Using Baxter Cameras

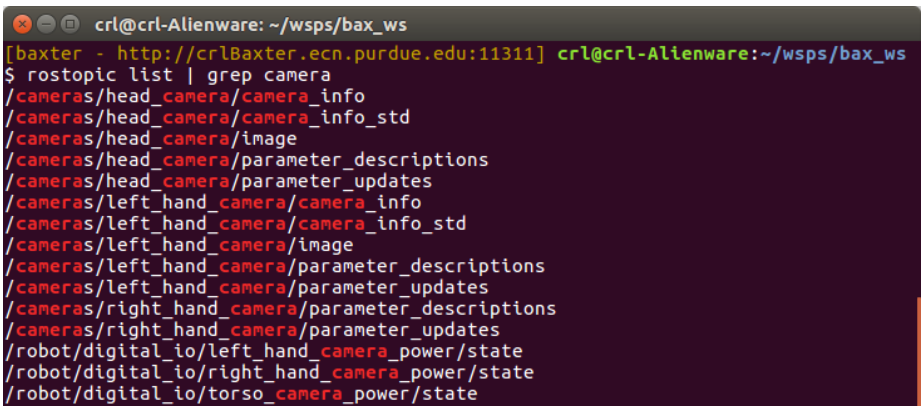
Baxter has 3 cameras – one in each hand and one on the head. All three are color cameras. But, at any one time only two cameras can be active due to 64-bit limitations with the Indigo Baxter's internal USB hub. Thus, only two cameras can be powered at once.

The cameras are named ‘left_hand_camera’, ‘right_hand_camera’, and ‘head_camera’

To list all the camera topics, connect to Baxter ($ ./baxter.sh) and then run

```
$ rostopic list | grep camera
```
(List all topics with word ‘camera’)

The following should get printed on the screen



Opening and closing cameras

The camera control tool provides the API to open and close different cameras. This is through the camera_control.py script which is part of the baxter_tools package.

The script has options for opening, closing and listing currently open cameras. Here’s the set of options for the file

```
usage: camera_control.py [-h] [-o CAMERA] [-r [X]x[Y]] [-c CAMERA] [-l] [-e]

optional arguments:
  -h, --help            show this help message and exit
  -o CAMERA, --open CAMERA
                        Open specified camera
  -r [X]x[Y], --resolution [X]x[Y]
                        Set camera resolution (default: 1280x800)
  -c CAMERA, --close CAMERA
                        Close specified camera
```

```
  -l, --list            List available cameras
  -e, --enumerate       Clear and re-enumerate connected devices
Valid resolutions:
[1280x800, 960x600, 640x400, 480x300, 384x240, 320x200]
```

Example Usage from Command Line

To **list currently open cameras**:

```
$ rosrun camera_control.py -l
```

```
$ rosrun baxter_tools camera_control.py -l
head_camera
left_hand_camera  -  (open)
right_hand_camera  -  (open)
```

Now if we want to open head_camera, we need to first close one of the hand cameras. Let us close right_hand_camera

Closing a camera

```
$ rosrun camera_control.py -c right_hand_camera
```

Now check the list of open cameras

```
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
$ rosrun baxter_tools camera_control.py -c right_hand_camera
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
$ rosrun baxter_tools camera_control.py -l
head_camera
left_hand_camera  -  (open)
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
```

Opening a camera

Example: open head_camera with resolution 1280x800

```
$ rosrun baxter_tools camera_control.py -o head_camera -r 1280x800
```

```
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
$ rosrun baxter_tools camera_control.py -o head_camera -r 1280x800
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
$ rosrun baxter_tools camera_control.py -l
head_camera  -  (open)
left_hand_camera  -  (open)
[baxter - http://crlBaxter.ecn.purdue.edu:11311] crl@crl-Alienware:~/wsps/bax_ws
```

NOTE: As only two cameras can be open simultaneously, trying to open the third camera will result in an error.

Example Usage via Launch File

Opening ‘right_hand_camera’ and setting resolution to 320x200

```
<launch>

  <node pkg="baxter_tools" type="camera_control.py" name="camera_con" args="-
o right_hand_camera -r 320x200"/>

</launch>
```

Example Usage from Python

Program to open or close cameras from code.

The following program shows two functions one for opening a camera with required resolutions and the other is to close the camera.

```
#!/usr/bin/python2

import rospy

from baxter_interface.camera import CameraController

# Open camera (camera is a string and res is a 2-element vector)

def open_cam(camera, res):

# Check if valid resolution

    if not any((res[0] == r[0] and res[1] == r[1]) for r in
CameraController.MODES):

                rospy.logerr("Invalid resolution provided.")

    # Open camera

    cam = CameraController(camera)  # Create camera object

    cam.resolution = res    # Set resolution

    cam.open()  # open


# Close camera

def close_cam(camera):

    cam = CameraController(camera)  # Create camera object

    cam.close() # close


# Example usage, assuming left_hand_camera is open and
right_hand_camera is closed (Check list of active cameras to know
which cameras are open)
```

```
if __name__ == '__main__':
    rospy.init_node('camera_control', anonymous=True) # init ros node
    # Close left_hand_camera
    close_cam('left_hand_camera')
    # Open right_hand_camera and set resolution to 1280x800
    open_cam('right_hand_camera',(1280,800))
```

(Refer to http://sdk.rethinkrobotics.com/wiki/Camera_Control_Tool for details regarding the camera control tool)

Viewing Baxter Camera Images

The images from Baxter's cameras can be viewed using either the image_view package or in rviz (a ros visualization tool).

