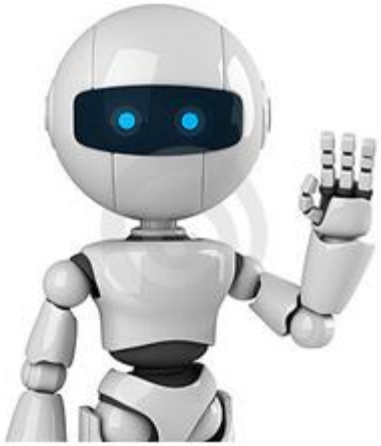
5. 机器人视觉系统



东北大学 张云洲

内容提纲

基本知识

1. 常用的开源库
2. 视觉传感器
3. ROS IDE
4. ROS cv_bridge软件包
5. 相机针孔模型

实验部分

1. 实验一：网络摄像头的使用
2. 实验二：cv_bridge程序包的使用
3. 实验三：在ROS包中使用pcl库

OpenCV、OpenNI、PCL

ROS机器人视觉系统三大支柱——

- OpenCV

  2D图像处理和机器学习；

- OpenNI

  提供深度相机（Kinect和Xtion）的驱动以及“Natural Interaction”库，实现骨架追踪。

- PCL

  Point Cloud Library，支持3D点云处理。

*Ecto：Willow Garage新推出的视觉框架，允许通过一个接口同时使用OpenCV和PCL。

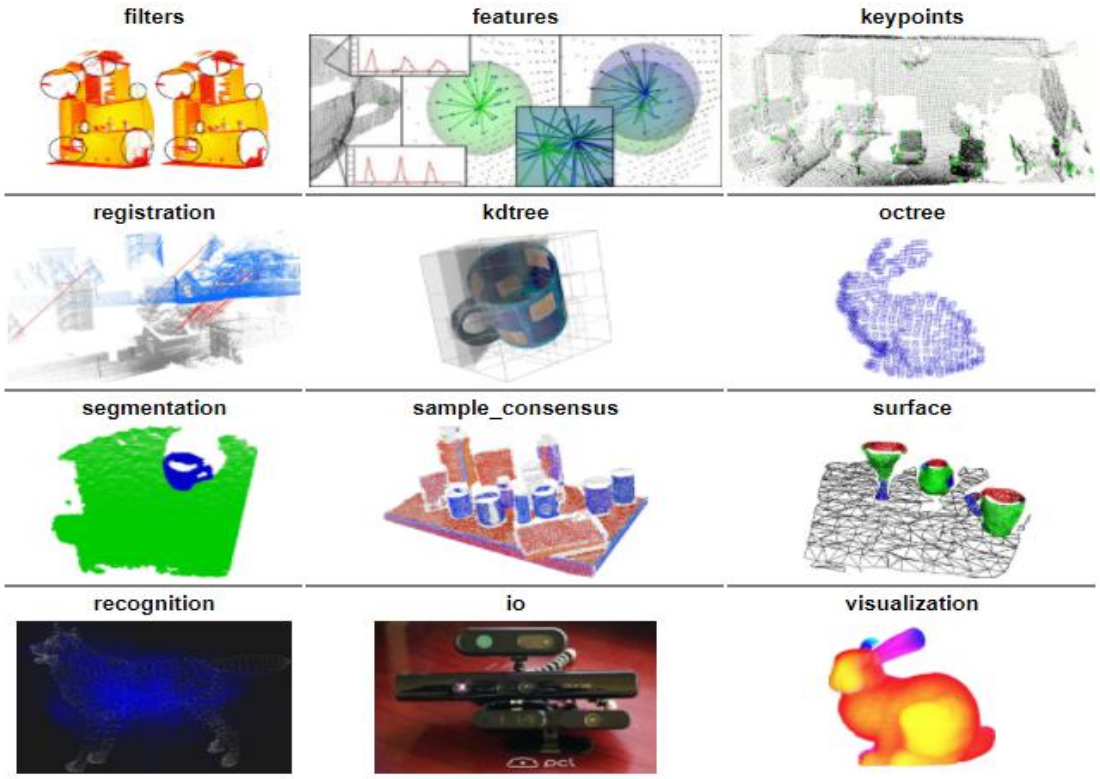
常用开源库

OpenCV是Intel 开源计算机视觉库，包含了超过2500种计算机视觉和机器学习算法。这些算法可以被应用在人脸识别、物体检测、运动跟踪、图像拼接、相似图像检索等等方面。广泛地被公司、研究机构等使用。具有C++、C、Python和java接口，支持Windows，Linux，Mac OS，iOS 和Android平台。

- core
- imgproc
- imgcodec
- highgui
- video
- videoio
- videostab
- shape
- softcascades
- calib3d
- feature2d
- nonfree
- objdetect
- ml
- flann
- photo
- stitching
- superres
- viz

常用开源库

PCL(**P**oint **C**loud **L**ibrary)是2D/3D图像和点云处理的大规模开源项目。由C++编写，包含滤波、特征提取、表面重建、对齐、分割等算法。跨平台，支持Linux、MacOS、Windows，以及Android和IOS。



