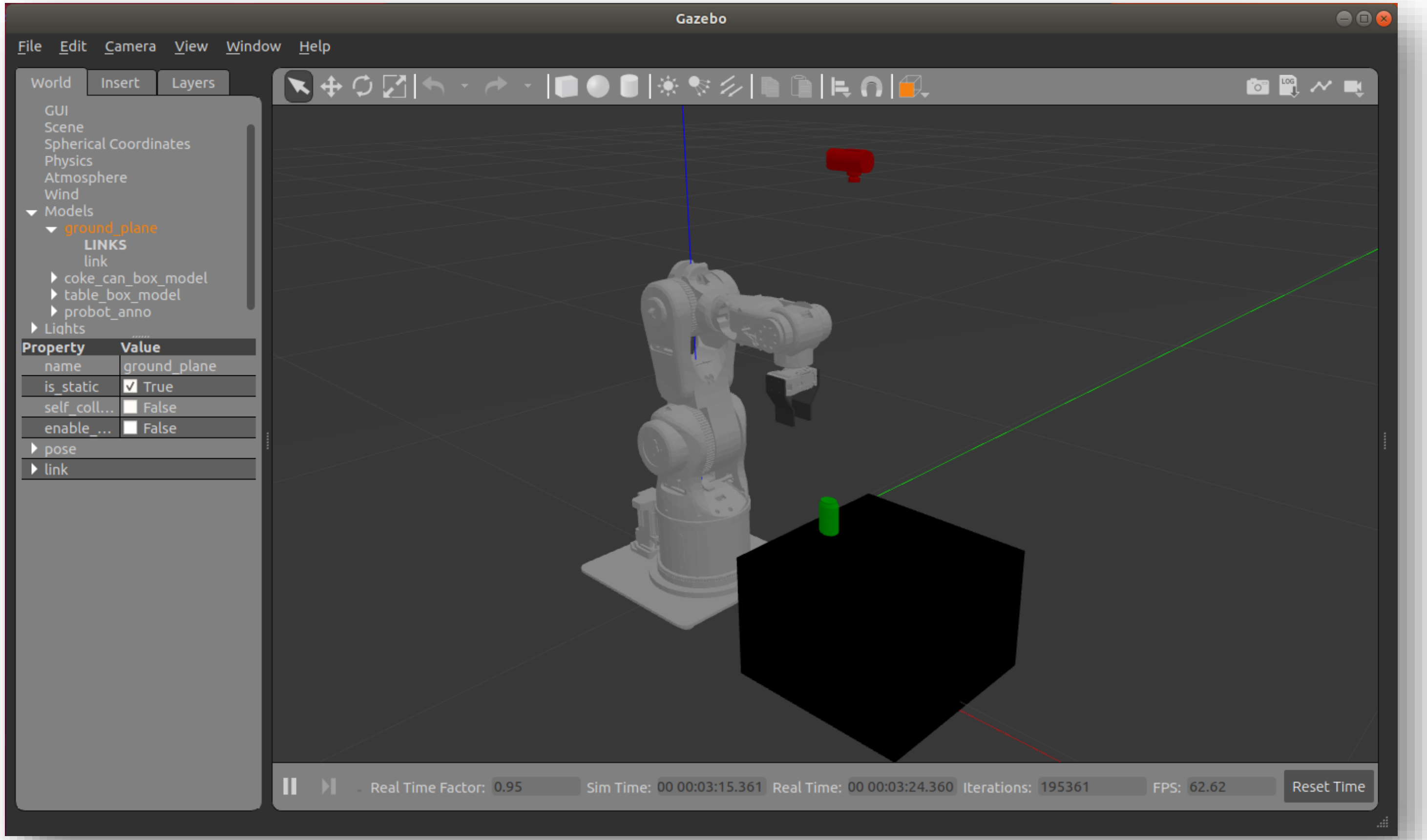
创建订阅者与发布者

转换成OpenCV图像数据

OpenCV图像处理

转换成ROS图像消息

3. ROS+OpenCV物体识别

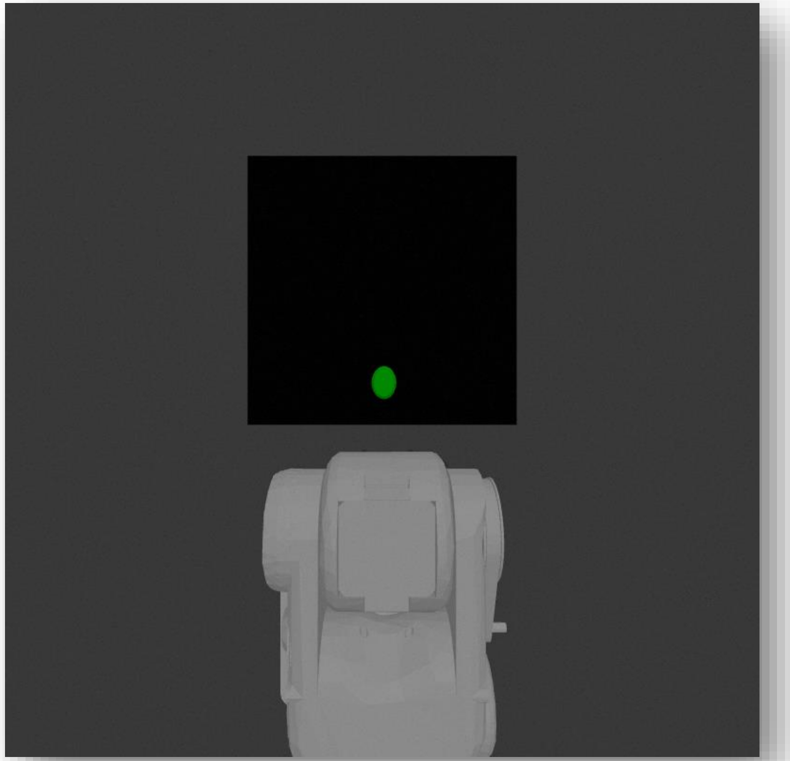


