Interfacing between ROS and OpenCV

March 7, 2014

1 Environment

- Ubuntu 12.04
- ROS Groovy Desktop Full
- Logitech 1080p camera

2 Install Camera Drivers

1. Open terminal, go to the directory where you want to install the camera driver package
2. Install the package by typing:

   ```
   $ sudo apt-get install ros-groovy-camera-umd
   ```

3. Once installed, type in the terminal

   ```
   roscd uvc_camera
   ```

   If you are now in the uvc_camera directory, then the installation is done

3 Find which device is the camera connected to

1. open terminal window and type in

   ```
   $ ls /dev/video*
   ```

   You will see something like :

   ```
   /dev/video0
   ```

   If we have only one camera connected, then the device used is always video0

4 Create a new package for image processing

1. Open a new terminal and in the workspace directory (Here is the home directory), type the following:

   ```
   $ roscreate-pkg testcamera image_transport roscpp std_msgs opencv2 cv_bridge uvc_camera
   ```

   This will create a ROS package called "testcamera"

   ```
   $ roscd testcamera
   $ mkdir launch
   $ cd launch
   $ vim uvcCameraLaunch.launch
   ```

   Then paste the following in the *uvcCameraLaunch.launch* file (The launch file and all the following other files are available in the directory *testcamera*)

   ```
   <!--xml-->
   <launch>
     <node ns="camera" pkg="uvc_camera" type="camera_node" name="uvc_camera" output="screen">
       <param name="width" type="int" value="640" />
       <param name="height" type="int" value="480" />
       <param name="fps" type="int" value="30" />
       <param name="frame" type="string" value="webcam" />
       <param name="device" type="string" value="/dev/video0" />
     </node>
   </launch>
   ```

   Save and exit. Please note the parameter device with value : */dev/video0*. Change this to the actual device path.

2. Before *rosmake* this package, we have to add its path to the *.bashrc* file. The *.bashrc* is in the home directory, you can edit using:

   ```
   $ vim .bashrc
   ```

   Add the following line at the end of the *.bashrc* file:

   ```
   export ROS_PACKAGE_PATH=~/testcamera:$ROS_PACKAGE_PATH
   ```

   You MUST restart the terminal after modify the *.bashrc* file

3. Then type:

   ```
   $ rosmake testcamera
   ```

   In the results there is something like:

   ```
   Built 39 packages with 0 failure
   ```

5 Test the Launch script

1. In a terminal type:

   ```
   $ roslaunch testcamera uvcCameraLaunch.launch
   ```

2. If you want to see the published topics, please open a new terminal and type:

   ```
   $ rostopic list
   ```

3. Let's look at the topic */camera/image_raw*, type:

   ```
   $ rosrun image_view image_view image:=/camera/image_raw
   ```

   Then you should see a new window pop-up with a continuous streaming video capture output of the web camera.

6 Write sample code for image processing

1. In a new terminal window, type the following:

   ```
   $ roscd testcamera/src
   ```

   Now create the source file: *main.cpp* and paste the following code in the file: (If you don't want to copy and paste, there is a file named *main.cpp* in *testcamera/src*, you can move it to your *src* directory)

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

namespace enc = sensor_msgs::image_encodings;

static const char WINDOW[] = "Image Processed";

image_transport::Publisher pub;

void imageCallback(const sensor_msgs::ImageConstPtr& original_image)
{
    cv_bridge::CvImagePtr cv_ptr;
    try
    {
        cv_ptr = cv_bridge::toCvCopy(original_image, enc::BGR8);
    }
    catch (cv_bridge::Exception& e)
    {
        ROS_ERROR("testcamera::main.cpp::cv_bridge exception: %s", e.what());
```

```
        return;
    }

    for(int i=0; i<cv_ptr->image.rows; i++)
    {
        for(int j=0; j<cv_ptr->image.cols; j++)
        {
            for(int k=0; k<cv_ptr->image.channels(); k++)
            {
                cv_ptr->image.data[i*cv_ptr->image.rows*4+j*3 + k] = \
                       255-cv_ptr->image.data[i*cv_ptr->image.rows*4+j*3 + k];
            }
        }
    }


    cv::imshow(WINDOW, cv_ptr->image);
    cv::waitKey(3);
    pub.publish(cv_ptr->toImageMsg());
}

int main(int argc, char **argv)
{

    ros::init(argc, argv, "image_processor");
    ros::NodeHandle nh;
    image_transport::ImageTransport it(nh);
    cv::namedWindow(WINDOW, CV_WINDOW_AUTOSIZE);
    image_transport::Subscriber sub = it.subscribe("camera/image_raw", 1, imageCallback);
    cv::destroyWindow(WINDOW);
    pub = it.advertise("camera/image_processed", 1);
    ros::spin();
    ROS_INFO("testcamera::main.cpp::No error.");
}
```

2. Then we have to modify the *CMakeList.txt* file:

```
$ roscd testcamera
$ vim CMakeList.txt
```

   Add the following line to the end of the *CMakeList.txt*:

```
rosbuild_add_executable(testcamera src/main.cpp)
```

3. Now we can compile and make package.

```
$ rosmake testcamera
```

4. Make sure that roscore is running, if not, please type:

```
$ roscore &
```

Then we are going to run the package:

```
$ rosrun testcamera testcamera
$ rosrun image_view image_view image:=/camera/image_processed
```

Then you should see a window pop-up with a continuous streaming video capture output of the webcam with image inverted. Press Ctrl-C to exit.

This instruction is in reference to [1]

References

[1] Siddhant. http://siddhantahuja.wordpress.com/2011/07/20/working-with-ros-and-opencv-draft/, 2011.