UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS DE LA COMPUTACION E INFORMÁTICA

CI2657- ROBÓTICA

Prof. Bach. Kryscia Daviana Ramírez Benavides

Tarea 3

Elaborado por:

Andrea Gómez Montero **B32896**
andrelgomez@hotmail.com

Andrés González Caldas **B12867**
andresgonzca@gmail.com

Elizabeth Gamboa Bermúdez **B22649**
erzagb@gmail.com

Jose Nixson Chaves Jimenez **B11882**
josenixsonchaves@gmail.com

5 de Setiembre del 2016

Tabla de Contenidos

Tarea 3

1 Tema

Q.Bo - Detección, aprendizaje y reconocimiento de personas y objetos.

2 Objetivos

Familiarizarse y realizar diferentes actividades con el robot Q.bo.

3 Enunciado

A cada grupo se le asignará un robot Q.bo. Luego cada grupo debe buscar información sobre el tema asignado y realizar lo siguiente:

1. Temas:
    a. OpenQbo (software).
    b. Q.bo (hardware).
    c. Odometría y PID.
    d. SLAM.
    e. Detección, aprendizaje y reconocimiento de personas y objetos.
    f. Reconocimiento de voz.

2. Cada grupo de proyecto tiene asignado un tema:
    a. Grupo 1.
    b. Grupo 1.
    c. Grupo 2.
    d. Grupo 3.
    e. Grupo 4.
    f. Grupo 5.

3. Deben generar un archivo donde expliquen y resuman el tema. Además, deben programar el robot con una función asociada al tema. Luego deben realizar una presentación a la clase.

Algunos URL de interés:

- http://thecorpora.com/blog/
- http://openqbo.org/wiki/doku.php

Referencias

Tutorial

- Conversación

Software

- Distribución de Linux
- Qbo Apps
- Firmware 1 (Control de motores, audio, sensores, boca)
- Firmware 2 (Control de energía)

Hardware

- Especificaciones de hardware
- Componentes
- Qbo Boards
- Control de hardware

Odometría y PID

- Odometría - PID
- Odometría en Qbo (ver BaseController e IMUController)

SLAM

- Usar el ASUS Xtion
- SLAM en Qbo
- Gmapping

Detección, aprendizaje y reconocimiento de personas y objetos

- Stack de Face Vision
- Reconicimiento de personas
- Reconocimiento de objetos

Reconocimiento de voz

- Reconocimiento de voz
- Sintesis de voz
- Julius Speech Recognition
- Festival Speech Synthesis

Se debe entregar un documento en Word con el desarrollo de su trabajo, siguiendo el formato del documento: **..\Robotica\Material\Ejemplos\Ejemplo Documento.doc**.

La tarea puede ser realizada en grupos de proyecto.

4 Desarrollo

A continuación se detallan tres secciones en las que se explica cómo es que se puede programar que el robot Q.bo detecte, reconozca y reconozca tanto personas como objetos.

1. Reconocimiento de objetos en el Q.bo

¿Cómo funciona el algoritmo?

El qbo_object_recognition es un paquete que contiene una implementación de un nodo capaz de reconocer objetos. Este también contiene otro nodo usado para propósitos de demostración, en el cual la persona puede preguntar al robot si puede identificar un objeto, o bien aprender un objeto por medio de comando de voz.

El nodo que permite que el Q.Bo pueda reconocer los objetos se llama orbit_client node, el cual es capaz de reconocer objetos usando imágenes estándard.

Como algoritmos que usa el Q.Bo para reconocer y clasificar los objetos es el clasificador SVM (Support Vector Machines) y Bags of Words. SVM son modelos de aprendizaje supervisados que permiten asociar con algoritmos para analizar datos y clasificarlos, de tal forma que permiten crear un análisis de regresión. Asimismo, Q.Bo crea en cada entrenamiento un archivo de vocabularios y hace algo similar como el SVM para cada objeto.

Métodos disponibles en el Q.bo.