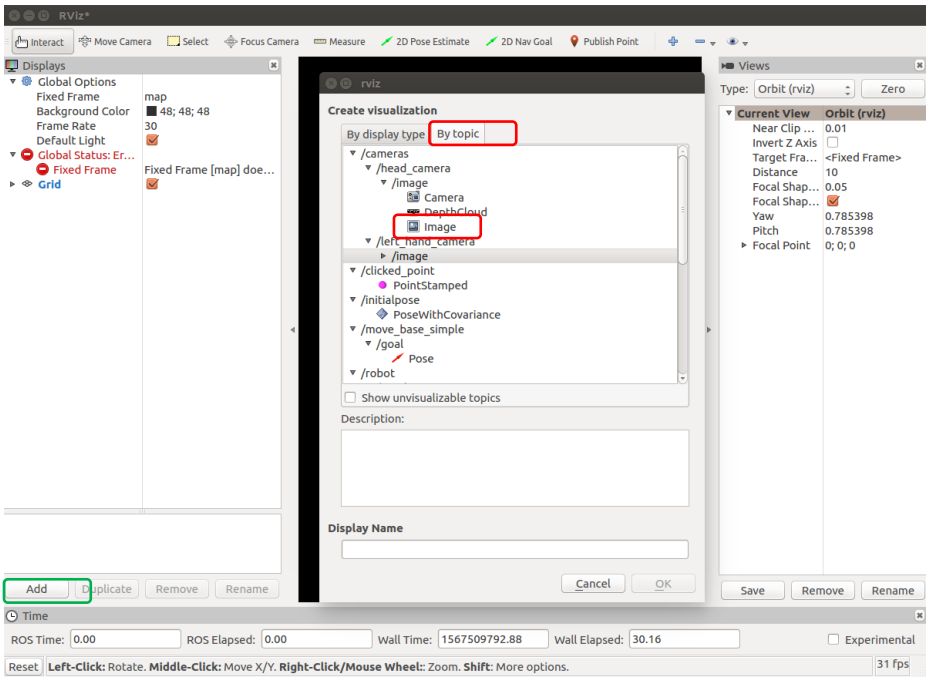
Example: Once a camera is open, we can view this through either of the above-mentioned tools. For example, to view the 'head-camera' using image_view run the following command.

```
rosrun image_view image_view image:=/cameras/head_camera/image
```

The same operation can be performed with rviz. To do this first start rviz

```
rosrun rviz rviz
```

Then click on 'Add' (Highlighted by green rectangle) > Then go to By topic tab > Then choose the image topic to view.



Working with images in code using OpenCV

Baxter camera images are in the format sensor_msgs/Image. This is different from the image format in OpenCV. Therefore, we need to convert these ROS images to OpenCV images if we are using them with OpenCV. This can be done using a ros package called 'cv_bridge'. This package provides tools for converting images from OpenCV format to ROS format and vice versa.

Example Usage in Code:

Following is a sample code for subscribing to the Baxter head_camera image and then converting it to OpenCV format and displaying it.

```
#!/usr/bin/python2

import rospy, cv2, cv_bridge

from sensor_msgs.msg import Image


bridge = cv_bridge.CvBridge()    # Initialize CV Bridge object
```

```
# Callback function to subscribe to images
def image_callback(ros_img):
    # Convert received image message to OpenCv image
    cv_image = bridge.imgmsg_to_cv2(ros_img, desired_encoding="passthrough")
    cv2.imshow('Image', cv_image)   # display image
    cv2.waitKey(1)


if __name__ == '__main__':
    rospy.init_node('Camera_Subscriber',anonymous=True)   # Initialze ROS node
    # Subscribe to head_camera image topic
    rospy.Subscriber('/cameras/head_camera/image', Image, image_callback)
    rospy.spin()    # sleep
    cv2.destroyAllWindows() # Destroy CV image window on shut_down
```

(cv_bridge tutorial - http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython).

Some basic OpenCV functions:

- Load or read image : `cv2.imread('filename')`
- Display image: `cv2.imshow(img)`
  `cv2.waitKey(0)`
- Resize image: `cv2.resize(img, (width, height))`
- Changing color space – HSV, RGB, Gray
  `cv2.cvtColor(image, cv2.COLOR_BGR2HSV)`
  `By default OpenCV uses a BGR color scheme(not RGB)`
- Accessing parts of image: The image is *width x height x channel* dimension numpy array. The width and height are the number of pixels in x and y and the channel refers to number of channels in the image (BGR - 3, HSV - 3, GRAY - 1)
- Elements can be accessed as follows `img[x,y,n]`, where `(x,y)` refers to the coordinates of the pixel and `n` is the desired channel.
  - Accessing elements example: To obtain the red color value of a BGR image at pixel position `(10, 20)` in image Img. This can be obtained by Img[10,20,2]

(More OpenCV functions – https://heartbeat.fritz.ai/opencv-python-cheat-sheet-from-importing-images-to-face-detection-52919da36433

OpenCV Tutorials - https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html)