bebop_autonomy Documentation

Release indigo-devel

Mani Monajjemi

September 10, 2015

Contents

bebop_autonomy is a ROS (Robot Operating System) driver for Parrot Bebop drone (quadrocopter), based on Parrot's official ARDroneSDK3. This driver has been developed in Autonomy Lab of Simon Fraser University by Mani Monajjemi.

[Source Code] [Support] [Bug Tracker] [Developer Forum]

CHAPTER 1

Features and Roadmap

Feature	Status	Notes
Core piloting	Yes	
H264 video decoding	Yes	Enhancement: #1
ROS Camera Interface	Yes	
Nodelet implementation	Yes	
Publish Bebop states as ROS topics	Yes	
Dynamically reconfigurable Bebop settings	Yes	*Configuring the Drone*
Inline build of ARDroneSDK3	Yes	Enhancement: #2
Bebop In The Loop tests	Yes	*Tests*
Joystick teleop demo	Yes	
TF Publisher	No (Planned)	#3
Odometry Publisher	No (Planned)	#4
Provide ROS API for on-board picture/video recording	No (Planned)	#5
GPS Support	Partial	Not fully tested
Mavlink Support	No	
Binary Release	No	
Support for Parrot Sky Controller	No	

CHAPTER 2

Table of Contents

2.1 Changelog and Release History

2.1.1 Changelog for package bebop_autonomy

0.2.0 (2015-09-10)

- Finalized documentation
- Remove bebop_autonomy's dependency to image_view
- Imrovements to code autogeneration scripts.
- CLAMP values for cmd_vels and anim_id
- Added contents to almost all doc pages
- Bebop In The Loop tests (first revision)
- Fixed more style (lint) issues
- Finalized the first revision of tests
- Add autogenerated docs for Settings, Topics and Params
- Contributors: Mani Monajjemi

0.1.2 (2015-09-05)

- Move 'state' params to their own param namespace
- Add missing unzip dep to package.xml
- Contributors: Mani Monajjemi

0.1.1 (2015-09-04)

- Add support for downloading and building ARDroneSDK3 during the build process
- Add flattrim, flip and navigatehome interfaces
- Add forward declaration to classes where it is possible

- Major bug fixes and improvements - Dynamic Reconfigure: Convert all two state int_t values to enum - Fix the private nodehandle bugs in State and Settings handlers - Fix the data flow of Settings between rosparam and dynamic reconfigure and bebop - Fix SDK enum types in C (I32 instead of U8) - Add Start/Stop streaming to Bebop interface class
- Add bebop_nodelet launch with image_view
- Organized DynR configs into groups + Moved the autogeneration report to a seperated file + build speed improvements
- Dynamically reconfigurable Bebop settings
- Add support to enable publishing of a specific State
- Add support to propogate states from bebop to ROS
- Auto-generated .msg and .h files based on libARCommands XML files
- New threading model for data retreival and publishing - Nodelet now manages its own thread to receive frames from Bebop - GetFrame() function abstracts all sync to access the rgb frame - All subscribers send commands to the Bebop in their callbacks
- Integreate ARSAL logs into ROS_LOG - Fix sync issues between frame grabber and publisher
- Improve video decode/publish pipeline - Adopt frame decoding from official examples - Thread safe access to raw frame ptr - Synchronised frame decoding and publishing
- Proof of concept ROS driver for bebop drone
- Contributors: Mani Monajjemi

2.1.2 Changelog for package bebop_tools

0.2.0 (2015-09-10)

- Move image_view nodelet demo to bebop_tools package
- Contributors: Mani Monajjemi

0.1.2 (2015-09-05)

- Initial release of joystick teleop for bebop_autonomy
- Contributors: Mani Monajjemi

0.1.1 (2015-09-04)

2.1.3 Changelog for package bebop_autonomy_msgs

0.2.0 (2015-09-10)

- Contributors: Mani Monajjemi

0.1.2 (2015-09-05)

- Contributors: Mani Monajjemi

0.1.1 (2015-09-04)

- Auto-generated .msg and .h files based on libARCommands XML files
- Contributors: Mani Monajjemi

2.2 Installation

2.2.1 Compiling From Source

Pre-requirements:

- ROS *Indigo* or *Jade* (Only tested on *Ubuntu*)
- Internet connection
- Ubuntu packages: `build-esstential`, `python-rosdep`, `python-catkin-tools`
- Basic familiarity with building ROS packages

```
$ sudo apt-get install build-essential python-rosdep python-catkin-tools
```

To compile from source, you need to clone the source code in a new or existing `catkin` workspace, use `rosdep` to install dependencies and finally compile the workspace using *catkin*. The following commands demonstrate this procedure in a newly created `catkin` workspace.

```
# Create and initialize the workspace
$ mkdir -p ~/bebop_ws/src && cd ~/bebop_ws
$ catkin init
$ git clone https://github.com/AutonomyLab/bebop_autonomy.git src/bebop_autonomy
# Update rosdep database and install dependencies
$ rosdep update
$ rosdep install --from-paths src -i
# Build the workspace
$ catkin build -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

The first time build may take up to 15 minutes, since ARDroneSDK3's build script downloads and compiles ~20 packages from Internet.

Note: On ROS *Jade*, if you receive a `rosdep` error about `ros-jade-joy-teleop` package, please clone `teleop_tools` in your workspace:

