Universidad de Concepción

Facultad de Ingeniería
El origen de grandes soluciones

Resumen y Aplicaciones

Robot Baxter

Felipe Sepúlveda Sepúlveda
Ingeniero Civil Industrial
Celular
(56) 9 49012627
Correo
felisepulveda@udec.cl

Hugo Ubilla Bustamante
Ingeniero Civil Industrial
Celular
(56) 9 97163624
(61) 478 025 112
Correo
hugoubilla@udec.cl

Resumen
Robot Operating System

Visitar la siguiente página para las instrucciones sobre la instalación de ROS junto a Baxter.

http://sdk.rethinkrobotics.com/wiki/Main_Page

Instalación de las aplicaciones de este documento u otros paquetes de ROS.

1) Pegar paquete en la carpeta **src** del espacio de trabajo de ROS.
2) Abrir terminal.
3) Iniciar sesión de ROS.
4) Dirigirse a la raíz del espacio de trabajo.
5) Escribir el comando catkin_make.
6) Esperar a que se complete la operación.

Creación de paquetes de ROS.

1) Iniciar sesión de ROS.
2) Abrir terminal.
3) Dirigirse a la carpeta **src** del espacio de trabajo de ROS.
4) Escribir el siguiente comando:

   catkin_create_pkg [nombre_paquete] [dependencias_separadas_por_un_espacio]

5) Dirigirse a la raíz del espacio de trabajo.
6) Escribir el comando catkin_make.
7) Esperar a que se complete la operación.

Algunos de los comandos más comunes de ROS.

- rosrun [nombre_paquete] [ejecutable]: Permite ejecutar un programa dentro de un paquete.

- roslaunch [nombre_paquete] [ejecutable_launch]: Permite ejecutar un conjunto de nodos a través de un archivo con extensión .launch.

- Comandos rostopic: Permite interactuar con los ROS topics.

    - rostopic list: Muestra una lista de tópicos activos.
    - rostopic pub: Publica información en un tópico.
    - rostopic echo: Imprime los mensajes de un cierto tópico en la pantalla.
    - rostopic type: Imprime el tipo de tópico.

- Comandos rosservice: Permite interactuar con los ROS services.
    - rosservice list: Muestra los servicios activos.
    - rosservice call: Llama a un servicio dado los argumentos.

- rosmsg/rossrv: Permite obtener información sobre la estructura de mensajes y servicios. Por ejemplo, al agregar la palabra “show” y luego el tipo de mensaje, se mostrará cada uno de los campos del mensaje.

- rosnode list: Muestra una lista de los nodos activos de la sesión actual.

- roscd [paquete[/subdir]]: Cambiar de directorio a un paquete. Similar a cd.

- rosls [paquete[/subdir]]: Muestra los archivos dentro de un paquete. Similar a ls.

Es importante que el lector utilice el comando “–h” luego de los comandos presentados para obtener más información.

Aplicación 1
Red Neuronal Objetos.

Este es un ejemplo de reconocimiento y depósito de distintos tipos de objetos en posiciones predeterminadas. Para la tarea de reconocimiento se utiliza una red neuronal entrenada bajo el conjunto de datos **'coco dataset**'. Este conjunto de datos está formado por 330 mil imágenes divididas en 80 categorías de objetos. Luego que el objeto es identificado, Baxter procede a recogerlo y depositarlo en una posición indicada al comienzo de la ejecución del script.

Instrucciones

Instalación

1) Instalar **'TensorFlow's Object Detection API'**, usando el siguiente comando: *'git clone https://github.com/tensorflow/models.git*'.

