Introducción a Robotic Operating System

Sensado

Martín Llofriu y Gonzalo Tejera
Facultad de Ingeniería :: Instituto de Computación

mina
network management | artificial intelligence

Contenido

- Obtención de imágenes
- Detección de patrones artoolkit
- Aspectos de rendimiento
- Visualización
- Calibración de la cámara
- Grabado de datos con la cámara calibrada
- Calibración de la estimación de posición
- Grabado de datos calibrados

Práctico: Obtención de Imágenes

Paquete usb_cam nos permite obtener imágenes de la cámara

Visitar http://wiki.ros.org/usb_cam

- Determinar ubicacíon del código fuente de usb_cam
- Determinar nodos existentes
- Determinar parámetros necesarios

Práctico: Obtención de Imágenes

- roslaunch nos permite ejecutar varios nodos a la vez
- Pero también es útil para especificar parámetros a pasar a un nodo

```
<launch>
  <node name="talker" pkg="rospy_tutorials" type="talker">
          <param name=”pametro1” value=”5”/>
          <param name=”pametro2” value=”hola”/>
  </node>
</launch>
```

Práctico: Obtención de Imágenes

- Crear un archivo roslaunch que:
  - Ejecute el nodo correspondiente de usb_cam
  - Pase los paramétros necesarios al nodo

- Para comprobar:
  - rostopic list/echo permite comprobar que se están publicando los tópicoss
  - Paquete image_view permite ver las imágenes de un tópico - http://wiki.ros.org/image_view

Práctico: detección de patrones artoolkit

Utilizaremos el paquete ar_pose para detectar patrones del tipo artoolkit

Práctico: detección de patrones artoolkit

- Pasos a seguir:
  - Investigar el nodo ar_pose - http://wiki.ros.org/ar_pose
    - Investigar los archivos roslaunch en el paquete

Práctico: detección de patrones artoolkit

- Pasos a seguir:
  - Modificar el archivo roslaunch anterior para incluir un nodo ar_pose
    - Abrir los archivos referenciados en los parámetros pasados al nodo ar_pose
    - los archivos de patrones está en la carpeta materiales

Práctico: detección de patrones artoolkit

- Pasos a seguir:
  - Utilizar el patrón impreso para probar la detección
    - rostopic list/echo

Aspectos de rendimiento

Usando 30fps y detección artoolkit, si ejecutamos **top** o **uptime** podemos ver que el sistema está altamente cargado (si no se congelo aún)

Para mejorar el rendimiento:

- Bajar fps (1-5)
- Bajar resolución (160x120)

Aspectos de rendimiento

Métricas a tener en cuenta:

- Carga del sistema (promedio de procesos en la cola de ejecución)
- Lag aparente en la detección de las marcas (importante para control)
  - Fallos de página (page faults): ps -o min_flt,maj_flt <pid>

Nuestro sistema estable en el largo plazo?

Aspectos de rendimiento

Una medida adicional: **Nodelets**

Los nodelets permiten ejecutar varios nodos en un mismo proceso:

- Evitan copia y uso del sistema operativo para el envío de mensajes
  - Menos lag y más throughput
  - Menos page faults??
- Ambos paquetes deben ser “nodelet aware”
- Cada paquete exporta nodelets que pueden ser cargados (loaded) en el espacio de un proceso existente

Práctico opcional: utilizar nodelets

Pasos a seguir:

- Instalar una copia modificada de **usb_cam: https://github.com/mllofriu/usb_cam**

Práctico opcional: utilizar nodelets

Pasos a seguir:

- Crear una copia del archivo roslaunch y modificarlo para que use nodelets:
  - Ejecutar un nodo nodelet del paquete nodelet con argumentos “manager”
  - Cambiar cada nodo a nodelet del paquete nodelet y pasar como argumentos “load <nombre_del_nodelet>”
    - Los nombres de los nodelets están en archivos plugins.xml dentro de los respectivos paquetes

Práctico opcional: utilizar nodelets

Ejemplo:
http://wiki.ros.org/nodelet/Tutorials/Running%20a%20nodelet

Información adicional: creando nodelets

Recurso:
http://wiki.ros.org/nodelet/Tutorials/Porting%20nodes%20to%20nodelets

Un ejemplo:

https://github.com/mllofriu/usb_cam

Práctico: Visualización

Utilizaremos el paquete rviz para visualizar las imágenes y los patrones detectados

Pasos a seguir (en placa):

- Ejecutar camara+ar_pose

Práctico: Visualización

Pasos a seguir (en pc):