```
$ cd ~/bebop_ws/src
$ git clone https://github.com/ros-teleop/teleop_tools.git
# Do rosdep steps again
```

2.3 Running the Driver

You can run *bebop_autonomy* either as a ROS Nodelet or as a standalone ROS Node. The former is recommended if you intend to perform any kind of processing on Bebop's video stream.

Note: If you compile the driver form source, do not forget to source your catkin workspace prior to running the driver. (i.e. `source ~/bebop_ws/devel/setup.[bash|zsh]`)

Note: Ensure that your Bebop's firmware is at least **2.0.29** and your computer is connected to Bebop's wireless network.

2.3.1 Running the driver as a Node

The executable node is called `bebop_driver_node`. It's recommended to run the Node in its own namespace and with default configuration. The driver package comes with a sample launch file `launch/bebop_node.launch` which demonstrates the procedure.

```
$ roslaunch bebop_autonomy bebop_node.launch
```

Listing 2.1: bebop_node.launch

```
<?xml version="1.0"?>
<launch>
    <group ns="bebop">
        <node pkg="bebop_autonomy" name="bebop_driver" type="bebop_driver_node" output="screen">
            <rosparam command="load" file="$(find bebop_autonomy)/config/defaults.yaml" />
        </node>
    </group>
</launch>
```

2.3.2 Running the driver as a Nodelet

To run the driver as a ROS Nodelet, you need to first run a Nodelet manager, then load the driver's Nodelet (`bebop_autonomy/BebopDriverNodelet`) in it, along with other Nodelets that need to communicate with the driver. *bebop_tools/launch/bebop_nodelet_iv.launch* is a sample launch file that demonstrates these steps by visualizing Bebop's video stream using an instance of image_view/image Nodelet. Similar to *bebop_node.launch*, it also runs everything in a namespace and loads the default configuration.

```
$ roslaunch bebop_tools bebop_nodelet_iv.launch
```

Listing 2.2: bebop_tools/launch/bebop_nodelet_iv.launch

```
<?xml version="1.0"?>
<launch>
    <!-- include the nodelet launch file from bebop_autonomy -->
    <include file="$(find bebop_autonomy)/launch/bebop_nodelet.launch" />
    <!-- use the same nodelet manager and namespace, then load image_view nodelet -->
    <group ns="bebop">
       <node pkg="nodelet" type="nodelet" name="bebop_image_view_nodelet"
          args="load image_view/image bebop_nodelet_manager">
          <remap from="image" to="image_raw" />
        </node>
    </group>
</launch>
```

2.4 Sending Commands to Bebop

Note: `bebop_tools` package comes with a launch file for tele-operating Bebop with a joystick using ROS joy_teleop package. The configuration file (key-action map) is written for Logitech F710 controller and located in `/bebop_tools/config` folder. Adapting the file to your own controller is straight-forward. To teleop Bebop while the driver is running execute `roslaunch bebop_tools joy_teleop.launch`.

Listing 2.3: bebop_autonomy/launch/bebop_nodelet.launch

```
<?xml version="1.0"?>
<launch>
    <group ns="bebop">
        <!-- nodelet manager -->
        <node pkg="nodelet" type="nodelet" name="bebop_nodelet_manager" args="manager" output="screen
        <!-- bebop_nodelet -->
        <node pkg="nodelet" type="nodelet" name="bebop_nodelet"
          args="load bebop_autonomy/BebopDriverNodelet bebop_nodelet_manager">
            <rosparam command="load" file="$(find bebop_autonomy)/config/defaults.yaml" />
        </node>
   </group>
</launch>
```

2.4.1 Takeoff

Publish a message of type `std_msgs/Empty` to `takeoff` topic.

```
$ rostopic pub --once std_msgs/Empty [namespace]/takeoff
```

2.4.2 Land

Publish a message of type `std_msgs/Empty` to `land` topic.

```
$ rostopic pub --once std_msgs/Empty [namespace]/land
```

2.4.3 Emergency

Publish a message of type `std_msgs/Empty` to `reset` topic.

```
$ rostopic pub --once std_msgs/Empty [namespace]/reset
```

2.4.4 Piloting

To move Bebop around, publish messages of type geometry_msgs/Twist to *cmd_vel* topic while Bebop is flying. The effect of each field of the message on Bebop's movement is listed below:

```
linear.x  (+)      Translate forward
          (-)      Translate backward
linear.y  (+)      Translate to left
          (-)      Translate to right
linear.z  (+)      Ascend
          (-)      Descend
angular.z (+)      Rotate clockwise
          (-)      Rotate counter clockwise
```

Acceptable range for all fields are `[-1..1]`. The drone executes the last received command as long as the driver is running. This command is reset to when *Takeoff*, *Land* or *Emergency* command is received. To make Bebop hover and maintain its current position, you need to publish a message with all fields set to zero to `cmd_vel`.

Note: TODO: Add the unit and min/max mapping for each field.

2.4.5 Moving the Virtual Camera

To move Bebop's virtual camera, publish a message of type geometry_msgs/Twist to *camera_control* topic. `angular.y` and `angular.z` fields of this message set **absolute** tilt and pan of the camera in **degrees** respectively.

Warning: The API for this command is not stable. We plan to use `JointState` message in feature versions.