Dentro de los métodos que contiene Q.Bo que permiten reconocer las imagenes con facilidad son las siguientes:

- Recognize Objects:
    - Recognize:
      Recognize recibe como parámetro del método input image topic, la imagen del objeto, en la cual, se identifica el objeto. Al final, el método retorna true/false si reconoce o no el objeto.
    - Recognize with stabilizer:
      Recognize with stabilizer hace casi lo mismo que el método Recognize, empero, este método cuenta con un estabilizador para reconocer con más precisión el objeto que se está reconociendo.
    - Recognize input image:
      Como indica su nombre, reconoce la imagen de entrada y retorna true/false si el objeto de la imagen se reconoce o no.
- Learn:
  El método Learn permite guardar la imagen para que el programa pueda aprender del objeto (nombre del objeto y la característica del objeto).

- Teach:
  Este método permite cargar de nuevo los nombres de los objetos y las imágenes de los objetos, para re-clasificarlos. Si encuentra una nueva imagen de un objeto entonces lo guarda en un folder principal del programa. Posteriormente de hacer el procedimiento anterior y de concluir con resultados exitosos, entonces el método devolverá true (taught), indicando que se ha enseñado con éxito y el robot está preparado para reconocer nuevos objetos.

- Update:
  El método simplemente actualiza o carga los objetos reconocidos y los guarda en el path del nodo.

Breve explicación de posibles parámetros para los métodos.

La mayoría de los casos, los parámetros que reciben los métodos son imágenes que permiten que el programa pueda aprender y reconocer un objeto. No obstante, también existen otros parámetros que son de fundamentales para aprender y reconocer fácilmente un objeto. A continuacion se muestra parametros más importantes que los metodos usan para que Q.Bo pueda reconocer los obejtos:

- Num images to capture: Indica el número de imágenes del objeto para el metodo de Learn.
- New object images path: Indica el path del folder para almacenar los nuevos objetos por aprender.
- Max time to recognize: Indica el tiempo máximo para reconocer un objeto. Este parámetro lo usa el método recognize.
- Input image topic: Indica la clasificación del objeto que se tiene que reconocer**.**

¿Cómo se usa?

Para usar el reconocedor de objetos, es necesario iniciar directamente desde la terminal con el siguiente comando:

rosrun qbo_object_recognition orbit_client

Otra forma de inicar orbit_client es ubicarse en el launcher que se ubica en el folder del launch.

roslaunch qbo_object_recognition object_recognition_demo.launch

Esto último permite iniciar todos los nodos necesarios para reconocer el objeto.

2. **Reconocimiento facial de personas en el Q.bo**

¿Cómo funciona el algoritmo?

El qbo_face_recognition es un paquete que es parte de la pila qbo_vision. Este paquete contiene la implementación de un nodo que puede aprender y reconocer un rostro, además, el paquete tiene una demostración donde el Q.bo puede identificar rostros o aprender el rostro que está detectando, esto mediante comandos de voz.

El nodo de qbo_face_recognition permite al usuario decidir hacer uso de cualquiera de los siguientes dos algoritmos:

1. Bags of Word con SVM: es un modelo de representación que se usa para procesar lenguajes naturales al igual que en la búsqueda y recuperación de información.

   El algoritmo en este modelo lo que hace es representar un documento, u oraciones como una bolsa o multiconjunto de palabras en donde cada elemento tiene asociado una multiplicidad que representa el número de veces que aparece. Además de usarse bastante en la clasificación de documentos, este modelo también se ha usado para clasificación de imágenes.

   En el campo visual, este algoritmo se usa de manera que las imágenes son tratadas de la misma manera, donde los multiconjuntos se llenan en base a las características de la imagen y las ocurrencias de estas características. También, junto a esto, se usan modelos SVM (modelos de aprendizaje supervisados) que permiten analizar los datos que reciben y poder clasificarlos.

2. PCA image y distancia euclidiana: PCA (Principal Components Analysis, por sus sigla en inglés) es una poderosa usada para diferentes áreas como el procesamiento de señales y de imágenes. Es un procedimiento lineal que lo que hace es que a partir de un conjunto de variables que pueden estar correlacionadas, convertirlas a un conjunto de variables que no estén correlacionadas de igual o menor tamaño donde las variables. Además, a los componentes se les determina una variabilidad correspondiente a los datos siendo analizados, de manera que el primer componente tenga el mayor valor; ordenando las variables por su importancia definida por este valor.

   Este método es usado en análisis de datos y también para hacer modelos predictivos. Además, mencionar lo que es la distancia euclidiana, que básicamente mide la distancia entre dos puntos en el espacio euclidiano.

Métodos disponibles en el Q.bo.

