docs Documentation

Release indigo-devel

Ilja Schirschow

November 23, 2016

Contents

This ROS package for the indigo distribution contributes to the design and implementation of a replaceable camera system for the Parrot AR-Drone 2.0. A low-cost 3D-printable camera mount system was designed that can be used, despite the Parrot AR.Drone 2.0, for other robots. The design is compact, light and modular, allowing exchanging or adding parts as desired. Based on experiments with additional weight, the drones payload is perceived as critical to carry the system but feasible.

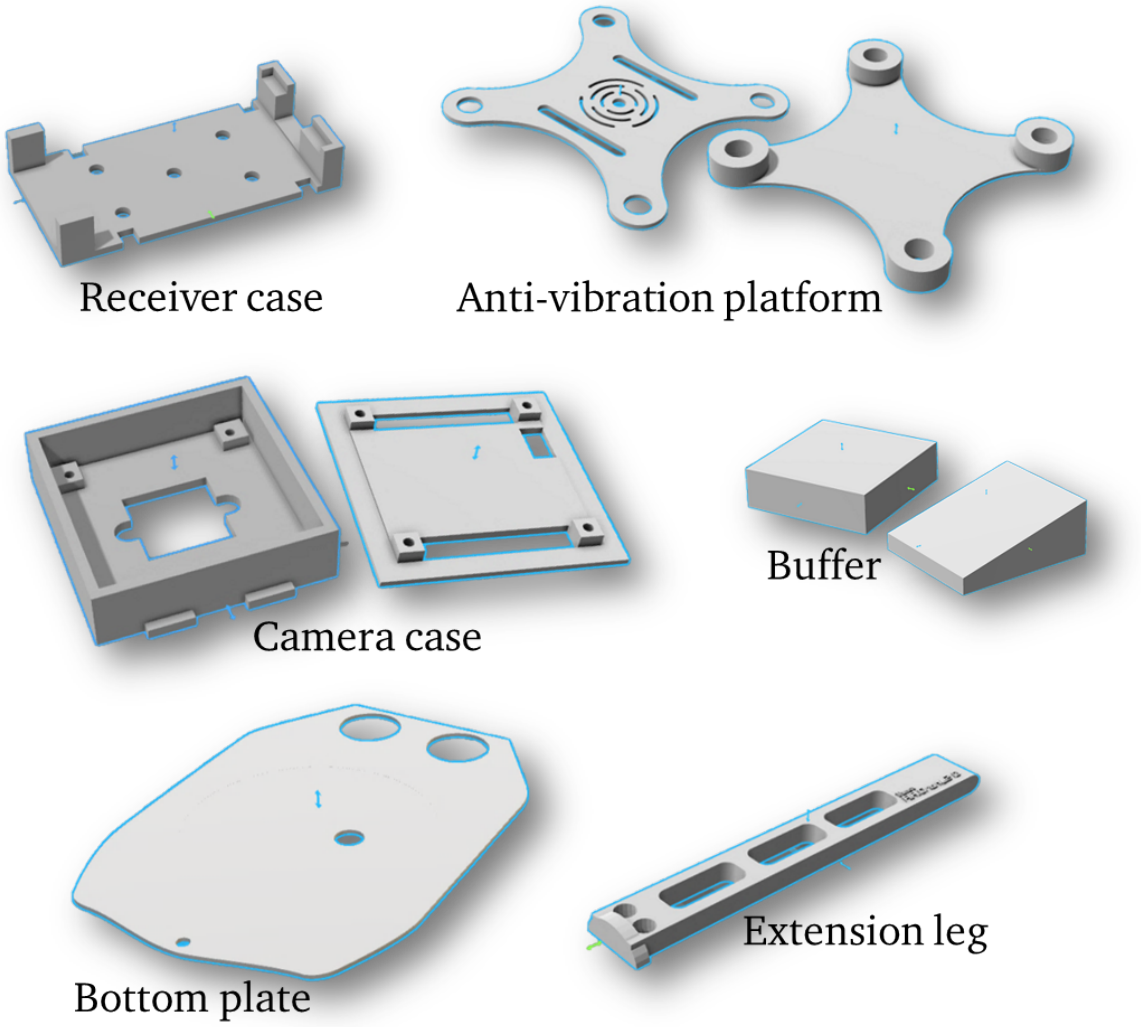
CHAPTER 1

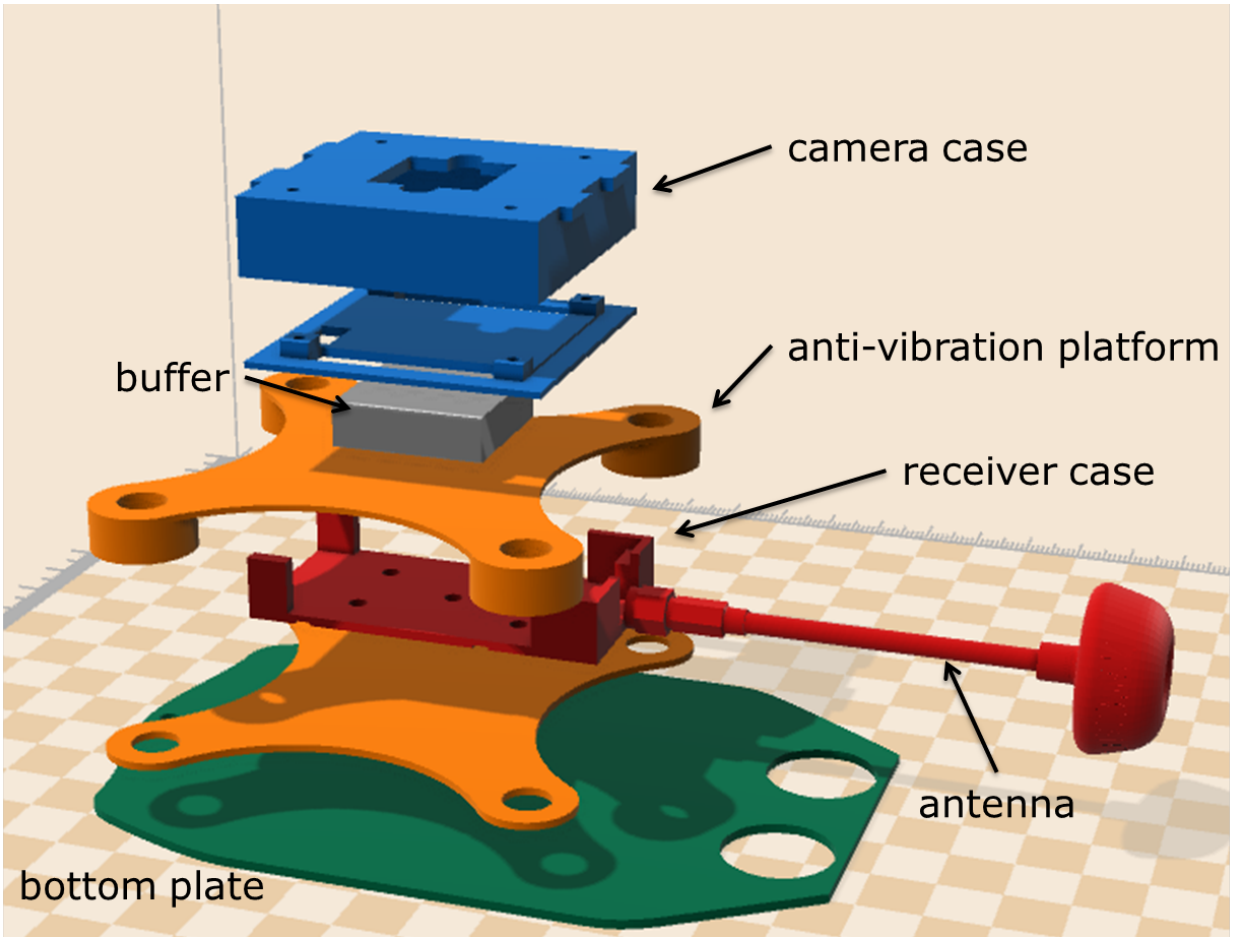
Table of Contents

1.1 External Camera System: Design

1.1.1 Mounting System

The .stl files can be found at: http://www.thingiverse.com/thing:1454407


Receiver case
Anti-vibration platform
Camera case
Buffer
Bottom plate
Extension leg

