BAXTER INTERFACE GUIDELINES

1. Go to: http://sdk.rethinkrobotics.com/wiki/Workstation_Setup

   Follow the guide starting from point 4.
   Work within your ROS workspace directory (in the tutorial: '`ros_ws`').
   At *pt 6*, the baxter.sh configuration for our lab is:
   - `baxter_hostname=`"B2.local"
   - `your_hostname=`"YOUR_HOSTNAME.local"
     - where YOUR_HOSTNAME can be recovered by typing in a terminal: `hostname`

NOTE 1: if you use another ROS version then 'indigo', you have to use the correct name in all the commands in which it occurs (included the `ros_version` flag in the baxter.sh)

NOTE 2: if you use a virtual machine, use the ip of your host machine and set the network adapter to bridged in the virtual machine options (tested on virtualbox).

2. to check that the Baxter SDK installation is successfully completed run:
   - `$ rosrun baxter_tools tuck_arms.py -u`
     or
   - `$ rosrun baxter_tools tuck_arms.py -t`

NOTE 3: every time that your system uses the Baxter you have to run all the ROS commands from a 'baxterised' terminal (to baxterise a terminal you can run in your workspace: `$ . baxter.sh`). Be aware that if others are using the robot in the lab, your are sharing the same ROS core and all the nodes and commands to the moving modules are overlapping. This causes unsuspected motions and it is strongly not recommended. So, arrange your work in order to not have multiple groups working at the same time (eventually, a scheduling Google calendar is available. Contact me luca.buoncompagni@edu.unige.it to book the robot).

3. to get familiar with Baxter check out: http://sdk.rethinkrobotics.com/wiki/Examples
   in particular the teaching by demonstrations example: *Joint Trajectory Playback Example*

NOTE 4: be careful when the robot is moving, it can be dangerous! Keep the red stopping button on your hand and be aware that the robot avoids only its arm. It would hit its head and surrounding objects (e.g., the table). Be especially carefull to the head, since the kinect is calibrated based on its position!

POINT CLOUD INTERFACE

We place the Kinect in the head of the robot and we develop a driver which calibrates the camera and streams the cloud in the Baxter ROS core. This driver manages the pan motion of the Baxter and the tilt motion of the Kinect (**ATTENTION:** do not use the tilt motion of the Baxter head because it would hit the camera and destroy the calibration!). Such a driver can be found in: https://github.com/EmaroLab/kinect_calibration and it is already installed in the PC placed at a robot side. Let see now how to run the driver from your PC:

1. install dependences:
   - `$ sudo apt-get install openssh-client`
   - `$ sudo apt-get install openssh-server`
   - `$ sudo apt-get install sshpass`
   - `$ sudo apt-get install ros-`*distr*`-freenect-launch`
     where, in the last, -*distr*- should be changed with the name of your ROS version

2. connect through SSH with the PC where the Kinect is connected (psw:eureka)
   - `$ ssh -X emarolab@130.251.13.36`

if no error occurs the procedure should store a key that will be automatically used all times later. Hence, you can skip this step in the future.

3. download the script:
   https://github.com/EmaroLab/kinect_calibration/blob/master/emarolab_kincect_driver/baxterisedRemoteTerminal.sh
   Give it execution permission and run it.
   Remember to switch on the robot first since you need the Baxter core!

NOTE: this script runs the driver but, since it is a complex task, the PC that host it can freeze. In order to limit this, run this script only when your are subscribing to the cloud. Close it (`ctrl+C`) otherwise.

4. The driver is ready when the kinect has moved parallel to the ground. Only at this time you can orientate the camera with the following commands for:
   - tilt motion:
     - open a ROS service in a baxterised terminal
       - `$ rosrun kinect_aux kinect_aux_node`
     - publish the deseride angle, for instance of -15° (range [-55; 0]), in another baxterised terminal
       - `$ rostopic pub /tilt_angle std_msgs/Float64 -- -15`
     - if you wish, check out the current position of the Kinect
       - `$ rostopic echo /cur_tilt_angle`
   - pan motion, for instance of 0.1rad, (not recommended since it has to be manually reset to 0.0rad all the time a new instance of the driver is launched). Avoid this unless extremely necessary.

     -
       ```
       rostopic pub /robot/head/command_head_pan baxter_core_msgs/HeadPanCommand
       -- 0.1 20
       ```

5. When the driver is on and you camera is in place. You can check out the cloud from an *rviz* instance opened from a baxterised terminal. The cloud is streamed as a `PointCloud2` message in the topic: /cameraB_depth_optical_frame

PITT OBJECT RECOGNITION

1. A stable version of the software architecture can be downloaded from: https://github.com/EmaroLab/primitive_identification_tracking_tagging
2. Download it in your ROS workspace
3. do (try more than one time) `$ catkin_make`
4. Run the Kinect driver (see previous guidelines)
5. To Run it, from a baxterised terminal, use:
   ```
   $ roslaunch pitt_object_table_segmentation table_segmentation.launch
   ```

Check out the parameters available in that launch file, especially the `pitt_show_...` flags, to visualize the results of the algorithms and monitor the system.

All those parameters are described on a draft documentation that you can find here:
https://github.com/EmaroLab/primitive_identification_tracking_tagging/blob/master/Doc/PITTreport.pdf
this document refer to an outdated architecture.

Documentation about the update shape of the architecture can be found here:
https://github.com/EmaroLab/primitive_identification_tracking_tagging/blob/master/Doc/PITT-publ.pdf

To know the real type of messages exchanged between the nodes of the architecture, check out the `pitt_msgs` package that contains all (and only) the definition of: the messages (msg) and services (srv) data exchanged.

REMARK

Check out our tutorials and guide lines at:
https://github.com/EmaroLab/docs

Your final project for the assignment must follow those standards and be uploaded on our Github, so it is better to know in advance!