bebop_autonomy Documentation

Release indigo-devel

Mani Monajjemi

Jun 13, 2017

Contents

bebop_autonomy is a ROS (Robot Operating System) driver for Parrot Bebop 1.0 and 2.0 drones (quadrocopters), based on Parrot's official ARDroneSDK3. This driver has been developed in Autonomy Lab of Simon Fraser University by Mani Monajjemi and other contributers (*List of Contributers*). This software is maintained by Sepehr MohaimenianPour (AutonomyLab, Simon Fraser University), *Thomas Bamford* (Dynamic Systems Lab, University of Toronto) and Tobias Naegeli (Advanced Interactive Technologies Lab, ETH Zürich).

[Source Code] [ROS wiki page] [Support] [Bug Tracker]

CHAPTER 1

Features and Roadmap

Feature	Status	Notes
SDK Version	3.10.1	Since v0.6
Support for Parrot Bebop 1	Yes	Tested up to Firmware 3.3
Support for Parrot Bebop 2	Yes	Tested up to Firmware 3.9
Support for Parrot Disco FPV	No	Not tested (help wanted)
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*
Use parrot_arsdk instead of building ARSDK3 inline	Yes	Since v0.6: #75
Bebop In The Loop tests	Yes	*Tests*
Joystick teleop demo	Yes	sec-pilot-teleop
TF Publisher	Yes	Since v0.5 (*TF*)
Odometry Publisher	Yes	Since v0.5 (*Odometery*)
Provide ROS API for on-board picture/video recording	Yes	Since v0.4.1 (*Take on-board Snapshot*)
GPS Support	Yes	Since v0.6 (*GPS*)
Support for 720p streaming	Yes	Since v0.6
Mavlink Support	No	
Binary Release	No	
Support for Parrot Sky Controller	No	

CHAPTER 2

Table of Contents

Changelog and Release History

Changelog for package bebop_driver

0.6.0 (2016-11-02)

- Use Parrot ARSDK as a thirdparty dependency - Instead of building the SDK inline, bebop_driver now utilizes the catkin version of Parrot ARSDK available at https://github.com/AutonomyLab/parrot_arsdk and in a binary form via the ROS buildfarm: ros-<DISTRO>-parrot-arsdk
- Rename liblibbebop to libbebop
- Full SDK 3.10.x support - ROS API change: states/ARDrone3/.. topics renamed to states/ardrone3/..
- Disabling video stabilization works with Firmware 3.9 (tested on Bebop

2.

- Camera control API fix for SDK 3.10.x
- Refactor H264 decode class to support dynamic picture size - Add support for determining picture size from the stream - Add dynamic buffer re-allocation on picture size change
- Use av_frame_alloc() instead of the depr. avcodec_alloc_frame()
- Fix inconsistent SDK 3.10.1 key values
- Update autogenerated msgs/headers/docs to SDK 3.10.1
- CMake: fix dynamic reconfigure dependency bug
- Rename topic 'navigate_home' to 'autoflight/navigate_home'
- Add autonomous flight plans
- Contributors: Jacob Perron, Mani Monajjemi

0.5.1 (2016-05-04)

- Add bebop_description as a build dep to bebop_driver (fixes #45)
- Fix inline build of arsdk to use the frozen manifest (fixes #46) - Prior to this release, the build script would always compile the development branch of ARSDK. This fix ensures that instead of *default.xml* manifest file - which represents the dev version of ARSDK package - *release.xml* is used by *repo*. This manifest file includes a certain hash for each ARSDK package that enforces a consistent build for ARSDK.
- Contributors: Mani Monajjemi

0.5.0 (2016-04-01)

- Based on Parrot ARSDK 3.8.3. Tested with Bebop 1.0 (2.0.57) and Bebop 2.0 (3.1.0)
- Update to SDK 3.8.3 - SDK 3.8.3 from https://github.com/Parrot-Developers/arsdk_manifests.git - SDK 3.8.3 git hash: 2930cc7f7a79173d51c1fc167475fa9fa6650def
- Add support for VideoStream v2.0
- Publish TF
- Experimental implementation of odometery (pose, velocity)
- Publish the GPS fix as a ROS standard message (closes #39) - Message type: sensor_msgs/NavSatFix - Topic: fix
- Add joint state publisher for camera's pan/tilt - Add a new param to enable/disable the TF publisher for odom - Add a new param to set the odom frame id
- Include bebop_description and robot_state_publisher in driver's launch files
- Add proper limitations for camera's pan/tilt joints
- Add explicit linkage to libav (fixes #32)
- Fix libav API inconsistency issues #30 #35 #36
- Improve H264 parameter update method - Implement a better way to pass SPS/PPS params to H264 decoder (SDK 3.8.x)
- Add a new CMake option "RUN_HARDWARE_TESTS" to explicitly enable hardware in the loop testing (disabled by default)
- Contributors: Mani Monajjemi, chartoin, Jake Bruce