2) Instalar cada una de las librerías necesarias indicadas en el bloc de notas adjunto llamado: **'Red Neuronal Objetos .txt'.**

3) Poner los scripts **'ssd_mobilenet_v1_coco_2018_01_28.py', 'MueveBaxter.py'** y **'traeImagen.py'** dentro de la carpeta **'models/research/object_detection'**

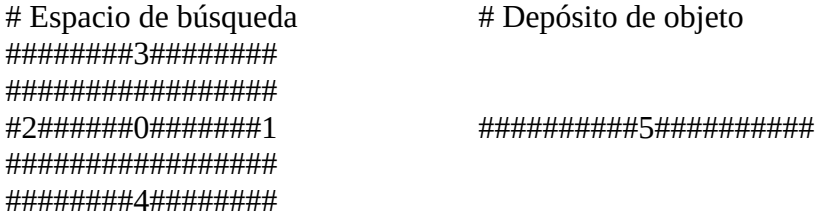
Montaje

El objeto a detectar debe contrastar con el fondo, para disminuir los problemas de detección. Colocar una cartulina bajo el brazo izquierdo del robot.

Ejecución

1) Seleccionar el tipo de objeto que Baxter debe reconocer. Para esto vamos a la siguiente carpeta **'models/research/object_detection/data'** y abrimos el archivo **'mscoco_label_map.pbtxt'**. Encontraremos las 80 clases distintas de objetos con las cuales la red neuronal esta entrenada. Buscamos el id que nos interese y lo agregamos al script **'ssd_mobilenet_v1_coco_2018_01_28.py'** en la variable **clase_index**.

2) Ejecutar el siguiente comando *'python ssd_mobilenet_v1_coco_2018_01_28.py'*. Este ejemplo está diseñado para usar el brazo izquierdo del robot. Esperar unos segundos mientras se resetean y abren las cámaras de Baxter. Aparecerá un mensaje en la pantalla que dice **'Ahora graba posiciones'**. Se debe tomar el brazo izquierdo del robot y grabar 6 posiciones, según el siguiente esquema:



La posición 0 corresponde al origen, y es donde Baxter posicionara la cámara al comenzar la ejecución del script. Las posiciones 1, 2, 3 y 4 definen el cuadrilátero donde Baxter buscara el objeto escogido, siendo la posición 3 la más cercana al cuerpo del robot. Finalmente, la posición 5 nos indicara el centro de masa del depósito donde el robot dejara el objeto una vez que lo ha tomado. Es importante señalar que el orden de las posiciones debe ser seguido al pie de la letra, de lo contrario se mostraran inconsistencias en la ejecución del ejemplo. Para grabar un movimiento, se debe tomar el brazo de Baxter, posicionarlo en la ubicación escogida con la cámara apuntando perpendicularmente a la mesa y presionar el botón central y redondo ubicado entre la articulación W0 y W1 de la figura x (button wheel). Una vez que las 6 posiciones han sido grabadas, se debe presionar cualquiera de los dos botones que están alrededor del botón central y redondo (above wheel o below wheel). Esto último dará inicio al script.

Otros

1) Los scripts están parametrizados, por lo que es posible cambiar la red neuronal y el conjunto de datos sobre la cual fue entrenada. Para esto se debe descargar la red neuronal deseada a la carpeta '**models/research/object_detection**', y modificar las variables **MODEL_NAME, PATH_TO_CKPT y PATH_TO_LABELS** del script '**ssd_mobilenet_v1_coco_2018_01_28.py**', para indicar el nombre de la red, su grafo y la información de las etiquetas respectivamente. Además, si el nuevo conjunto de datos sobre el cual esta entrenada la nueva red tiene un distinto número de categorías de objetos, se debe modificar la variable **NUM_CLASSES**.

2) Es posible cambiar el brazo del robot, modificando las instrucciones contenidas entre los comentarios #Instrucciones brazo y #Fin instrucciones brazo, cambiando la palabra '**left**' por '**right**'.