1. /qbo_face_recognition/recognize (qbo_face_msgs::RecognizeFace): dada una imagen, retorna el nombre de la persona y un booleano que indica si la persona fue reconocida por el nodo.

2. qbo_face_recognition/recognize_with_stabilizer (qbo_face_msgs::RecognizeFace): al igual que el anterior, dada una imagen, retorna el nombre de la persona y un booleano para indicar si fue reconocida o no. La diferencia con el anterior, es que este utiliza un estabilizador que permite hacer un reconocimiento con más precisión. Los atributos de este estabilizador pueden ser cambiados mediante los parámetros del nodo.
3. /qbo_face_recognition/get_name (qbo_face_msgs::GetName): al igual que los otros, retorna el nombre de la persona y el booleano que indica el reconocimiento.
4. /qbo_face_recognition/teach (qbo_face_msgs::Teach): este método lo que hace es que a partir de una carpeta o una dirección de una carpeta con nombres e imágenes, carga las imágenes y las guarda en el directorio principal de trabajo, después se cambia el booleano "taught" a un valor verdadero, indicando que está listo para reconocer los nuevos rostros.

Breve explicación de posibles parámetros para los métodos.

Estos son los parámetros de los métodos que utiliza, todos poseen un valor predeterminado pero pueden ser modificados para hacer un reconocimiento más exacto, o por el contrario menos exacto, esto para reconocer patrones más fácilmente ya que no siempre los rostros son iguales a las imágenes almacenadas.

`/qbo_face_recognition/faces_path` (string, default: PKG_PATH/faces/faces_db)
Ruta con las imágenes de entrenamiento.

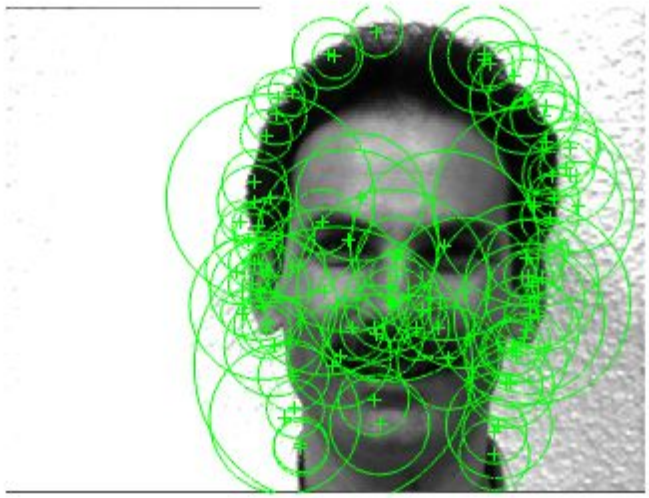
/qbo_face_recognition/uptate_path (string, default: PKG_PATH/faces/new_faces)
Ruta donde se almacenan los nuevos rostros. Teach service

/qbo_face_recognition/recognition_type (int, default: 1)

/qbo_face_recognition/stabilizer_threshold (int, default: 4)
limite en que el estabilizador tendrá una correcta identificación

/qbo_face_recognition/stabilizer_max (int, default: 7)
Valor máximo del estabilizador.

/qbo_face_recognition/num_of_desc_per_face_ (int, default: 50)
Número de descripciones que se extraen del rostro.



/qbo_face_recognition/bow_certainty_threshold (double, default: 0.05)
Certeza de clasificadores de los algoritmos, el valor esta entre 0 y 1

/qbo_face_recognition/descriptors_match_threshold (double, default: 0.23)
descriptores SURF umbral partido

/qbo_face_recognition/linear_kernel (bool, default: false)
Número de descripciones que se extraen del rostro.

/qbo_face_recognition/pca_dimension (int, default: 40)
Número de valores propios que utiliza el algoritmo de análisis.

/qbo_face_recognition/pca_image_height (int, default: 100)
Altura de la cara para el reconocimiento facial.

¿Cómo se usa?

Iniciar el nodo de qbo_face_recognition
puede iniciarse directamente desde la terminal con el comando:

rosrun qbo_face_recognition qbo_face_recognition

También con el lauch ubicado el folder de "launch/", el cual permite ingresar valores deseados para no usar los predeterminados, esto de la siguiente manera.

roslaunch qbo_face_recognition qbo_face_recognition.launch

Iniciar la demo es mas facil, solo necesita utilizar este lauch, y se ejecutaran todos los nodos necesarios.

roslaunch qbo_face_recognition qbo_face_recognition_demo.launch

Funciones relacionadas.