0.4.1 (2016-02-17)

- Add ROS API for recording on-board picture/video (closes #5)
- Add a new ROS topic for taking on-board snapshot: *snapshot*
- Add a new ROS topic for toggling on-board video recording: *record*
- Update the docs
- Add curl as a rosdep build dep (fixes #33)
- Fix a bug in bebop_driver's nodelet destructor
- Fix a bug in ASyncSub class
- Contributors: Mani Monajjemi

0.4.0 (2016-01-17)

- Update Parrot SDK to 3.7.5 (from 3.6) - Remove upstream XML hash from .msg files to minmize msg type changes from now on - New Topic and Message type for *DefaultCameraOrientation*
- Add cmd_vel timeout for safety - The driver now sends stop command if no new cmd_vel is received within a pre-defined timeout period. This timeout is set to 0.1s by default and can be changed via *cmd_vel_timeout* parameter.
- Fix right-jand rule bug of angular.z @jacobperron (fixes #26)
- Patch ARSDK to fix Sanselan's old URL - This is temporary and must be reverted when this is fixed upstream. Issue reported here: Parrot-Developers/ARSDKBuildUtils#61
- Add bebop ip address as ROS parameter (fixes #19) - (Param name: *bebop_ip*, default value: 192.168.42.1)
- Fix CameraInfo issues (closes #10) - Fix bugs in loading camera calibration data and update the tests - Add a sample calibration file for bebop camera: bebop_camera_calib.yaml - Load camera calibration file by default in both node/nodelet launch files
- Remove redundant bebop.launch file (closes #11)
- Fix coordinate system inconsistencies (fixes #13) - Fix cmd_vel.linear.y sign error - Use attitude values in tests instead of velocities
- Contributors: Anup, Mani Monajjemi, Jacob Perron

0.3.0 (2015-09-17)

- Renamed package to bebop_driver
- Built against ARSDK3_version_3_6
- bebop_autonomy is now a metapackage - bebop_autonomy is the ROS metapackage name - Rename bebop_autonomy package to bebop_driver - Rename bebop_autonomy_msgs to bebop_msgs
- Contributors: Mani Monajjemi

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

Changelog for package bebop_tools

0.6.0 (2016-11-02)

- Added Xbox 360 config file
- Make flattrim joystick command match comment
- Contributors: Jacob Perron, Thomas Bamford

0.5.1 (2016-05-04)

0.5.0 (2016-04-01)

- Improve the organization of launch and param files
- Contributors: Mani Monajjemi

0.4.1 (2016-02-17)

- Update the joystick configuration file for taking snapshots
- Make bebop's namespace in joy_teleop launch file a parameter
- Contributors: Mani Monajjemi

0.4.0 (2016-01-17)

- Fix cmd_vel.linear.y sign issue in joystick config file
- Contributors: Mani Monajjemi

0.3.0 (2015-09-17)

- Renamed package to bebop_tools
- Contributors: Mani Monajjemi

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)

Changelog for package bebop_msgs

0.6.0 (2016-11-02)

- Update autogenerated msgs/headers/docs to SDK 3.10.1
- Contributors: Mani Monajjemi

0.5.1 (2016-05-04)

0.5.0 (2016-04-01)

- Update to SDK 3.8.3 - SDK 3.8.3 from https://github.com/Parrot-Developers/arsdk_manifests.git - SDK 3.8.3 git hash: 2930cc7f7a79173d51c1fc167475fa9fa6650def
- Contributors: Mani Monajjemi

0.4.1 (2016-02-17)

- Fix dynamic_reconfigure's inconsistency - The inconsisteny was between the filename and the target name
- Contributors: Mani Monajjemi