视觉传感器

罗技网络摄像头

Bumblebee2 双目相机

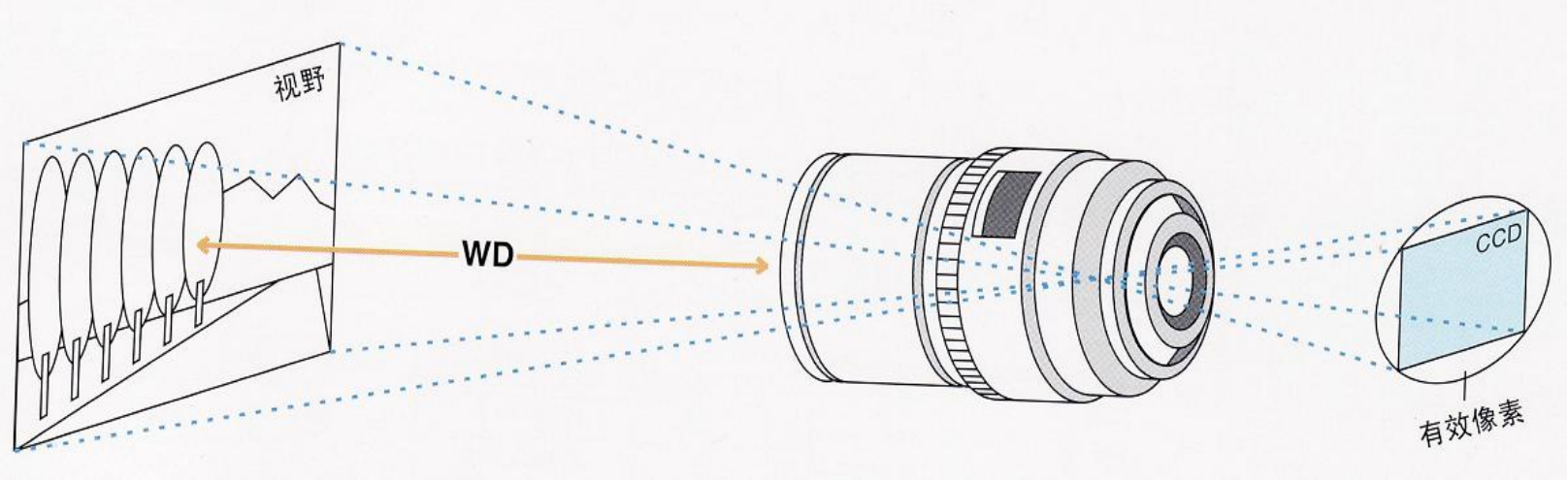
ZED双目相机

Xtion 3D感应摄像头

Kinect v1

Kinect v2

图像传感器CCD



被摄物体的图像经过镜头聚焦在图像传感器CCD上，CCD根据光的强弱积累相应比例的电荷，各个像素积累的电荷在视频时序的控制下，逐点外移，经滤波、放大处理后，形成视频信号输出。CCD的尺寸用对角线长度表示，通常以像素为单位。

图像传感器关键参数

1. **亮度**

   亮度是画面总体的明亮程度，其值越大，画面越明亮。通常使用从0%（黑色）至100%（白色）的百分比来度量。

2. **锐度**

   锐度又叫“清晰度”，它是反映图像平面清晰度和图像边缘锐利程度的一个指标。

3. **对比度**

   画面最亮的部分和最暗部分的比，其值越大，明暗对比越强（配合亮度调整，确保画面整体层次丰富、清晰）。

4. **视场角**

   镜头的视野范围，分为水平和垂直视场角。焦距影响视场角，焦距越大视场角越小，焦距越小，视场角越大。

图像传感器关键参数

5. 灰度

灰度也可认为是亮度，简单的说就是色彩的深浅程度。灰度图像而言，每个像素的亮度用一个数值（即灰度值）来表示，通常数值范围在0到255之间，即可用一个字节来表示，0表示黑、255表示白，而其它表示灰度级别。

6. 色调

是对一幅画面的整体颜色评价。一幅画虽然用了多种颜色，但总体有一种倾向，是偏兰或偏红，是偏暖或偏冷等等。这种颜色上的倾向就是一副画的色调。如右图为红色调、暖色调。

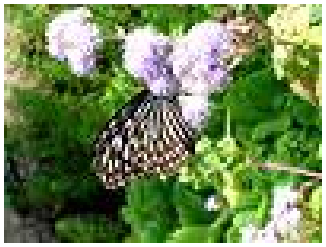
7. 分辨率

图像分辨率是指数字化图像的大小,以水平和垂直的像素点表示。常见的图像分辨率有640*480,800*600,1024*768。

图像传感器关键参数

对比度设定：

对比度-弱



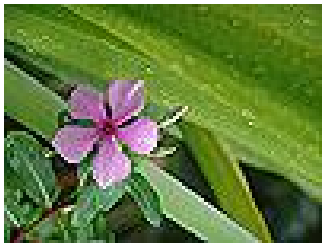
对比度-标准

对比度-强

锐度设定：

锐度-柔和

锐度-标准



锐度-锐化

色度设定：

色度-淡

色度-标准

色度-柔浓

图像传感器关键参数

9. 自动增益（AGC）

通过检测视频信号的平均电平，在较大范围内调节内部电路放大器的增益，使摄像机在不同景物照度下都能够输出标准的视频信号。这种电路是自动增益控制电路，这种调节为自动增益控制。