物体识别例程

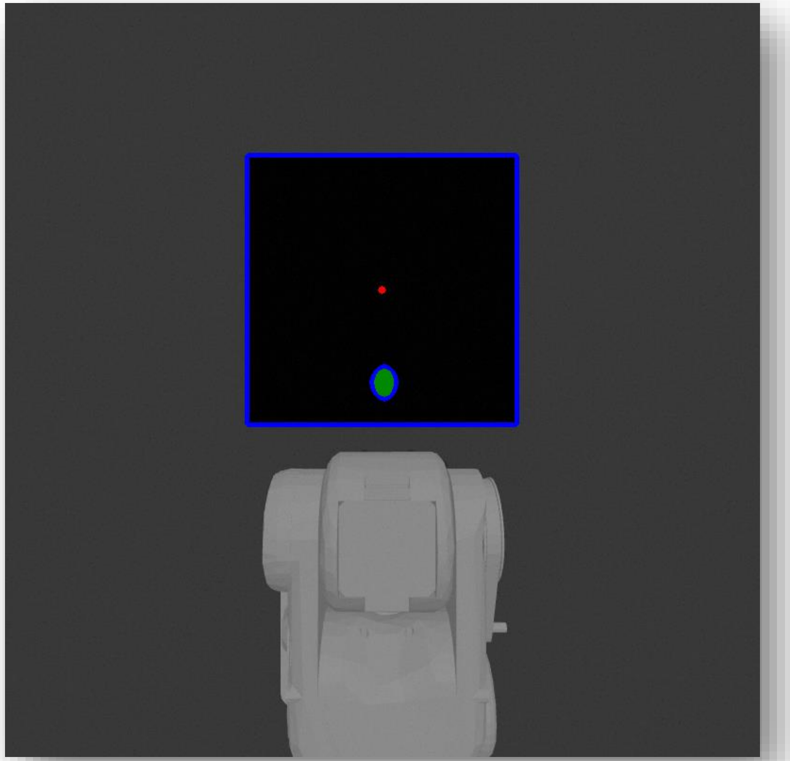
$ roslaunch probot_gazebo probot_anno_with_gripper_gazebo_world.launch