0.4.0 (2016-01-17)

- Update Parrot SDK to 3.7.5 (from 3.6)
- New Topic and Message type for *DefaultCameraOrientation*
- Contributors: Mani Monajjemi

0.3.0 (2015-09-17)

- Renamed to bebop_msgs
- Contributors: Mani Monajjemi

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

Changelog for package bebop_description

0.6.0 (2016-11-02)

0.5.1 (2016-05-04)

0.5.0 (2016-04-01)

- The initial version of bebop_description pacakge - Add a simple kinematic model of Bebop as URDF/Xacro (base_link, camera joints and the optical frame)
- Contributors: Mani Monajjemi

0.4.1 (2016-02-17)

0.4.0 (2016-01-17)

0.3.0 (2015-09-17)

0.2.0 (2015-09-10)

0.1.2 (2015-09-05)

0.1.1 (2015-09-04)

Installation

Compiling From Source

Pre-requirements:

- ROS *Indigo*, *Jade* or *Kinetic* (Only tested on *Ubuntu*)
- 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.

Note: Since version 0.6, Parrot ARSDK, the main underlying dependency of *bebop_autonomy* is not build inline anymore. Instead, the slightly patched and catkinized version of it, called parrot_arsdk, is fetched as a dependency during the `rosdep install` step below. This dramatically decreases the compile time of the package compared to previous versions (e.g. from ~15 minutes to less than a minute on a modern computer). If you are re-compiling from source, you need to clean your workspace first: `$ catkin clean [-y]`.

```
# 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 (including parrot_arsdk)
$ rosdep update
$ rosdep install --from-paths src -i
# Build the workspace
$ catkin build -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

Running the Driver

You can run Bebop's ROS drivereither 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.

Running the driver as a Node

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

```
$ roslaunch bebop_driver bebop_node.launch
```

Listing 2.1: bebop_node.launch

```
<?xml version="1.0"?>
<launch>
    <arg name="namespace" default="bebop" />
    <arg name="ip" default="192.168.42.1" />
    <arg name="drone_type" default="bebop1" /> <!-- available drone types: bebop1, bebop2 -->
    <arg name="config_file" default="$(find bebop_driver)/config/defaults.yaml" />
    <arg name="camera_info_url" default="package://bebop_driver/data/$(arg drone_type)_camera_calib.yaml" />
    <group ns="$(arg namespace)">
        <node pkg="bebop_driver" name="bebop_driver" type="bebop_driver_node" output="screen">
            <param name="camera_info_url" value="$(arg camera_info_url)" />
            <param name="bebop_ip" value="$(arg ip)" />
            <rosparam command="load" file="$(arg config_file)" />
        </node>
        <include file="$(find bebop_description)/launch/description.launch" />
    </group>
</launch>
```

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_driver/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 its own 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_driver -->
    <arg name="namespace" default="bebop" />
    <arg name="ip" default="192.168.42.1" />
    <include file="$(find bebop_driver)/launch/bebop_nodelet.launch">
      <arg name="namespace" value="$(arg namespace)" />
      <arg name="ip" value="$(arg ip)" />
    </include>
    <!-- use the same nodelet manager and namespace, then load image_view nodelet -->
    <group ns="$(arg namespace)">
       <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>
```

Listing 2.3: bebop_driver/launch/bebop_nodelet.launch

```
<?xml version="1.0"?>
<launch>
    <arg name="namespace" default="bebop" />
    <arg name="ip" default="192.168.42.1" />
    <arg name="drone_type" default="bebop1" /> <!-- available drone types: bebop1,␣
↪bebop2 -->
    <arg name="config_file" default="$(find bebop_driver)/config/defaults.yaml" />
    <arg name="camera_info_url" default="package://bebop_driver/data/$(arg drone_
↪type)_camera_calib.yaml" />
    <group ns="$(arg namespace)">
        <!-- nodelet manager -->
        <node pkg="nodelet" type="nodelet" name="bebop_nodelet_manager" args="manager
↪" output="screen"/>
        <!-- bebop_nodelet -->
        <node pkg="nodelet" type="nodelet" name="bebop_nodelet"
          args="load bebop_driver/BebopDriverNodelet bebop_nodelet_manager">
            <param name="camera_info_url" value="$(arg camera_info_url)" />
            <param name="bebop_ip" value="$(arg ip)" />
            <rosparam command="load" file="$(arg config_file)" />
        </node>
        <include file="$(find bebop_description)/launch/description.launch" />
    </group>
</launch>
```

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 is located in `bebop_tools/config` folder. Adapting the file to your own controller is straightforward. To teleop Bebop

while the driver is running execute `roslaunch bebop_tools joy_teleop.launch`.