10. 背光补偿（BLC）

背光补偿也称逆光补偿，它可以有效补偿摄像机在逆光环境下拍摄时画面主体黑暗的缺陷。适用于很亮的背景区域和很暗的前景目标，主要有中心测光和多点测光两种方式。如三洋摄像机VCC-MD800P有中心测光和多点测光。

无背光补偿

普通背光补偿

理想的背光补偿

图像传感器关键参数

12. 超宽动态（WDR）

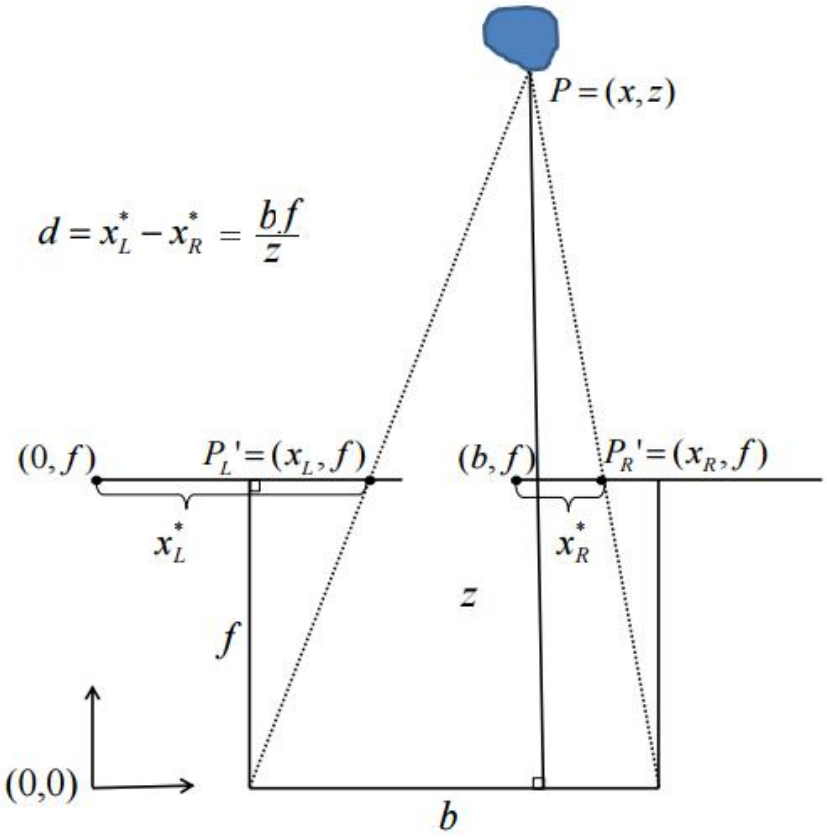
- 在非常强烈的对比下让摄像机看到影像的特色。
- 具有3:1动态范围的摄像机不能同时得到室内和室外清楚的影像。
- 宽动态摄像机比传统摄像机的对比度超出数十倍，可以对室内景物和室外景物分别进行完美曝光，然后叠加这两次曝光的影像，得到清晰的图像。