- Exportar ROS_MASTER_URI
- Ejecutar rviz: rosrun rviz rviz
- Incluir una cámara
- Incluir un “visual marker”

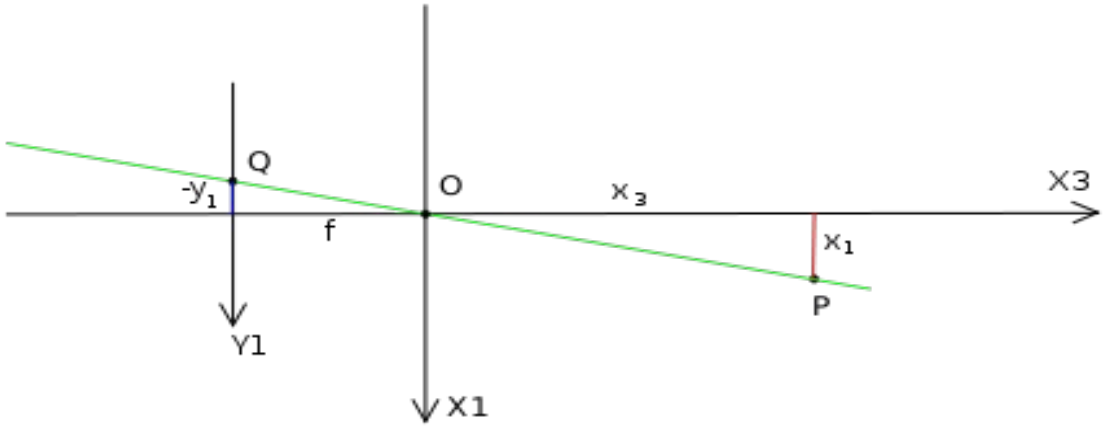
Práctico Opcional: calibración de la cámara

- Recurso: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration

Práctico Opcional: calibración de la cámara

En la estimación de la posición de las marcas, ar_pose utiliza los “parámetros intrínsecos” de la cámara.

Estos parámetros incluyen la distancia focal en *x* e *y.*



Práctico Opcional: calibración de la cámara

También incluyen información sobre la distorción radial generada por el lente.

Práctico Opcional: calibración de la cámara

- Usuaremos el paquete camera_calibration
- Ejemplo de ejecución: rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.108 image:=/camera/image_raw camera:=/camera
  - 8x6 es el tamaño del patrón (usar uno menos en cada dimensión)
  - 0.108 es el tamaño del cuadrado en metros
  - /camera/image_raw y /camera refieren a los tópicos donde se publica la informacíon

Práctico Opcional: calibración de la cámara

- Pasos a seguir:
  - Conectar la cámara a la pc
  - Ejecutar usb_cam
  - Ejecutar el nodo de calibración
  - Mover el patron (o la cámara) para tomar diferentes imágenes
  - Una vez tomadas las imágenes, calibrar y guardar
  - El archivo de calibración es salvado automáticamente

Práctico Opcional: calibración de la cámara

- De ahora en adelante, ejecutaremos usb_cam direccionandolo a ese archivo de calibración

Práctico: grabar datos

Grabaremos datos de distancia con la cámara calibrada para diferentes valores conocidos de *x* e *y* (en el plano)

Práctico: grabar datos

Pasos a seguir:

- Ejecutar usb_cam y ar_pose
- Colocar la cámara a una distancia conocida en *x*
- Utilizar rosbag para grabar varias capturas del tópico /visualization_marker o /ar_pose_markers
- Variar la distancia en *x* y repetir
- Repetir el experimento variando solo *y*

Práctico: calibración de la estimación

Trataremos de comparar los valores estimados de posición con los reales conocidos.

Pasos a seguir:

- Desarrollar un programa (sugerido python) que escuche el tópico grabado y calcule el promedio de la posición a través del tiempo
- Varianza puede ser tambien interesante
- Graficar los promedios obtenidos para los diferentes set de datos

Práctico: calibración de la estimación

Existe una relación lineal entre las coordenadas estimadas?

Cuál es el coeficiente?

Cómo podemos arreglarlo?

Práctico: calibración de la estimación

Opciones:

- Modificar el código de ar_pose para incluir esta corrección
- Crear un nodo que realize la corrección y publique un set diferente de datos
- Modificar las dimensiones conocidas de la marca para compensar

Práctico: grabado de datos calibrados

Grabar datos utilizando la solución calibrada y comprobar nuevamente la relación entre lo estimado y lo real