Takeoff

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

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

Land

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

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

Emergency

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

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

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 counter clockwise
          (-)      Rotate 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 zero 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`.

The `linear.x` and `linear.y` parts of this message set the **pitch** and **roll** angles of the Bebop, respectively, hence control its forward and lateral accelerations. The resulting pitch/roll angles depend on the value of `~PilotingSettingsMaxTiltCurrent` parameter, which is specified in degrees and is dynamically reconfigurable (*Dynamically Reconfigurable Parameters for Bebop*).

The `linear.z` part of this message controls the **vertical velocity** of the Bebop. The resulting velocity in m/s depends on the value of `~SpeedSettingsMaxVerticalSpeedCurrent` parameter, which is specified in meters per second and is also dynamically reconfigurable (*Dynamically Reconfigurable Parameters for Bebop*). Similarly, the `angular.z` component of this message controls the rotational velocity of the Bebop (around the z-axis). The corresponding scaling parameter is `SpeedSettingsMaxRotationSpeedCurrent` (in degrees per sec).

```
roll_degree       = linear.y  * max_tilt_angle
pitch_degree      = linear.x  * max_tilt_angle
ver_vel_m_per_s   = linear.z  * max_vert_speed
rot_vel_deg_per_s = angular.z * max_rot_speed
```

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. The field of view of this virtual camera (based on our measurements) is ~80 (horizontal) and ~50 (vertical) degrees.

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

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

GPS Navigation

Start Flight Plan

An autonomous flight plan consists of a series of GPS waypoints along with Bebop velocities and camera angles encoded in an XML file.

Requirements that must be met before an autonomous flight can start:

- Bebop is calibrated
- Bebop is in outdoor mode
- Bebop has fixed its GPS

To start an autonomous flight plan, publish a message of type std_msgs/String to *autoflight/start* topic. The `data` field should contain the name of the flight plan to execute, which is already stored on-board Bebop.

Note: If an empty string is published, then the default 'flightplan.mavlink' is used.

Warning: If not already flying, Bebop will attempt to take off upon starting a flight plan.

The Flight Plan App allows easy construction of flight plans and saves them on-board Bebop.

An FTP client can also be used to view and copy flight plans on-board Bebop. *FileZilla* is recommended:

```
$ sudo apt-get install filezilla
$ filezilla
```

Then open *Site Manager* (top left), click *New Site*:

- *Host*: 192.168.42.1
- *Protocol*: FTP
- *Encrpytion*: Use plain FTP
- *Logon Type*: Anonymous
- Connect.

Pause Flight Plan

To pause the execution of an autonomous flight plan, publish a message of type std_msgs/Empty to *autoflight/pause* topic. Bebop will then hover and await further commands. To resume a paused flight plan, publish the same message that was used to start the autonomous flight (ie. to the topic *autoflight/start*). Bebop will fly to the lastest waypoint reached before continuing the flight plan.

Note: Any velocity commands sent to Bebop during an autonomous flight plan will pause the plan.

Stop Flight Plan

To stop the execution of an autonomous flight plan, publish a message of type std_msgs/Empty to *autoflight/stop* topic. Bebop will hover and await further commands.

Navigate Home

To ask Bebop to autonomously fly to it's home position, publish a message of type std_msgs/Bool to *autoflight/navigate_home* topic with the `data` field set to `true`. To stop Bebop from navigating home, publish another message with `data` set to `false`.

Warning: The topic has changed from *navigate_home* to *autoflight/navigate_home* after version 0.5.1.

Flat Trim

Error: Test fails, probably not working.