过度曝光室外影像被清除

曝光不足室内影像变黑

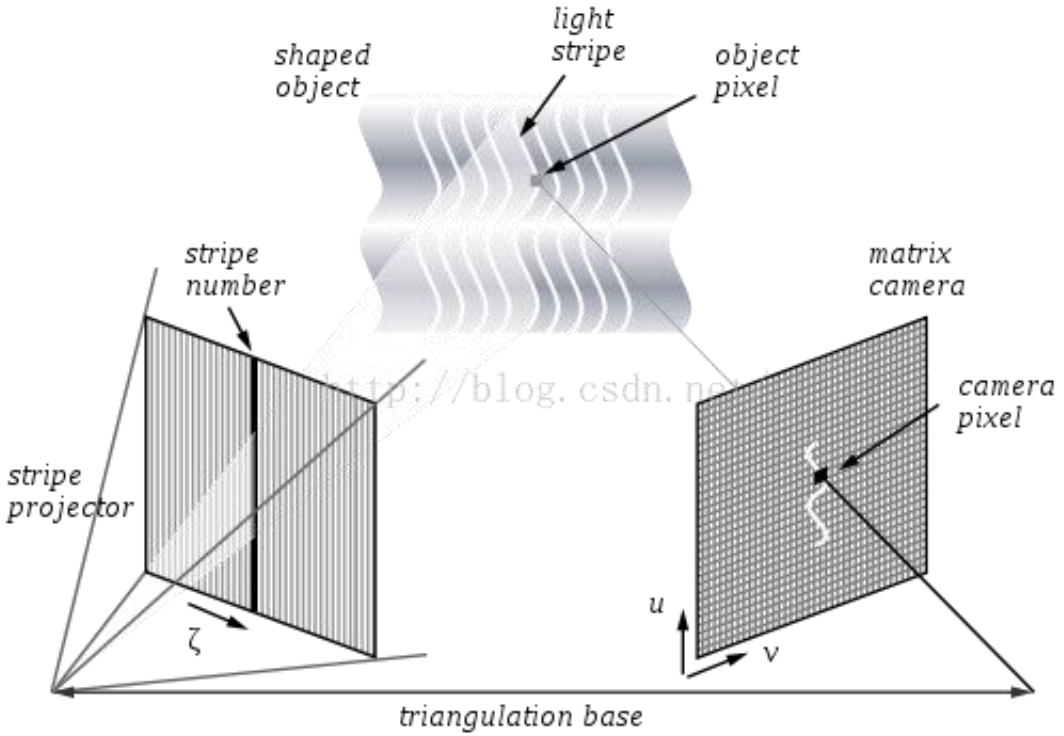
宽动态摄像

视觉传感器



双目立体测距

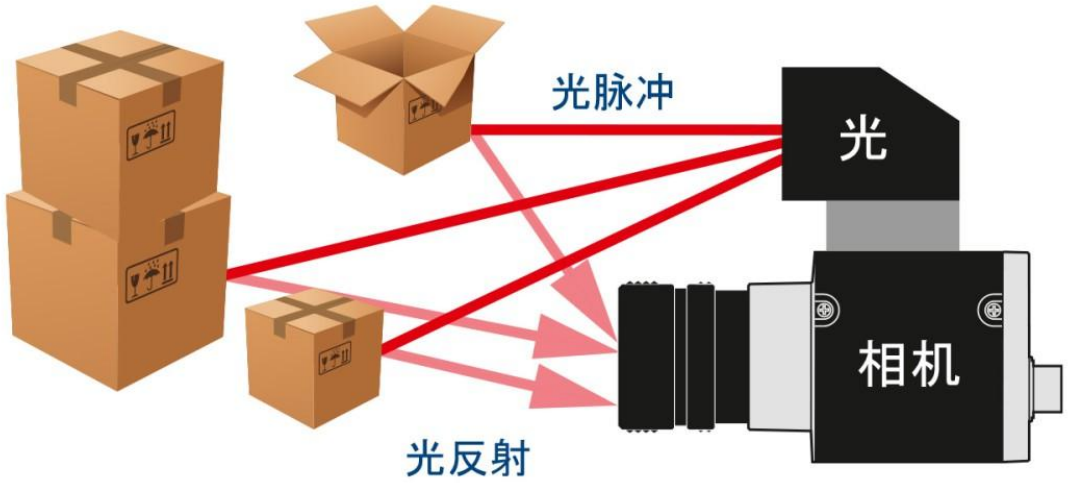
双目匹配采用三角测量原理完全基于图像处理技术，通过寻找两个图像中的相同的特征点得到匹配点，从而得到深度值。



结构光测距

通过投影一个预先设计好的图案作为参考图像(编码光源)，将结构光投射至物体表面，再使用摄像机接收该物体表面反射的结构光图案。通过该图案在摄像机上的位置和形变程度来计算物体表面的空间信息

