Note: This tutorial assumes that you have completed the previous tutorials: ROS tutorials (/ROS/Tutorials), Using CvBridge to Convert Between ROS Images and OpenCV Images (/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages).

Please ask about problems and questions regarding this tutorial on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Writing a Simple Image Subscriber (C++)

Description: This tutorial shows how to subscribe to images using any available transport. By using the `image_transport` subscriber to subscribe to images, any image transport can be used at run-time. To learn how to actually use a specific image transport, see the next tutorial.

Tutorial Level: BEGINNER

Next Tutorial: Running the Simple Image Publisher and Subscriber with Different Transports (/image_transport/Tutorials/ExaminingImagePublisherSubscriber)

Contents

catkin | rosbuild

0.1 Writing a Simple Image Subscriber

Here we'll create the subscriber node which will display an image topic on screen.

(Assuming you have created your package in `~/image_transport_ws`)

```
$ cd ~/image_transport_ws/src/image_transport_tutorial
```

0.0.1 The Code

Have a look at the `src/my_subscriber.cpp` file:

Toggle line numbers

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>

void imageCallback(const sensor_msgs::ImageConstPtr& msg)
{
  try
  {
    cv::imshow("view", cv_bridge::toCvShare(msg, "bgr8")->image);
    cv::waitKey(30);
  }
  catch (cv_bridge::Exception& e)
  {
    ROS_ERROR("Could not convert from '%s' to 'bgr8'.", msg-
>encoding.c_str());
  }
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "image_listener");
  ros::NodeHandle nh;
  cv::namedWindow("view");
  cv::startWindowThread();
  image_transport::ImageTransport it(nh);
  image_transport::Subscriber sub = it.subscribe("camera/image", 1, imageCallb
ack);
  ros::spin();
  cv::destroyWindow("view");
}
```

0.0.2 The Code Explained

Now, let's break down the code piece by piece. For lines not explained here, review Writing a Simple Publisher and Subscriber (C++) (http://www.ros.org/wiki/ROS/Tutorials/WritingPublisherSubscriber(c++)).

Toggle line numbers

```
2 #include <image_transport/image_transport.h>
3
```

`image_transport/image_transport.h` includes everything we need to publish and subscribe to images.

Toggle line numbers

```
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>

```

These headers will allow us to display images using OpenCV's simple GUI capabilities.

Toggle line numbers

```
void imageCallback(const sensor_msgs::ImageConstPtr& msg)
```

This is the callback function that will get called when a new image has arrived on the "camera/image" topic. Although the image may have been sent in some arbitrary transport-specific message type, notice that the callback need only handle the normal sensor_msgs/Image (http://docs.ros.org/api/sensor_msgs/html/msg/Image.html) type. All image encoding/decoding is handled automagically for you.

Toggle line numbers

```
{
  try
  {
    cv::imshow("view", cv_bridge::toCvShare(msg, "bgr8")->image);
    cv::waitKey(30);
  }
  catch (cv_bridge::Exception& e)
  {
    ROS_ERROR("Could not convert from '%s' to 'bgr8'.", msg->encoding.c_s
tr());
  }
```

The body of the callback. We convert the ROS image message to an OpenCV image with BGR pixel encoding, then show it in a display window. See this tutorial (/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages) for more on ROS-OpenCV image conversion.

Toggle line numbers

```
  ros::NodeHandle nh;
  cv::namedWindow("view");
```

Create an OpenCV display window.

Toggle line numbers

```
  cv::startWindowThread();
```

We create an `ImageTransport` instance, initializing it with our `NodeHandle`. We use methods of `ImageTransport` to create image publishers and subscribers, much as we use methods of `NodeHandle` to create generic ROS publishers and subscribers.

Toggle line numbers

```
25   image_transport::ImageTransport it(nh);
```

Subscribe to the "camera/image" base topic. The actual ROS topic subscribed to depends on which transport is used. In the default case, "raw" transport, the topic is in fact "camera/image" with type sensor_msgs/Image (http://docs.ros.org/api/sensor_msgs/html/msg/Image.html). ROS will call the "imageCallback" function whenever a new image arrives. The 2nd argument is the queue size.

`subscribe()` returns an `image_transport::Subscriber` object, that you must hold on to until you want to unsubscribe. When the `Subscriber` object is destructed, it will automatically unsubscribe from the "camera/image" base topic.

There are versions of the `subscribe()` function which allow you to specify a class member function, or even anything callable by a Boost.Function object.

Toggle line numbers

```
27   ros::spin();
```

Dispose of our display window.

In just a few lines of code, we have written a ROS image viewer that can handle images in both raw and a variety of compressed forms. In fact, this is a stripped-down version of image_view (/image_view).

0.1 Building your node

Just run:

```
$ catkin_make
```

Next let's run our publisher and subscriber with different image transports (/image_transport/Tutorials/ExaminingImagePublisherSubscriber).

Wiki: image_transport/Tutorials/SubscribingToImages (last edited 2015-05-13 15:12:11 by StefanOsswald (/StefanOsswald))