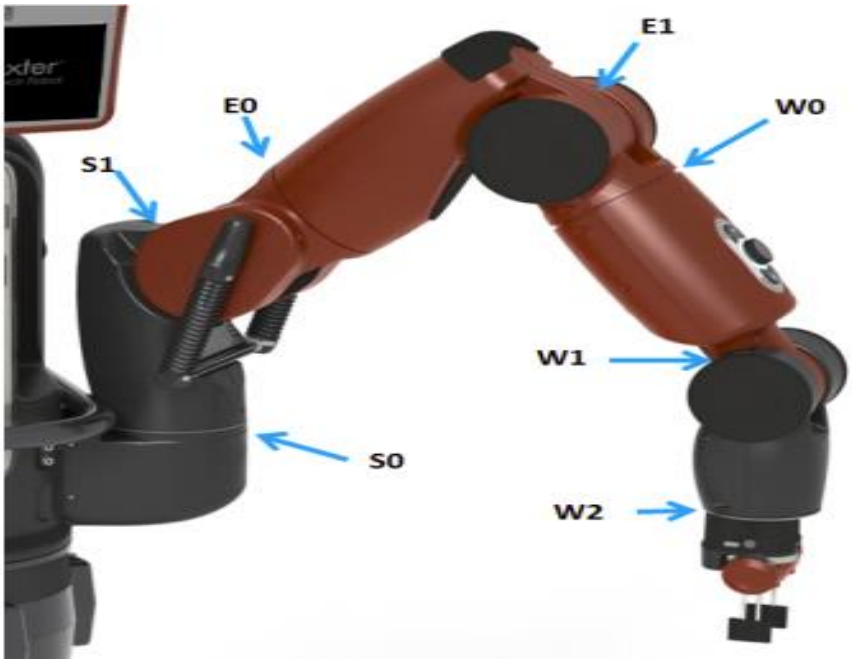


Figura 1: “Articulaciones del brazo izquierdo de Baxter”

Aplicación 2
Red Neuronal de Herramientas y Kmeans para colores, usando ambos brazos de Baxter simultáneamente.

Este es un ejemplo de reconocimiento y clasificación de herramientas y colores. Para la tarea de reconocimiento de herramientas, se utiliza una red neuronal, entrenada sobre un conjunto de 5 clases de herramientas utilizando la teoría de análisis complejo de contornos. Los tipos de herramientas son alicates, destornilladores, taladros, llaves y martillos. Se usa el brazo izquierdo. Para la tarea de reconocimiento de colores se utiliza el algoritmo Kmeans, entrenado sobre 3 distintos tipos de colores. Los colores son rojo, azul y verde. Se usa el brazo derecho.

Instrucciones

Instalación

1) El reconocedor de herramientas está compuesto de 4 scripts: **'DNNGood.py', 'CAGood.py', 'MueveGood.py' y 'traeImagen.py'**; 4 archivos **'my_model_final_good2.ckpt.data-00000-of-00001', 'my_model_final_good2.ckpt.index' , 'my_model_final_good2.ckpt.meta' y Plantillasaux.txt;** y 37 imágenes. Todos deben colocarse en la misma carpeta.

2) El reconocedor de colores está compuesto de 3 scripts: **'Kmeans.py', 'traeImagen.py' y 'MueveKMeans.py'**; y 2 archivos **'featuring.out' y 'multi700x490.jpg'**. Todos deben colocarse en la misma carpeta, y separados de la carpeta anterior.

Montaje

1) El reconocedor de herramientas se ejecuta en el brazo izquierdo. Este necesita una cartulina blanca no reflectante y un conjunto de herramientas, las cuales fueron diseñadas utilizando goma eva. Las herramientas se deben colocar sobre la cartulina, separadas unas de otras, y la cartulina se ubica en el sector del espacio de trabajo del brazo izquierdo del robot. Además, debemos colocar 5 depósitos para cada uno de los tipos de herramientas. Estos depósitos fueron diseñados utilizando cartón normal, y deben colocarse alrededor de la cartulina blanca, dentro del espacio de trabajo del brazo izquierdo.