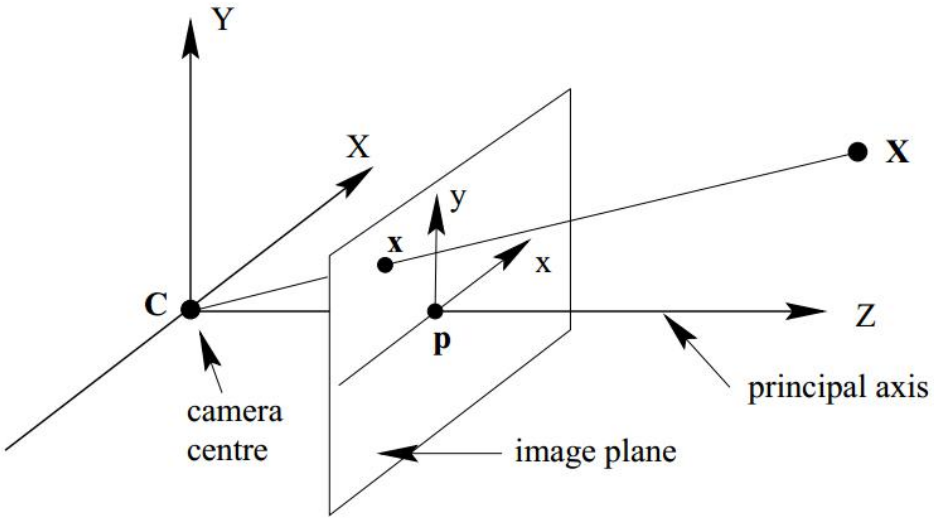
视觉传感器

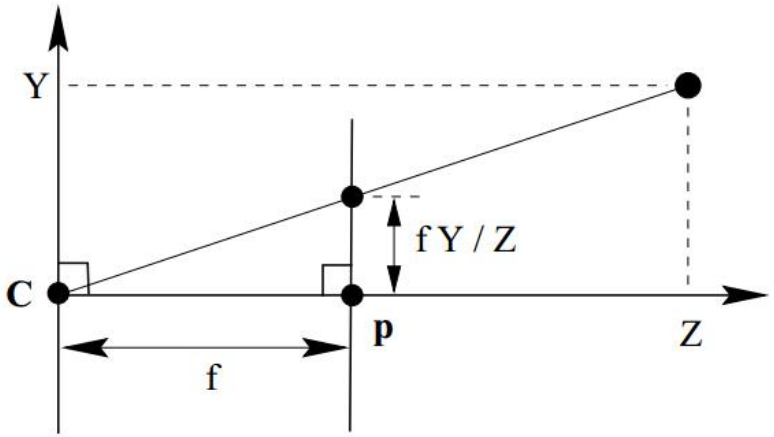


ToF测距

基于光从光源到目标再返回相机所需的时间——距离越长，所花时间越长。对光源和图像采集的同步方式确保能从图像数据中提取和计算该距离。

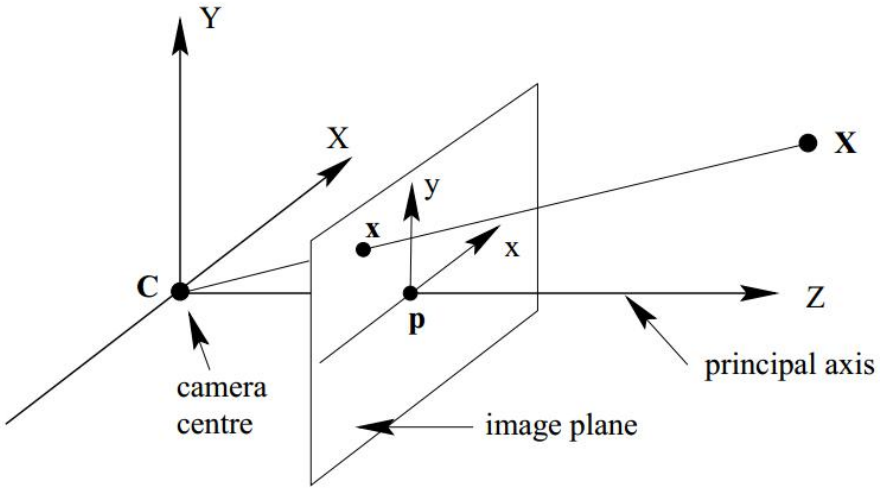
针孔相机模型

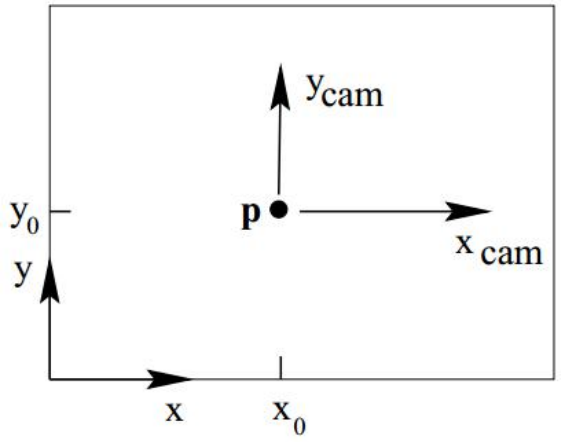




$$(X,Y,Z)^T \rightarrow \left(f\frac{X}{Z}, f\frac{Y}{Z} \right)^T$$

针孔相机模型

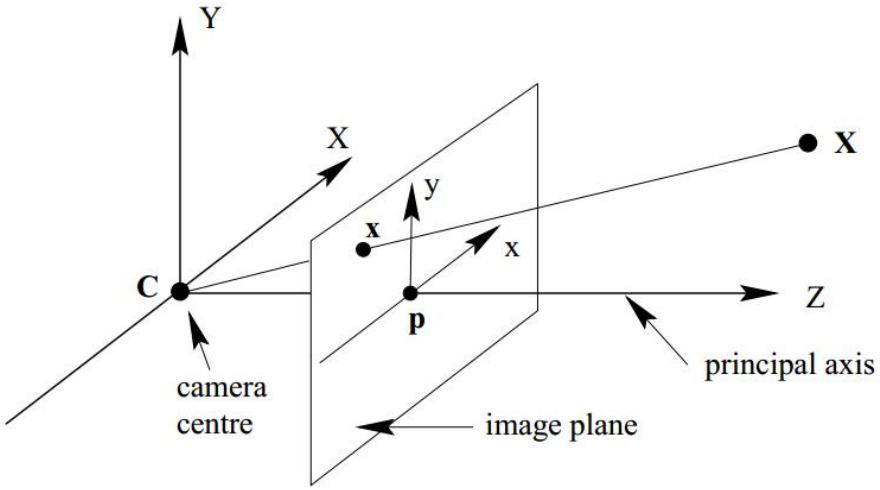


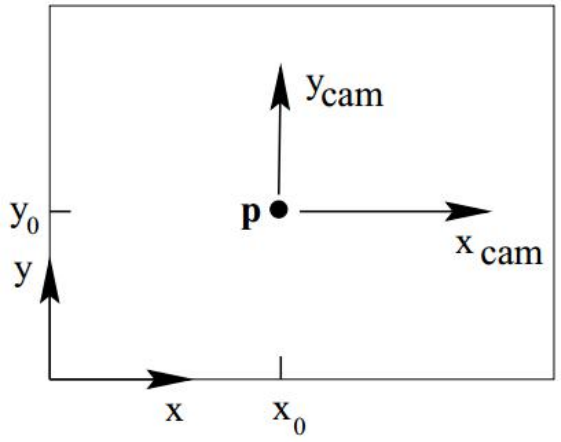


$$(X,Y,Z)^T \rightarrow \left(f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y \right)^T$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

针孔相机模型





从物理成像平面到像素坐标系，它们之间的关系是缩放和平移。