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

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

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
```

Take on-board Snapshot

New in version 0.4.1.

To take a high resolution on-board snapshot, publish a `std_msgs/Empty` message on `snapshot` topic. The resulting snapshot is stored on the internal storage of the Bebop. The quality and type of this image is not configurable using the ROS driver. You can use the official FreeFlight3 app to configure your Bebop prior to flying. To access the on-board media, either connect your Bebop over USB to a computer, or use a FTP client to connect to your Bebop using the following settings:

- Default IP: `192.168.42.1`
- Port: `21`
- Path: `internal_000/Bebop_Drone/media`
- Username: `anonymous`
- Password: *<no password>*

Set camera exposure

It is possible to set camera exposure by publishing `std_msgs/Float32` message on `set_exposure` topic. Note that this functionality is not supported in Bebop1 Fw 3.3.0.

- Exposure value range: `-3.0 .. +3.0`

Toggle on-board Video Recording

New in version 0.4.1.

To start or stop on-board high-resolution video recording, publish a `std_msgs/Bool` message on the `record` topic. The value of `true` starts the recording while the value of `false` stops it. Please refer to the previous section for information on how to access the on-board recorded media.

Reading from Bebop

Camera

The video stream from Bebop's front camera is published on `image_raw` topic as `sensor_msgs/Image` messages. *bebop_driver* 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**. The field of view of this virtual camera (based on our measurements) is ~80 (horizontal) and ~50 (vertical) degrees.

To set the location of camera calibration data, please check this page: *Configuring Bebop and the Driver*. Since v0.4, the package ships with a default camera caliberation file located at `bebop_driver/data/bebop_front_calib.yaml`. Both default node/nodelet launch files, load this file when executing the driver.

Standard ROS messages

Odometery

New in version 0.5.

- ROS Topic: `odom`
- ROS Message Type: `nav_msgs/Odometry`

The driver integerates visual-inertial velocity estimates reported by Bebop's firmware to calculate the odometery. This message contains both the position and velocity of the Bebop in an ENU aligned odometery frame also named as `odom`. This frame name is configurable (see *Driver Parameters*) The cooridnate frame convention complies with ROS REP 103 (*ROS Standard Message Types (i.e Twist, Odometery) - REP 103*). Please not that since odometery is calculated from Bebop States (see *States (aka Navdata)*), the update rate is limited to **5 Hz**.

GPS

New in version 0.5.

- ROS Topic: `fix`
- ROS Message Type: `sensor_msgs/NavSatFix`

Joint States (Pan/Tilt of The Virtual Camera)

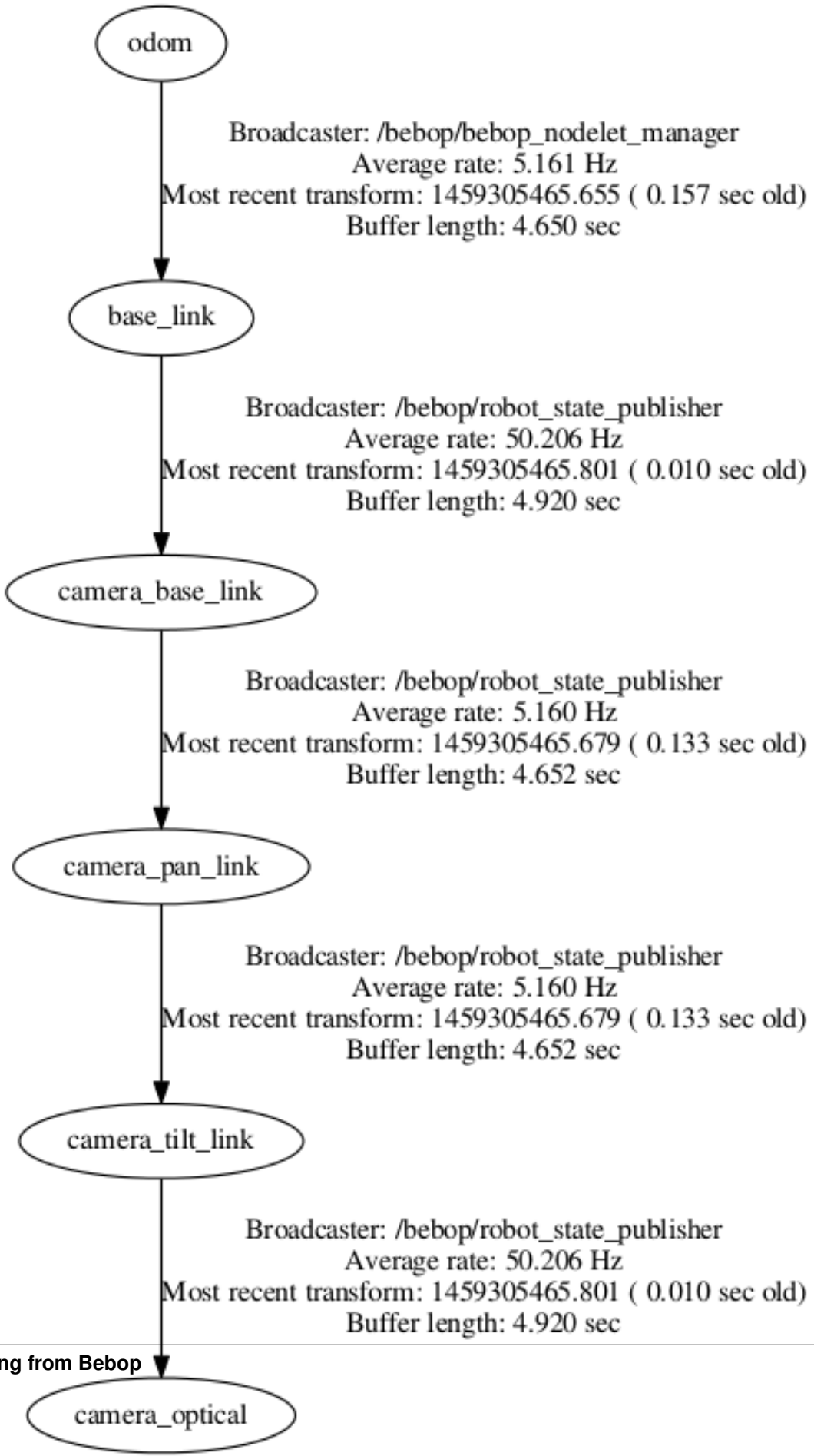
New in version 0.5.

- ROS Topic: `joint_states`
- ROS Message Type: `sensor_msgs/JointState`

TF

New in version 0.5.

The driver updates the following TF tree based on a simple kinematic model of the Bebop (provided by `bebop_description`) pacakge, the current state of the virtual camera joints and the calculated odometery (if `publish_odom_tf` is set, see *Driver Parameters*).


odom
Broadcaster: /bebop/bebop_nodelet_manager
Average rate: 5.161 Hz
Most recent transform: 1459305465.655 (0.157 sec old)
Buffer length: 4.650 sec
base_link