2) El reconocedor de colores se ejecuta en el brazo derecho. Este necesita una cartulina blanca no reflectante, y un conjunto de figuras geométricas de distintos colores, las cuales fueron diseñadas utilizando goma eva. Las figuras geométricas se deben colocar sobre la cartulina, separadas unas de otras, y la cartulina se ubica en el sector del espacio de trabajo del brazo derecho del robot. Además, debemos colocar 3 depósitos para cada uno de los colores. Estos depósitos fueron diseñados utilizando cartón normal, y deben colocarse alrededor de la cartulina blanca, dentro del espacio de trabajo del brazo derecho.

Ejecución

1) Primero se debe ejecutar el reconocedor de herramientas. Para esto ejecutamos el siguiente

comando *'python CAGood.py'*. Aparecerá un mensaje en la pantalla que dice **'Ahora graba posiciones'**. Se debe tomar el brazo izquierdo del robot y grabar 10 posiciones, según el siguiente esquema:

```
# Espacio de búsqueda          # Depósito de objeto
########3########              ##########5##########
#################              ##########6##########
#2######0#######1              ##########7##########
#################              ##########8##########
########4########              ##########9##########
```

La posición 0 corresponde al origen, y es donde Baxter posicionara la cámara al comenzar la ejecución del script. Las posiciones 1, 2, 3 y 4 definen el cuadrilátero donde Baxter buscara el objeto escogido, siendo la posición 3 la más cercana al cuerpo del robot. Finalmente, las posiciones 5, 6, 7, 8 y 9 nos indicara el centro de masa de los depósitos donde el robot dejara las herramientas una vez que las haya tomado. Para grabar un movimiento, se debe tomar el brazo de Baxter, posicionarlo en la ubicación escogida con la cámara apuntando perpendicularmente a la mesa y presionar el botón central y redondo ubicado entre la articulación W0 y W1 de la figura x (button wheel). Una vez que las 10 posiciones han sido grabadas, se debe presionar cualquiera de los dos botones que están alrededor del botón central y redondo (above wheel o below wheel). Esto último dará inicio al script.

2) Luego, ejecutamos el reconocedor de colores. Para esto ejecutamos el siguiente comando *'python Kmeans.py'*. Aparecerá un mensaje en la pantalla que dice **'Ahora graba posiciones'**. Se debe tomar el brazo izquierdo del robot y grabar 8 posiciones, según el siguiente esquema:

```
# Espacio de búsqueda          # Depósito de objeto
########3########
#################              ##########5##########
#2######0#######1              ##########6##########
#################              ##########7##########
########4########
```

La posición 0 corresponde al origen, y es donde Baxter posicionara la cámara al comenzar la ejecución del script. Las posiciones 1, 2, 3 y 4 definen el cuadrilátero donde Baxter buscara el objeto escogido, siendo la posición 3 la más cercana al cuerpo del robot. Finalmente, las posiciones 5, 6 y 7 nos indicara el centro de masa de los depósitos donde el robot dejara las figuras geométricas una vez que las haya tomado. Para grabar un movimiento, se debe tomar el brazo de Baxter, posicionarlo en la ubicación escogida con la cámara apuntando perpendicularmente a la mesa y presionar el botón central y redondo ubicado entre la articulación W0 y W1 de la figura x (button wheel). Una vez que las 8 posiciones han sido grabadas, se debe presionar cualquiera de los dos botones que están alrededor del botón central y redondo (above wheel o below wheel). Esto último dará inicio al script.

Otros

1) Tanto las herramientas y figuras geométricas están fabricadas de goma eva. Para un correcto

funcionamiento de los ejemplos, y considerando que Baxter utiliza sus grippers, los objetos deben ser de material más sólido, por tanto, se pueden diseñar herramientas más resistentes y de otros materiales.