物理成像平面的x'轴缩放 α 倍后得到像素坐标系的u轴；

物理成像平面的y'轴缩放 β 倍后得到像素坐标系的v轴；

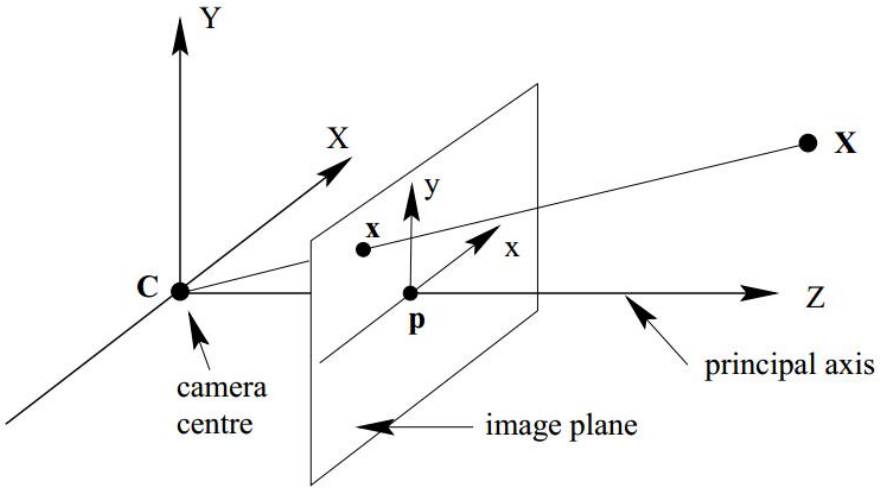
$$f_x = \alpha f,\ f_y = \beta f$$

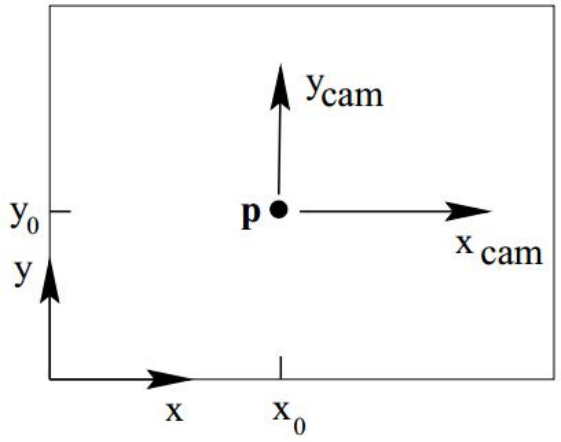
物理成像平面的原点平移后得到像素坐标系的原点；

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & & p_x & 0 \\ & f_y & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

https://blog.csdn.net/JeffYoung_registered/article/details/77370640

针孔相机模型





从相机坐标恢复3D空间点的坐标：

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = d \begin{bmatrix} 1/f_x & & -p_x/f_x \\ & 1/f_y & -p_y/f_y \\ & & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