Broadcaster: /bebop/robot_state_publisher
Average rate: 50.206 Hz
Most recent transform: 1459305465.801 (0.010 sec old)
Buffer length: 4.920 sec
camera_base_link
Broadcaster: /bebop/robot_state_publisher
Average rate: 5.160 Hz
Most recent transform: 1459305465.679 (0.133 sec old)
Buffer length: 4.652 sec
camera_pan_link
Broadcaster: /bebop/robot_state_publisher
Average rate: 5.160 Hz
Most recent transform: 1459305465.679 (0.133 sec old)
Buffer length: 4.652 sec
camera_tilt_link
Broadcaster: /bebop/robot_state_publisher
Average rate: 50.206 Hz
Most recent transform: 1459305465.801 (0.010 sec old)
Buffer length: 4.920 sec
camera_optical

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

Configuring Bebop and the Driver

Driver Parameters

Following parameters are set during driver's startup:

~bebop_ip

Sets the IP addres of the Bebop. The default value is `192.168.42.1`.

~reset_settings

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

~sync_time

Setting this parameter to `true` will synchronize drone time with your ROS system time. Default value is `false`.

~camera_info_url

Sets the location of the camera calibration data. Default is empty string. For more information check this documentation.

Note: Since v0.4, the package comes with a default camera calibration file located at `bebop_driver/data/bebop_front_calib.yaml`.

~cmd_vel_timeout

New in version 0.4.

Sets the safety timeout for piloting `cmd_vel` commands in seconds. Deafult is set to **0.1** seconds (100 miliseconds). If no piloting command is received by the driver within this timeout period, the driver issues a stop command which causes the drone to hover.

~odom_frame_id

New in version 0.5.

Sets the `frame_id` of the odometery message (see *Standard ROS messages*) and the odometery frame used in the TF tree (see *TF*). The default value is `odom`.

~publish_odom_tf

New in version 0.5.

Enables the publishing of `odom` to `base_link` TF transform (see *TF*). The default value is `true`.

~camera_frame_id

Deprecated since version 0.5.

Sets the `frame_id` of camera and image messages. The default value is `camera_optical`.

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

Coordinate System Conventions

ROS Standard Message Types (i.e Twist, Odometery) - REP 103

+x	forward
+y	left
+z	up
+yaw	CCW

- More information: http://www.ros.org/reps/rep-0103.html

Bebop Velocities

+x	East
+y	South
+z	Down

Bebop Attitude

+x	forward
+y	right
+z	down
+yaw	CW

SDK's setPilotingPCMD

+roll	right
+pitch	forward
+gaz	up
+yaw	CW

Contribute

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.

List of Developers

- Mani Monajjemi

List of Contributers

- Anup Parikh
  - #19 Add bebop ip address as ROS parameter
- Jacob Perron
  - #26 Bebop now follows right-hand rule
  - #57 Add autonomous flight plan capabilities to driver
- Jake Bruce
- Charuvahan Adhivarahan
  - #30 Fixed breaking build due to legac macro usage

- Thomas Bamford
  - Added Xbox 360 config file
- Sepehr Mohaimenianpour

Acknowledgments

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

Frequently Asked Questions

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.

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)*)

Under The Hood

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

Automatic Code Generation

TBA

Threading Model

TBA

Publishing the States

TBA

Configuring the Drone

TBA

Tests

Enabling Bebop In The Loop Tests

