Localization System using the Microsoft Kinect

David Sanders

The localization system using the Kinect for the drone was part of my 2014 summer project, and was developed on a Ubuntu 12.10 64-bit machine with ROS hydro. This document intends to provide guide, advice and other comments for users and developers of this system.

1 Getting Started

This section will go through the requirements for running the system.

1.1 The Robot Operating System

The localization system uses the Robot Operating System (ROS), so ROS will need to be installed. ROS is supported on Ubuntu, and I used ROS hydro on a Ubuntu 12.10 64-bit machine. Setting up the development environment can be done through the ROS tutorials, listed below among other useful links.

Installing the latest version of ROS:
`http://wiki.ros.org/ROS/Installation`

Installing ROS hydro (instead of latest version):
`http://wiki.ros.org/hydro/Installation`

ROS wiki:
`http://wiki.ros.org/`

ROS tutorials:
`http://wiki.ros.org/ROS/Tutorials`

1.2 Markers

Special markers called AR Markers were used to track the drone. These markers can be handmade, but you will need a pattern file for each marker. Examples of marker images and pattern files corresponding to these images can be found in `data/my_markers/`.

The links to software for making AR Markers and online pattern file makers are provided below.

About AR Markers:
`http://www.artoolworks.com/support/library/Creating_and_training_new_ARToolKit_markers`

Creating AR Markers:

http://www.buildar.co.nz/buildar-free-version/

Creating Pattern Files:
http://flash.tarotaro.org/blog/2008/12/14/artoolkit-marker-generator-online-released/

1.3 Camera

The Microsoft Kinect was used as the camera for this tracking system. You will need to install drivers for running the Kinect on Ubuntu, and the ROS package `openni_launch` to run the Kinect with ROS. Finally, to track the markers, install the ROS packages `usb_cam`, `ar_pose` and the ROS package `ar_kinect` if you are using the Kinect.

If you are not using the Kinect, you will only need the ROS packages `usb_cam` and `ar_pose`. However, you will also need a camera calibration file.

Installing drivers for running the Kinect on Ubuntu:
http://igorbarbosa.com/articles/how-to-install-kin-in-linux-mint-12-ubuntu/

openni_launch wiki:
http://wiki.ros.org/openni_launch

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

ar_pose wiki:
http://wiki.ros.org/ar_pose

ar_kinect wiki:
http://wiki.ros.org/ar_kinect

2 Running the system

2.1 Initialize ROS

Initialize ROS with the command
```
roscore
```

2.2 Run the Kinect

Run the Kinect with the command
```
roslaunch openni_launch openni_launch
```

2.3 Reconfigure from Point Cloud to Registered Point Cloud

Change from point cloud to registered point cloud with the command
```
rosrun rqt_reconfigure rqt_reconfigure
```

A window will appear, and from the list on the left hand side, open the drop down menu **camera** and click on **drivers**. From the parameters that appear on the right, click on the **depth_registration** checkbox.

2.4 Detect Markers

Initiate marker tracking and data publishing with the command
`roslaunch drone kinect.launch`

The launch file is `launch/kinect.launch`, and by default uses the configuration file `data/kinect_ar_markers` and the pattern files indicated in this configuration file.

2.5 Run Localization System

Start running the system with the command
`roslaunch drone marker_tracker.launch`

The launch file is `launch/marker_tracker.launch`, and there are three configuration files it sets as default parameters - `.client.config`, `server.config` and `marker.config` which are located in `config/`.
Example of client/server config. file:

```
#Comment
STATE ON #use OFF if not using client/server
HOST 192.168.1.1
PORT 7890
NETWORK TCP #TCP or UDP
```

Example of marker config. file:

```
#The number of patterns to be recognized
2

# Pattern 1
# Rotation Matrix mapping drone from marker
0 1 0
0 0 -1
-1 0 0
# Distances from centre of drone to marker in drone's frame
# x in metres
-0.132
# y in metres
-0.052
# z in metres
-0.255

# Pattern 2
# Rotation Matrix mapping drone from marker
0 1 0
0 0 -1
-1 0 0
# Distances from centre of drone to marker in drone's frame
# x in metres
```

```
0.132
# y in metres
-0.052
# z in metres
-0.255
```

3 Areas of improvement and other comments

3.1 Markers not being detected

If you are experiencing detecting markers, it may be a good idea to reset everything from step 2.1. You can check if the camera is working properly with the command
`rosrun image_view image_view image:=/camera/rgb/image_color`

3.2 The my_matrix library

This library contains self implemented matrices and matrix operations. Although these should be replaced with proper matrix libraries, they take up only about 5% of the total computation time.

3.3 Current setup

The data processing is implemented in the `processDataToDrone()` function found in `src/marker_tracker.cp`
Currently, there are four markers placed on the drone, and the roll, pitch and yaw angles are computed.