2) El tamaño de las herramientas debe ser adecuado para el ancho del gripper. Dependerá del ejecutor del código decidir este ancho, recordando que el gripper se posicionará sobre el centro de masa del objeto para agarrarlo.

3) El ejecutor del código podrá modificar los scripts para utilizar los vacuum grippers. Esta modificación ya corre por su cuenta.

Aplicación 3
Simulación de movimiento con Kinect

Este es un ejemplo donde Baxter simula el movimiento de los brazos del cuerpo humano, utilizando un dispositivo kinect.

Instrucciones

Instalación

1) Seguir detalladamente cada uno de los pasos de instalación de librerías del bloc de notas adjunto llamado **'Kinect.txt'**.

2) Instalar el paquete **'kinect_based_arm_tracking'** con el siguiente comando: *'git clone https://github.com/seanshi007/kinect_based_arm_tracking.git'.*

Montaje

Posicionar a un individuo al frente del dispositivo kinect, a una distancia moderada. Despejar los alrededores de Baxter, para que pueda moverse con libertad sin colisionar con obstáculos.

Ejecución

1) Este ejemplo está compuesto de 2 script: **'tf_listen_v8_puber.py'** y **'tf_listen_v8_controller.py'**. Debe abrir una terminal y usar el siguiente comando: *'roslaunch openni_launch openni.launch'*. Luego, abrir otra terminal y ejecutar *'rosrun openni_tracker openni_tracker'*. Luego, abrir otra terminar y ejecutar *'rosrun rviz rviz'*.

   Dentro del software rviz, debemos seleccionar en **Fixed Frame** la opción **'openni_depth_frame'**. Luego buscamos en **Add** y seleccionamos la opción **'TF'**. El software rviz debería mostrarnos un mapeo de nuestro cuerpo en pantalla, indicándonos que estamos listos para el siguiente paso. Ahora abrimos el script **'tf_listen_v8_puber.py'** y modificamos las variables **user_index** y **base** con los valores **'1'** y **'openni_depth_frame'** respectivamente. Posteriormente, abrimos una terminal y ejecutamos el siguiente comando *'python tf_listen_v8_puber.py'*. Finalmente, abrimos otra terminal y ejecutamos el siguiente comando *'python tf_listen_v8_controller.py'*. Con todos estos pasos, ya podremos movernos y Baxter emulara nuestros movimientos.

Aplicación 4
Pick And Learn

Este es un ejemplo de reconocimiento de un determinado tipo de objeto.

Instrucciones

Instalación

1) Se debe instalar el paquete **baxter_pickandlearn** con el siguiente comando: *'git clone https://github.com/haxelion/baxter_pickandlearn.git'.*

2) Se debe compilar el paquete anterior. Para esto volvemos a la raíz de nuestro espacio de trabajo y ejecutamos el siguiente comando: *'catkin_make'*.

Montaje

Este ejemplo está diseñado para ejecutarse con el brazo derecho de Baxter. Se debe utilizar un conjunto de objetos exactamente iguales, cuyos colores contrasten con el fondo, que podría ser una cartulina.

Ejecución

1) Se debe ejecutar el siguiente comando: *'rosrun baxter_pickandlearn pickandlearn'*.

2) Luego se debe sostener el brazo derecho del robot, e intentar tomar un objeto. Para esto se debe presionar el botón central y redondo ubicado entre la articulación W0 y W1 de la figura x (button wheel). Esto cerrará el gripper con el objeto agarrado, para posteriormente desplazar el brazo hacia el contenedor y presionar el mismo botón para soltar el objeto. Luego volvemos a la posición inicial, y repetimos este proceso al menos 3 veces, para asegurarse que Baxter es capaz de aprender la forma del objeto. Luego, se debe presionar el botón below wheel que está por debajo del botón central, para dar inicio al proceso de recogimiento.

Aplicación 5
Baxter Gato

Baxter es capaz de jugar gato a través de dos usuarios conectados en la misma red.

