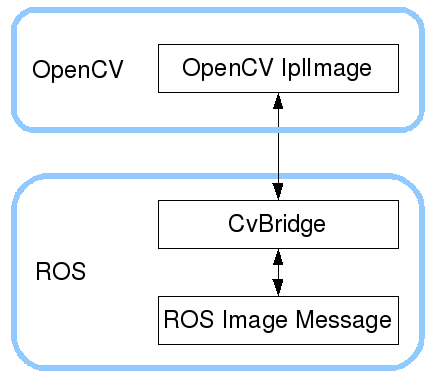
ROS: How to send and receive images between Python and C++ languages using sensor\_msgs and OpenCV images

1. Concepts

ROS passes around images in its own [sensor\_msgs/Image](http://docs.ros.org/api/sensor_msgs/html/msg/Image.html) message format, but many users will want to use images in conjunction with OpenCV. CvBridge is a ROS library that provides an interface between ROS and OpenCV. CvBridge can be found in the [cv\_bridge](http://wiki.ros.org/cv_bridge) package in the [vision\_opencv](http://wiki.ros.org/vision_opencv) stack.

Reference: <http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython>



<http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython>

1. Demo architecture

Topic

/photo\_taking/output\_video

Image(sensor\_msgs.msg.Image)

subscribe

Python

Video-recording

C++

Video-receive

`

publish

Tips: data format is Image(sensor\_msgs.msg.Image)

1. Python module, record video and send to topic

import roslib

import rospy

from std\_msgs.msg import String

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge, CvBridgeError

self.image\_pub = rospy.Publisher('/photo\_taking/output\_video', Image, queue\_size=10)

def record(self):

timer\_start = time.time()

timer\_current = 0

self.video\_cap = cv2.VideoCapture(self.device\_index)

while(self.open==True):

ret, video\_frame = self.video\_cap.read()

if (ret==True):

self.video\_out.write(video\_frame)# todo test video recording show on the screen, data follow

tmp\_image = self.bridge.cv2\_to\_imgmsg(video\_frame, "bgr8")

self.image\_pub.publish(tmp\_image)

# print( str(counter) + " " + str(self.frame\_counts) + " frames written " + str(timer\_current))

self.frame\_counts += 1

else:

break

1. C++ module receive the video frame

#include <ros/ros.h>

#include <std\_msgs/String.h>

#include <sstream>

#include <time.h>

#include <string.h>

#include <netdb.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <pthread.h>

#include <errno.h>

#include <unistd.h>

#include <image\_transport/image\_transport.h>

#include <cv\_bridge/cv\_bridge.h>

#include <sensor\_msgs/image\_encodings.h>

#include <sensor\_msgs/CompressedImage.h>

#include <std\_msgs/Byte.h>

#include "opencv2/objdetect/objdetect.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

image\_transport::ImageTransport it(n);

image\_sub = it.subscribe("/photo\_taking/output\_video", 1, ImageCallback);

void ImageCallback (const sensor\_msgs::ImageConstPtr& msg)

{

cv\_bridge::CvImagePtr cv\_ptr;

std\_msgs::Byte face\_msgs;

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 temp;

cv\_ptr->image.copyTo(temp); //Copy the pointer image to Opencv image 'temp' for processing

cv\_ptr->image.copyTo(frame); //Copy the pointer image to Opencv image 'frame' for processing

cv\_image<bgr\_pixel> cimg(temp); // Define Dlib image cmig

if(take\_photo ==true) // Take the picture

{

image\_pub.publish(msg);

take\_photo = false;

photoName = "/photo\_tmp/Photo" + IntToString(photoNum) +".jpg";

while(FileExists(photoName))

{

photoNum++;

photoName = "/photo\_tmp/Photo" + IntToString(photoNum) +".jpg"; // Change the photoName by increase its ID

}

imwrite(photoName,temp); //Save image

cv::imshow(photoName, temp); //Display saved image

cv::waitKey(3);

std\_msgs::String photo\_link;

photo\_link.data = photoName;

photo\_link\_pub.publish(photo\_link);

photoName = "";

}

}

image\_transport should always be used to subscribe to and publish images. It provides transparent support for transporting images in low-bandwidth compressed formats.

Reference: <http://wiki.ros.org/image_transport>

1. Realsense\_camera

RealSense Camera package allowing access to Intel 3D cameras and advanced modules

This package provides ROS node(s) for using the Intel® [RealSense](http://wiki.ros.org/RealSense)™ R200, F200 and SR300 cameras.

You should install first, reference: <http://wiki.ros.org/realsense_camera>

**Published Topics:**

color/image\_raw ([sensor\_msgs/Image](http://docs.ros.org/api/sensor_msgs/html/msg/Image.html)), Color rectified image. RGB format.

Reference: <http://wiki.ros.org/realsense_camera>

1. Python and C++ module can also receive RealSense Camera data.

self.img\_sub = rospy.Subscriber('/color/image\_raw', Image, self.img\_callback)

def img\_callback(self, msgs):

if self.record:

self.frames.append(self.bridge.imgmsg\_to\_cv2(msgs, "bgr8"))

file\_name = "{}{}.png".format(self.file\_name, self.i)

cv2.imwrite(file\_name, self.frames[len(self.frames)-1])

C++ receive RealSense Camera data demo:

Reference: <http://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages>

1. Reference:
2. <http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython>
3. <http://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages>
4. <http://wiki.ros.org/realsense_camera>
5. <http://wiki.ros.org/image_transport>
6. <https://stackoverflow.com/questions/27080085/how-to-convert-a-cvmat-into-a-sensor-msgs-in-ros>