El paquete que contiene la funcionalidad de reconocimiento facial (qbo_face_recognition) es uno de los componentes del paquete un paquete de visión facial (qbo_face_vision) el cual consiste de una serie de algoritmos de visión de la computadora para seguir, rastrear y reconocer caras. Por esto, a continuación se explican brevemente los otros paquetes que formar parte del de visión facial, los cuales contienen funcionalidades relacionadas a la de reconocimiento facial:

I. qbo_face_tracking: Al usar este paquete es posible detectar y rastrear una cara que recibe como entrada por medio de los algoritmos de visión de la computadora y luego publica tanto la cara en la imagen como un mensaje con su posición y distancia aproximada de la cámara.

Este paquete usa 3 métodos diferentes para encontrar y seguir caras:

A. Detector facial OpenCV de Haar para encontrar la cara en la imagen.

B. Un rastreador CamShift para seguir rastreando la cara, usando un histograma de colores y el color de piel de la cara.

La principal ventaja de utilizar este filtro se basa en su desempeño computacional, ya que el tiempo que se dura en computar un frame es significativamente menor al detector facial Haar. Además, con este filtro la la tasa media del frame aumente, lo que nos permite un mejor rastreamiento del movimiento de la cabeza.

C. Un filtro de Kalman para estimar y predecir la posición y velocidad de la cara.

Este filtro es muy útil para estimar la posición y velocidad de la cámara es frames individuales en caso de que la cara no sea detectada.

II. qbo_face_following: Al utilizar este paquete se tiene que al recibir una posición de la cara como parámetro, manda órdenes de mover a la cabeza y la base de modo que siga el rostro.

Este paquete se suscribe a un mensaje de posición y distancia de la cara, usa un id de la persona interno para el control y publica comandos de movimiento.

III. qbo_face_msgs: Define los mensajes y servicios que ROS usa los nodos del qbo_face_vision.

En el folder de msg hay un archivo llamado FacePosAndDist.msg, la cual esta publicada por qbo_face_tracking e indica la posición de la cara rastreada en pixeles, distancia desde la camara ademas del tamaño de la imagen y el tipo de rastreo usado (si se utilizó el algoritmo de Haar Cascade o el de CAM shift filter). En la figura 1, se puede ver el contenido del archivo FacePosAndDist.msg.

```
Header header
float32 u
float32 v
float32 distance_to_head
int32 image_width
int32 image_height
bool face_detected
string type_of_tracking
```

Figura 1. Se muestran las variables que el archivo FacePosAndDist.msg contiene.

Buscar videos de ejemplos para enseñar en la clase.

http://www.antena3.com/noticias/tecnologia/robot-reconocimiento-facial-mismo_201111305749a40e4beb2888806613ee.html video de Q.BO reconociendo imagenes.

https://www.youtube.com/watch?v=T4z11lIKnDU **video Qbo reconociendo y aprendiendo objetos y personas.**

http://www.actualidadgadget.com/qbo-un-dispositivo-que-da-la-nota/

Referencias

[1] OpenQboWiki, qbo_object_recognition. Official Site, visitado: 3 de Setiembre del 2016. URL: http://openqbo.org/wiki/doku.php?id=qbo_apps:ros_pack:qbo_object_recognition

[2] OpenQboWiki, qbo_face_following. Official Site, visitado: 3 de Setiembre del 2016. URL: http://openqbo.org/wiki/doku.php?id=qbo_apps:ros_pack:qbo_face_following

[3] OpenQboWiki, qbo_face_recognition. Official Site, visitado: 3 de Setiembre del 2016. URL: http://openqbo.org/wiki/doku.php?id=qbo_apps:ros_pack:qbo_face_recognition

[4] OpenQboWiki, qbo_face_msgs. Official Site, visitado: 3 de Setiembre del 2016. URL: http://openqbo.org/wiki/doku.php?id=qbo_apps:ros_pack:qbo_face_msgs

[5] OpenQboWiki, qbo_face_tracking. Official Site, visitado: 3 de Setiembre del 2016. URL: http://openqbo.org/wiki/doku.php?id=qbo_apps:ros_pack:qbo_face_tracking