Instrucciones

Instalación

Se debe copiar y pegar la carpeta **baxter_gato** preferentemente en la carpeta **src** del espacio de trabajo.

Montaje

1) Al frente de Baxter se pone un tablero de gato con un tamaño aproximado a 38 x 38 cm. Frontal a cada brazo se ubica una fila de piezas que corresponde a los círculos y a las cruces respectivamente.

2) Guardar posiciones iniciales: Ejecutar el archivo **configurador_gato.py**. Se mostrará las instrucciones en la pantalla. El programa pedirá llevar el gripper a la primera posición de la fila de las piezas y luego al centro de la esquina inferior derecha del tablero (visto desde la perspectiva de Baxter). Se pedirá realizar lo anterior una vez por cada brazo. Si la posición entregada no es correcta, se pedirá ingresarla nuevamente. Finalmente se genera un archivo llamado **posiciones.py** que contiene las posiciones iniciales.

   Las posiciones deben ser copiadas del archivo **posiciones.py** y pegadas dentro del archivo **servidor_gato.py** en el constructor de la clase gato. En la parte que dice “Reemplazar aquí”.
   Importante: Se debe haber definido antes el tamaño del tablero y el largo de cada pieza (ver otros)

Ejecución

1) Se deben ejecutar tres terminales con una sesión de activa ROS. Cada terminal puede ser ejecutado en la misma o en diferentes computadoras.

2) En el primer terminal se ejecuta el archivo **servidor_gato.py**. Se pide seleccionar si el jugador 1 o el jugador 2 inicial la partida. En el segundo terminal se debe ejecutar **jugador1.py**. En el tercer terminal se debe ejecutar **jugador2.py**.

3) El jugador que inicia la partida debe ingresar la jugada. Las jugadas están numeradas del 1 al 9. Cada una de estas se destaca en la pantalla de Baxter.

4) Los jugadores van alternando sus jugadas hasta que se produzca un empate o haya un ganador.

Otros

1) Cambiar tamaño del tablero: En el archivo **configurador_gato.py**, específicamente en el

constructor de la clase **configurador_gato**, existe el objeto llamado **self.caja** generado a través de la clase **tablero_gato**. El primer y segundo valor corresponden en metros al alto y ancho respectivamente. El tercer y cuarto valor corresponden a la cantidad de separaciones. En este caso, como es un gato, tiene 3 separaciones en el largo y 3 separaciones en el ancho.

2) Cambiar largo de cada pieza: La piezas se ponen en fila frente a cada brazo. Cada una de estas piezas tienen un largo dentro de la fila. Este valor se cambiar en el archivo **configurador_gato.py**. En el constructor de la clase **configurador_gato** existe una variable que se llama **self.separacion**, esta se puede modificar con el valor correspondiente en metros.

Aplicación 6
Simulador de Producción

Baxter va tomando piezas una a una a través de un alimentador y las va ingresando en una caja subdividida en m x n partes. Luego, cuando la caja esta llena, lleva la caja a una zona de cajas llenas. Finalmente, toma una caja de la zona de cajas vacias y la deja en posición para ser llanada otra vez.

Instrucciones

Instalación

Copiar la carpeta "**linea_de_produccion**" dentro de la carpeta de **ros_ws/src.** Luego, volver a la raíz del espacio de trabajo y ejecutar el comando *catkin_make*.

Montaje

Se requiere un grupo de piezas con la misma forma puestas en fila (o siendo alimentadas a través de una cinta transportadora). Además, algunas cajas subdividas en diferentes compartimentos como se ve en la figura 2.