相机驱动

1. **Webcam相机驱动**

uvc_camera

usb_cam

2. **Kinect v1/xtion驱动**

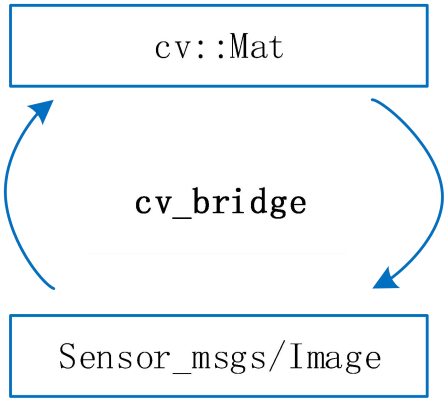
openni
libfreenect

3. **Kinect v2驱动**

Libfreenect2 + iai_kinect2

ROS软件包

cv_bridge



cv_bridge

electric | fuerte | groovy | hydro | **indigo** | jade | kinetic | lunar | Documentation Status

vision_opencv: *cv_bridge* | *image_geometry*

Package Summary

Released | Continuous integration | Documented

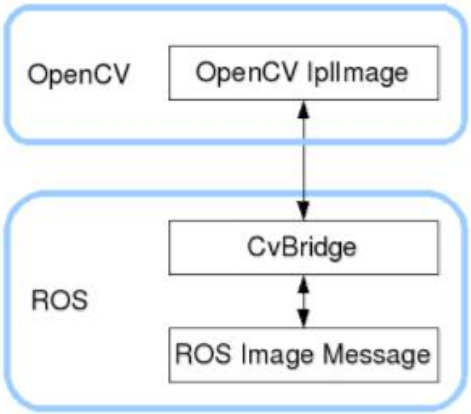
This contains CvBridge, which converts between ROS Image messages and OpenCV images.

- Maintainer status: maintained
- Maintainer: Vincent Rabaud <vincent.rabaud AT gmail DOT com>
- Author: Patrick Mihelich, James Bowman
- License: BSD
- Bug / feature tracker: https://github.com/ros-perception/vision_opencv/issues
- Source: git https://github.com/ros-perception/vision_opencv.git (branch: indigo)

To learn how to interface OpenCV with ROS using CvBridge, please see the tutorials page.

Package Links
Code API
Tutorials
FAQ
Changelog
Change List
Reviews

Dependencies (3)
Used by (145)
Jenkins jobs (6)



cv_bridge：OpenCV的格式不是ROS通用的，需要通过它进行转换。

ROS软件包

pcl_ros

- 可以直接发布和订阅pcl::PointCloud<T>类型的ROS消息。
- 一些点云处理工具，如PCD 文件变为点云消息，点云转换为PCD文件

ROS中点云的格式：

- sensor_msgs/PointCloud
- sensor_msgs/PointCloud2
- pcl::PointCloud<T>

pcl/ Tutorials

A comprehensive list of PCL tutorials can be found on PCL's external website. Here are a few of the tutorials that you might want to check out:

- Reading point cloud data from PCD files
- Downsampling a PointCloud using a VoxelGrid filter
- Planar model segmentation
- Spatial change detection on unorganized point cloud data
- Smoothing and normal estimation based on polynomial reconstruction
- Fast triangulation of unordered point clouds
- Aligning object templates to a point cloud
- Comprehensive list of tutorials for using pcl v1.7.2 library alongside ROS hydro (branch to the hydro_devel) check pcl documentation for explanation

More information about PCL integration in ROS (e.g. point cloud data types, publishing/subscribing) can be found in the ROS/PCL overview.

The following tutorial describes how to use any of the existing tutorials on http://pointclouds.org in a ROS ecosystem using nodes or nodelets.

ROS IDE

http://wiki.ros.org/IDEs

- Eclipse
- CodeBlocks
- Emacs
- Vim
- NetBeans
- QtCreator
- PyCharm
- kDevelop
- JetBrains
- RoboWare Studio
- …

ROS IDE

- 为IDE导入ROS环境变量

kDevelop

kDevelop has excellent C++ support, GDB integration and does semantic syntax highlighting with individual colors for different variables. Such as QtCreator, kDevelop supports opening CMake projects out of the box.

kDevelop must know the ROS environment variables, therefore start kDevelop from a terminal that has sourced your catkin workspace already. Alternatively, create the following desktop file according to the remarks in the general section and mark it as exectuable:

```
cd ~/Desktop
touch kDevelop.desktop
# Now edit the file using the editor of your choice and insert the text from the box below
chmod +x kDevelop.desktop
```

```
[Desktop Entry]
Type=Application
Terminal=false
Exec=bash -i -c "kdevelop"
Name=kDevelop
Icon=kdevelop
```

实验一：网络摄像头的使用

Step1: Webcam测试

安装工具camorama或者cheese。直接用apt-get命令安装。查看摄像头图像。

Step2: Webcam驱动包安装（使用uvc_camera，也可以使用usb_cam）

```
# 从源码安装
sudo apt-get innstall git-core # 如果没有安装git则需要这一步
cd ~/catkin_ws/src/
git clone https://github.com/ericperko/uvc_cam.git
cd ..
catkin_make

# 直接安装
sudo apt-get install ros-indigo-uvc-camera
```

Step3: ROS显示摄像头图像

```
roslaunch uvc_cam test_uvc.launch # 或者  rosrun uvc_camera uvc_camera_node
rosrun image_view image_view image:=/camera/image_raw
```