```
angular.y (+)      tilt down
          (-)      tilt up
angular.z (+)      pan left
          (-)      pan right
```

2.4.6 GPS Navigation

Warning: Not fully integrated/tested yet.

2.4.7 Flat Trim

Error: Test fails, probably not working.

Publish a message of type `std_msgs/Empty` to `flattrim` topic.

```
$ rostopic pub --once std_msgs/Empty [namespace]/flattrim
```

2.4.8 Flight Animations

Warning: Be extra cautious when performing any flight animations, specially in indoor environments.

Bebop can perform four different types of flight animation (flipping). To perform an animation, publish a message of type *std_msgs/UInt8* to *flip* topic while drone is flying. The *data* field determines the requested animation type.

```
0       Flip Forward
1       Flip Backward
2       Flip Right
3       Flip Left
```

2.5 Reading from Bebop

2.5.1 Camera

The video stream from Bebop's front camera is published on `image_raw` topic as `sensor_msgs/Image` messages. *bebop_autonomy* complies with ROS camera interface specifications and publishes camera information and

calibration data to `camera_info` topic. Due to limitations in Parrot's ARDroneSDK3, the quality of video stream is limited to **640 x 368 @ 30 Hz**.

To set the location of camera calibration data, please check this page: Configuring Bebop and the Driver

2.5.2 States (aka Navdata)

Unlike Parrot ARDrone, Bebop does not constantly transmit all on-board data back to the host device with high frequency. Each state variable is sent only when its value is changed. In addition, the publication rate is currently limited to **5 Hz**. The driver publishes these states **selectively** and when **explicitly** enabled through a ROS parameter. For example setting `~states/enable_pilotingstate_flyingstatechanged` parameter to `true` will enable the publication of flying state changes to topic `states/ARDrone3/PilotingState/FlyingStateChanged`. List of all such parameters and their corresponding topics and message types are indexed in the following pages:

Common States `autogenerated/common_states_param_topic`

Bebop-specific States `autogenerated/ardrone3_states_param_topic`

2.6 Configuring Bebop and the Driver

2.6.1 Driver Parameters

Following parameters are set during driver's startup:

~reset_settings

Setting this parameter to `true` will reset all Bebop configurations to factory defaults. Default value is `false`.

~camera_info_url

Sets the location of the camera caliberation data. Default is empty string.

~camera_frame_id

Sets the `frame_id` of camera and image messages. Default value is `camera`.

2.6.2 Dynamically Reconfigurable Parameters for Bebop

Following ROS parameters change Bebop's settings. They can be tweaked during runtime using dynamic reconfigure GUI. Setting *~reset_settings* parameter to `true` will reset all these settings to factory defaults.

`autogenerated/ardrone3_settings_param`

2.7 Contribute

2.7.1 Contribute to bebop_autonomy

You can contribute to *bebop_autonomy* by:

- Reporting bugs using driver's Issue Tracker on Github.
- Submitting patches, new features, sample codes, documentation and supplementary materials (i.e. launch and configuration files) as Github Pull Requests.
  - Please check current open issues and *Features and Roadmap* section for a list of known bugs and feature request.
- Joining driver's developers forum and participate in technical discussions on new features, bugs and roadmap.

2.7.2 List of Contributers

- Mani Monajjemi

2.7.3 Acknowledgments

- Mike Purvis for his help with designing the initial architecture of the driver.

2.8 Frequently Asked Questions

2.8.1 Is *bebop_autonomy* based on *ardrone_autonomy*?

No. ardrone_autonomy is based on Parrot's legacy SDK for AR-Drone 1.0 and 2.0, while *bebop_autonomy* uses Parrot's new SDK for its third generation drones. Since these two SDKs and their underlying protocols are totally different and incompatible, we had to develop *bebop_autonomy* from scrath.

2.8.2 Is *bebop_autonomy* compatible with *ardrone_autonomy*?

Not completely.

- Topic names, types and coordinate frame conventions for core piloting tasks are identical, however there is no explicit namespacing (i.e. `takeoff` instead of `ardrone/takeoff`)
- *bebop_autonomy* does not expose services for *Flight Animations* or *Flat Trim*; topics are used instead.
- Front camera video stream is published on `image_raw` topic only.
- Parameter names, types and effects are different.
- AR-Drone *Navdata* is replaced by Bebop *States* (see *States (aka Navdata)*)

2.9 Under The Hood

This page contains information about the architecture of the driver and different techniques used for its development.

2.9.1 Automatic Code Generation

TBA

2.9.2 Threading Model

TBA

2.9.3 Publishing States

TBA

2.9.4 Configuring the Drone

TBA

2.9.5 Tests

TBA

2.10 License

2.10.1 Parrot ARDrone3 SDK

2.10.2 bebop_autonomy

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of [project] nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search