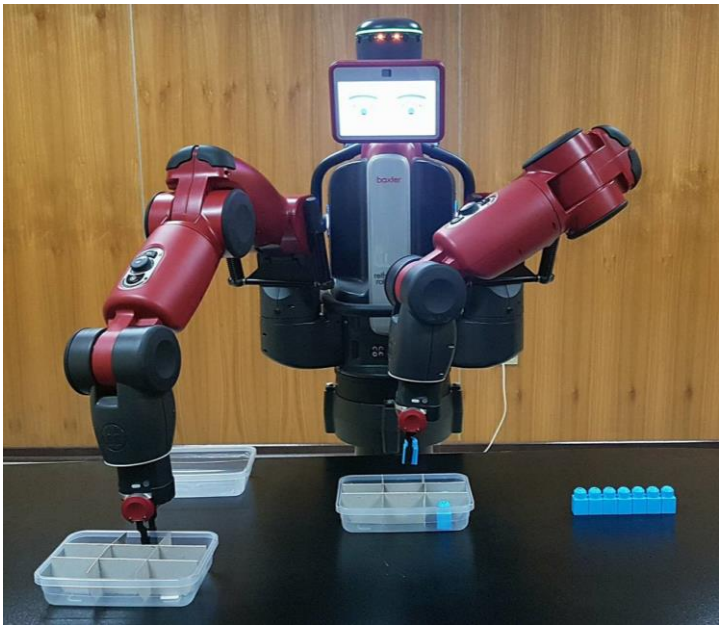


Figura 2: Montaje Simulador de Producción

Ejecución

1) Se ejecuta el archivo utilizando el siguiente comando "rosrun linea_de_produccion movimiento_linea_de_produccion". Si no aparece, dar permisos de ejecución al archivo mediante el comando chmod +x nombre_archivo.py.

2) El programa pide guardar posiciones inicial de la fila y posición inicial de la caja utilizando el brazo izquierdo. La posición inicial de la caja corresponde a la esquina inferior derecha visto

desde la perspectiva de Baxter.

3) Con el brazo derecho, según las instrucciones que aparecen en la pantalla de Baxter, seleccionar un punto de agarre de la caja que se va a empezar a llenar. Seleccionar lo mismo para el sector de cajas vacías y para el sector de cajas llenas.

Aplicación 7
Visual Servoing

Baxter encuentra y toma bolas de golf y las coloca dentro de un contenedor,

Instrucciones

Instalación

1) Se copia la carpeta **visual servoing** preferentemente dentro del entorno de trabajo. Se debe crear una carpeta llamada **Visual_Servoing** en la carpeta home (~/) de la computadora.

2) En el archivo **setup.dat** se debe ingresar el brazo que se utilizará y la distancia en metros de la cámara del brazo y la superficie donde se encuentran las pelotas.

Montaje

Al frente de Baxter se ubica hacia la derecha las bolas de golf y a la izquierda el contenedor. Las posiciones son vistas desde la perspectiva de Baxter. El montaje se ve en la figura 3.

Figura 3: Montaje Visual Servoing

Ejecución

1) Se ejecuta el archivo **visual_servoing.py** mediante el comando **python visual_servoing.py.**

2) Baxter empieza a buscar contenedor hasta encontrarlo. Luego, comienza a buscar las bolas golf hasta reconocerla. Finalmente toma las pelotas una por una y las va ingresando en el contenedor.

Anexo

Libro Recomendado para aprender ROS

Brian Gerkey, William Smart, Morgan Quigley. *Programming Robots with ROS.* O'Reilly, 2015.

Páginas Recomendadas

Robot API
http://sdk.rethinkrobotics.com/wiki/API_Reference#tab=Simulator_API

Paquete de Python “baxter_interface”
http://api.rethinkrobotics.com/baxter_interface/html/index.html

Paquete de Python “rospy”
http://docs.ros.org/lunar/api/rospy/html/

Paquete de Python “tf”
http://docs.ros.org/api/tf/html/python/

Paquete de Python “tf2_ros”
http://docs.ros.org/kinetic/api/tf2_ros/html/python/

Paquete de Python “moveit_commander”
http://docs.ros.org/lunar/api/moveit_commander/html/annotated.html