这里也可以使用rqt_image_view、rviz工具

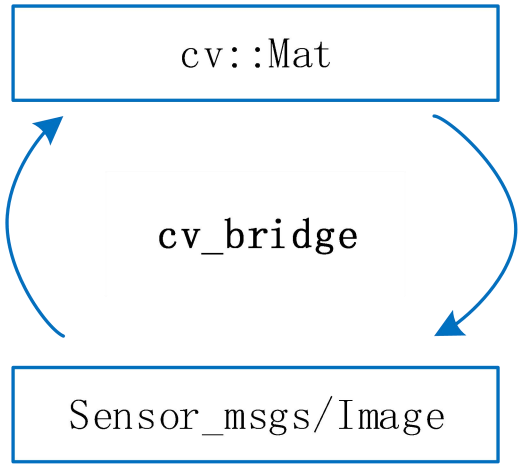
实验二：ROS包中使用OpenCV

```cpp
#include <ros/ros.h>
#include <iostream>
#include <opencv2/opencv.hpp>

int main(int argc, char *argv[])
{
    ros::init(argc, argv, "use_opencv");
    ros::NodeHandle n;

    if(argc != 2){
        std::cout << "Usage: rosrun use_opencv use_opencv_node image_name" << std::endl;
        return -1;
    }

    cv::Mat rgb = cv::imread(argv[1]);
    if(rgb.empty()){
        std::cout << "Can not load image: " << argv[0] << std::endl;
    }

    cv::Mat gray;
    cv::cvtColor(rgb, gray, CV_RGB2GRAY);

    cv::imshow("rgb", rgb);
    cv::imshow("gray", gray);

    cv::waitKey(0);

    ros::spin();
}
```

实验二： cv_bridge程序包的使用

Part1: 订阅图像Msg并且显示

Part2: 发布处理后的图像



参考资料：

- 分布和订阅消息 http://wiki.ros.org/cn/ROS/Tutorials
- cv_bridge Tutorials http://wiki.ros.org/cv_bridge/Tutorials

实验二： cv_bridge程序包的使用

sensor_msgs/Image.msg

```
koker@ubuntu:~/catkin_ws$ cat /opt/ros/indigo/share/sensor_msgs/msg/Image.msg
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image
#

Header header        # Header timestamp should be acquisition time of image
                     # Header frame_id should be optical frame of camera
                     # origin of frame should be optical center of cameara
                     # +x should point to the right in the image
                     # +y should point down in the image
                     # +z should point into to plane of the image
                     # If the frame_id here and the frame_id of the CameraInfo
                     # message associated with the image conflict
                     # the behavior is undefined

uint32 height         # image height, that is, number of rows
uint32 width          # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding       # Encoding of pixels -- channel meaning, ordering, size
                      # taken from the list of strings in include/sensor_msgs/im
age_encodings.h

uint8 is_bigendian    # is this data bigendian?
uint32 step           # Full row length in bytes
uint8[] data          # actual matrix data, size is (step * rows)
```

实验二： cv_bridge程序包的使用

When converting a ROS sensor_msgs/Image message into a `CvImage`, `CvBridge` recognizes two distinct use cases:

1. We want to modify the data in-place. We have to make a copy of the ROS message data.
2. We won't modify the data. We can safely share the data owned by the ROS message instead of copying.

`CvBridge` provides the following functions for converting to `CvImage`:

切换行号显示

```
1  // Case 1: Always copy, returning a mutable CvImage
2  CvImagePtr toCvCopy(const sensor_msgs::ImageConstPtr& source,
3                      const std::string& encoding = std::string());
4  CvImagePtr toCvCopy(const sensor_msgs::Image& source,
5                      const std::string& encoding = std::string());
6
7  // Case 2: Share if possible, returning a const CvImage
8  CvImageConstPtr toCvShare(const sensor_msgs::ImageConstPtr& source,
9                            const std::string& encoding = std::string());
10 CvImageConstPtr toCvShare(const sensor_msgs::Image& source,
11                            const boost::shared_ptr<void const>& tracked_object,
12                            const std::string& encoding = std::string());
```

实验三： 在ROS包中使用pcl库

Part1: 把深度图转换为点云

Part2: 结合彩色图像生成彩色点云

参考资料：

- 高翔博客 http://www.cnblogs.com/gaoxiang12/p/4652478.html
- ROS PCL Tutorials http://wiki.ros.org/cn/pcl/Tutorials
- ROS消息同步机制 http://wiki.ros.org/message_filters

实验三： 在ROS包中使用pcl库

安装PCL: PCL官方安装教程 http://pointclouds.org/downloads/linux.html

```
sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
sudo apt-get update
sudo apt-get install libpcl-all

sudo apt-get install ros-indigo-pcl-ros
sudo apt-get install ros-indigo-pcl-msgs
```

创建ROS包:

```
catkin_create_pkg my_pcl_tutorial pcl_ros roscpp cv_bridge sensor_msgs
cd my_pcl_tutorial/src
touch my_pcl_tutorial_node.cpp
```