```
$ cd /path/to/bebop_ws
$ catkin clean --cmake-cache
$ catkin build bebop_driver --cmake-args -DRUN_HARDWARE_TESTS=ON
```

Running Bebop In The Loop Tests

Warning: Bebop in the loop tests perform live unit tests with a real robot. Please proceed with caution and execute the tests in an area with at least 5 meters of clearance radius (empty space) around the Bebop.

```
$ cd /path/to/bebop_ws/build/bebop_driver
$ make tests
$ rostest --text bebop_driver bebop_itl_test.test
```

Upgrading the Bebop SDK

Warning: Since version 0.6, Parrot ARSDK, the main underlying dependency of *bebop_autonomy* is not build inline anymore. Instead, the slightly patched and catkinized version of it, called parrot_arsdk, is fetched as a dependency. **The following documentation needs to be updated**.

1. Update the `GIT_TAG` of `ARDroneSDK3` in `bebop_driver/CMakeLists.txt::add_custom_target::./repo init ... -b GIT_TAG` to your desired commit hash, branch or tag (release). The official upstream repository is hosted here.
2. Checkout (or browse) the upstream repository at the same hash used in step (1) and open `release.xml` file. From this file, extract the commit hash of `arsdk-xml` from the `revision` property of this XML tag: `<project name="arsdk-xml" ... revision="" />`.
3. Open `bebop_driver/scripts/meta/generate.py` and update `LIBARCOMMANDS_GIT_HASH` variable to the hash obtained in step (2).
4. Change the working diretory to `bebop_driver/scripts/meta`, then execute `generate.py` from the command line. This will regenerate all automatically generated message definitions, header files and documentations.
5. Copy the generated files to their target locations by calling `./install.sh`.
6. In `bebop_driver/include/bebop_driver/autogenerated/ardrone3_setting_callbacks.h` change the following varialbles:

- `ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_MAXDISTANCECHANGED_VALUE` to `ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_MAXDISTANCECHANGED_CURRENT`

- ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_BANKEDTURNCHANGED_VALUE to ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_BANKEDTURNCHANGED_STATE
- ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_CIRCLINGRADIUSCHANGED_VALUE to ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_CIRCLINGRADIUSCHANGED_CURR
- ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_CIRCLINGALTITUDECHANGED_VAL to ARCONTROLLER_DICTIONARY_KEY_ARDRONE3_PILOTINGSETTINGSSTATE_CIRCLINGALTITUDECHANGED_CU

These changes are required becasue the upstream XML file is inconsistent for a couple of variable names.

7. Remove `build` and `devel` space of your `catkin` workspace, then re-build it.

License

Parrot ARDrone3 SDK

bebop_autonomy (driver and tools)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search