$ rosrun probot_vision vision_manager

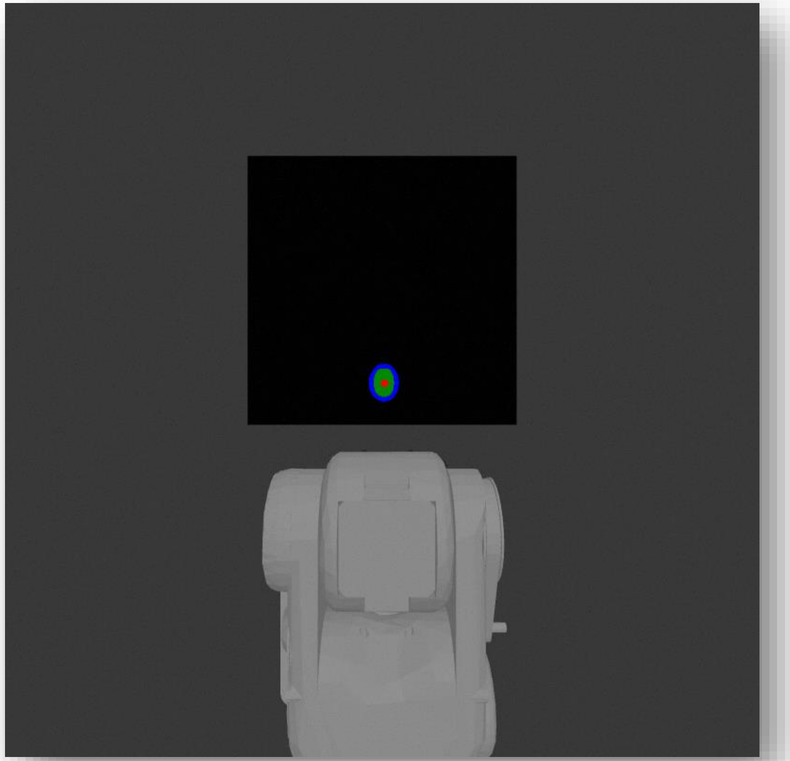
3. ROS+OpenCV物体识别



原始图像



桌面识别



物体识别

3. ROS+OpenCV物体识别

```
void VisionManager::detectTable(const sensor_msgs::ImageConstPtr &msg, cv::Rect &tablePos)
{
    // Extract Table from the image and assign values to pixel_per_mm fields
    cv::Mat BGR[3];

    try
    {
      cv_ptr_ = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
    }
    catch (cv_bridge::Exception &e)
    {
      ROS_ERROR("cv_bridge exception: %s", e.what());
      return;
    }

    cv::Mat &image = cv_ptr_->image;

    split(image, BGR);
    cv::Mat gray_image_red = BGR[2];
    cv::Mat gray_image_green = BGR[1];
    cv::Mat denoiseImage;
    cv::medianBlur(gray_image_red, denoiseImage, 3);

    // Threshold the Image
    cv::Mat binaryImage = denoiseImage;
    for (int i = 0; i < binaryImage.rows; i++)
    {
        for (int j = 0; j < binaryImage.cols; j++)
        {
            int editValue = binaryImage.at<uchar>(i, j);
            int editValue2 = gray_image_green.at<uchar>(i, j);

            if ((editValue >= 0) && (editValue < 20) && (editValue2 >= 0) && (editValue2 < 20))
            { // check whether value is within range.
                binaryImage.at<uchar>(i, j) = 255;
            }
            else
            {
                binaryImage.at<uchar>(i, j) = 0;
            }
        }
    }
```

转换成OpenCV图像数据

图像处理

3. ROS+OpenCV物体识别

```cpp
dilate(binaryImage, binaryImage, cv::Mat());

// Get the centroid of the of the blob
std::vector<cv::Point> nonZeroPoints;
cv::findNonZero(binaryImage, nonZeroPoints);
cv::Rect bbox = cv::boundingRect(nonZeroPoints);
cv::Point pt;
pt.x = bbox.x + bbox.width / 2;
pt.y = bbox.y + bbox.height / 2;
cv::circle(image, pt, 2, cv::Scalar(0, 0, 255), -1, 8);

// Update pixels_per_mm fields
pixels_permm_y = bbox.height / table_length;
pixels_permm_x = bbox.width  / table_breadth;

tablePos = bbox;

// Test the conversion values
std::cout << "Pixels in y" << pixels_permm_y << std::endl;
std::cout << "Pixels in x" << pixels_permm_x << std::endl;

// Draw Contours - For Debugging
std::vector<std::vector<cv::Point>> contours;
std::vector<cv::Vec4i> hierarchy;

cv::findContours(binaryImage, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, cv::Point(0, 0));
for (int i = 0; i < contours.size(); i++)
{
    cv::Scalar color = cv::Scalar(255, 0, 0);
    cv::drawContours(image, contours, i, color, 1, 8, hierarchy, 0, cv::Point());
}
```

图像处理

```cpp
// Output modified video stream
image1_pub_.publish(cv_ptr_->toImageMsg());
}
```

转换成ROS图像消息

本讲小结

ROS图像接口

- 二维图像：sensor_msgs/Image、sensor_msgs/CompressedImage
- 三维图像： sensor_msgs/PointCloud2

摄像头内参标定

- 避免摄像头内部光学器件造成的数据源畸变误差
- ros-melodic-camera-calibration

ROS+OpenCV物体识别

- CvBridge：转换ROS与OpenCV之间的图像数据
- 物体识别流程：
  - ROS驱动摄像头，发布图像消息
  - 将ROS图像消息转换成OpenCV图像数据
  - OpenCV图像处理
  - OpenCV图像转换成ROS消息

本讲作业

1. 安装ROS摄像头驱动功能包，驱动笔记本摄像头，并使用ROS可视化工具显示图像；
2. 在摄像头采集到的图像中，识别某一白色物体，将识别到的物体边缘绘制出来，并在ROS可视化工具中显示识别结果（识别物体和背景色有较大区别）。

扩展阅读

- usb_cam
http://wiki.ros.org/usb_cam

- sensor_msgs
http://wiki.ros.org/sensor_msgs

- cv_bridge Tutorials
http://wiki.ros.org/cv_bridge/Tutorials

- OpenCV Tutorials
https://docs.opencv.org/master/d9/df8/tutorial_root.html

- 《ROS机器人开发实践》，第7章

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