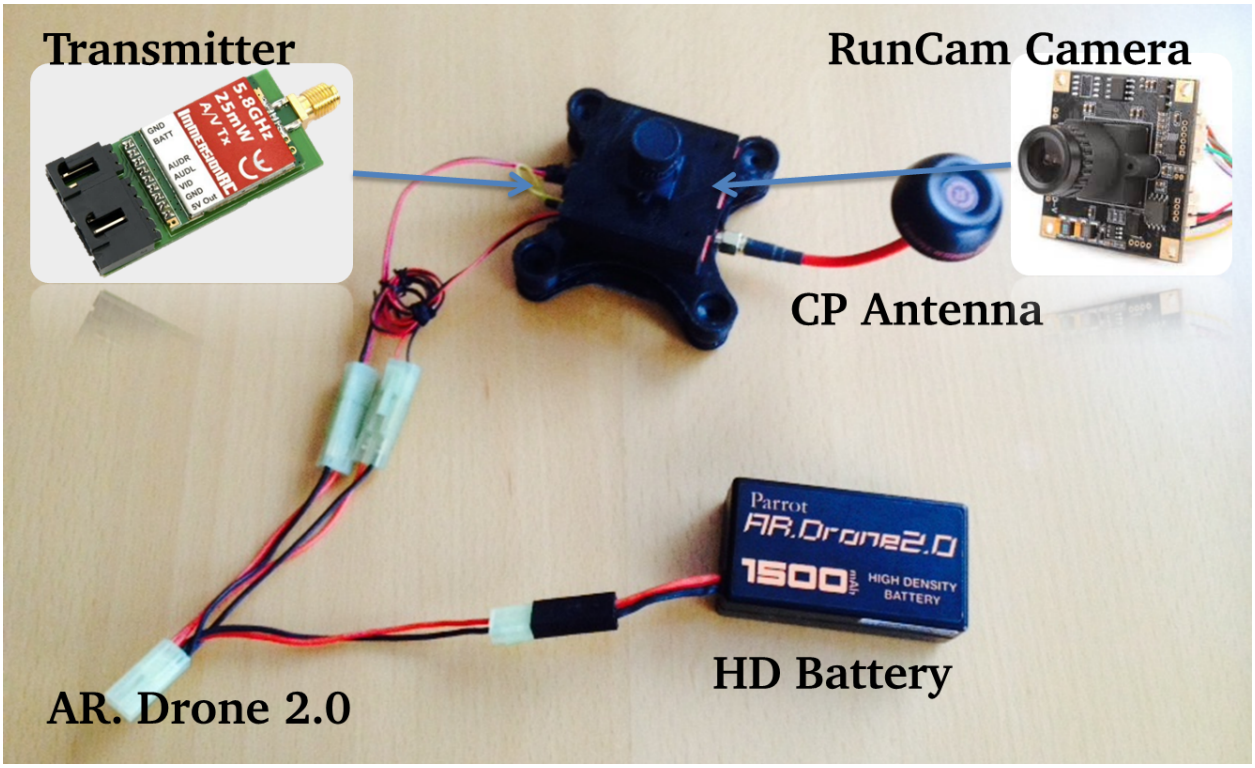


1.1.2 Physical Installation

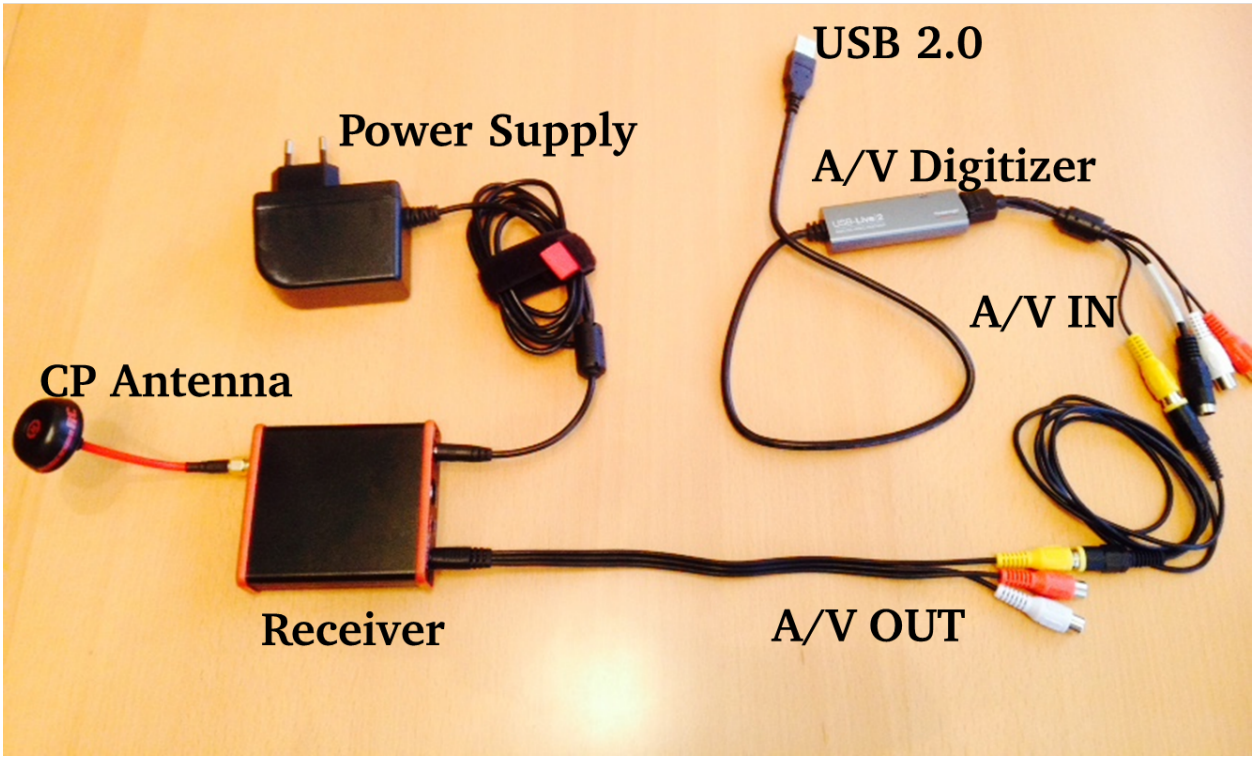
Transmitter-Receiver Components

Name	Specification	Price in €
Camera	RunCam 800TVL Sky 2	33,70
Transmitter	ImmersionRC 25mW 5.8GHz	59,90
Receiver	ImmersionRC UNO5800 v4	79,90
A/V Digitizer	Hauppauge USB-Live2	42,99
Antenna	ImmersionRC Fatshark CP	54,90

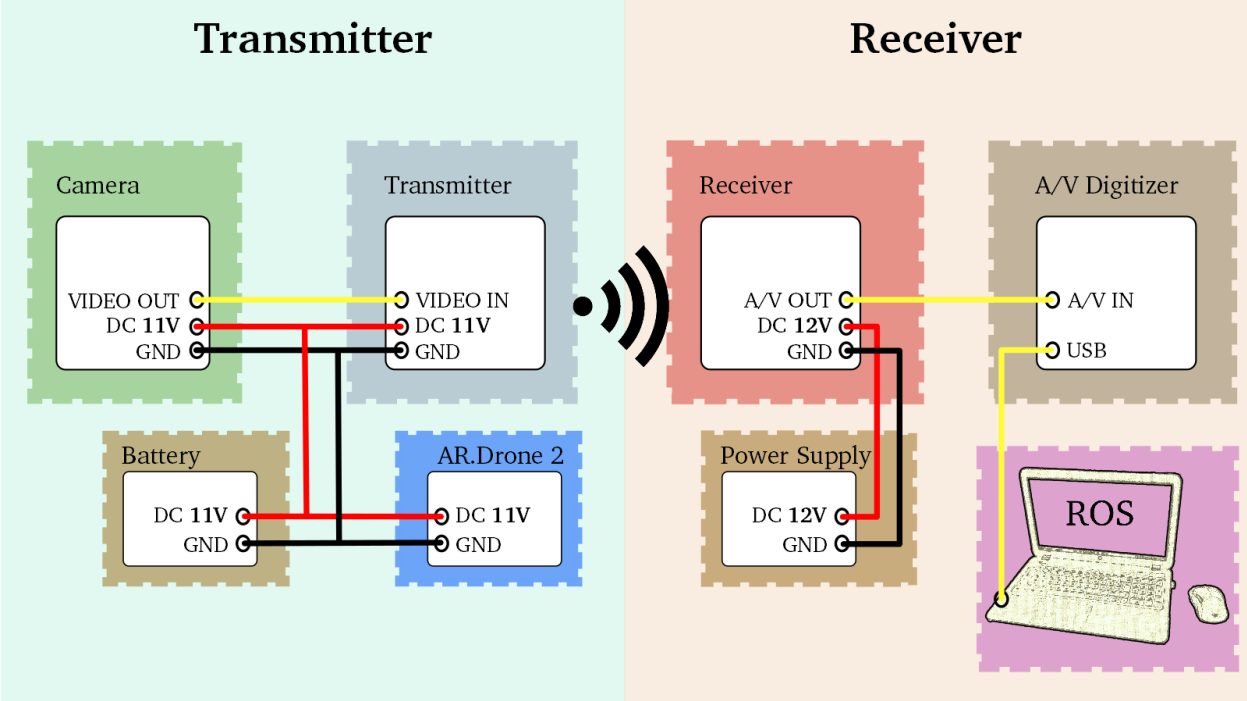
Transmitter



Receiver



1.1.3 Electrical Installation



1.2 Installation

1.2.1 ROS Dependancies

Install the OpenCV3 package for ROS

```
sudo apt-get install ros-indigo-opencv3
```

Install the usb_camera/uvc_camera drivers

```
sudo apt-get install ros-indigo-usb-cam
sudo apt-get install ros-indigo-uvc-camera
```

1.2.2 Package Installation

Clone and build the package from source into your catkin workspace

```
cd ~/catkin_ws/src
git clone https://ille90@bitbucket.org/ille90/external_camera_tf.git
cd ~/catkin_ws/src/external_camera_tf
catkin build --this
```

1.3 Usage

1.3.1 Launch files

The *camera_external.launch* file runs the camera driver. The drone can be controlled with a Xbox 360 controller using *joystick_controller.launch* file.

- camera_external.launch
- joystick_controller.launch

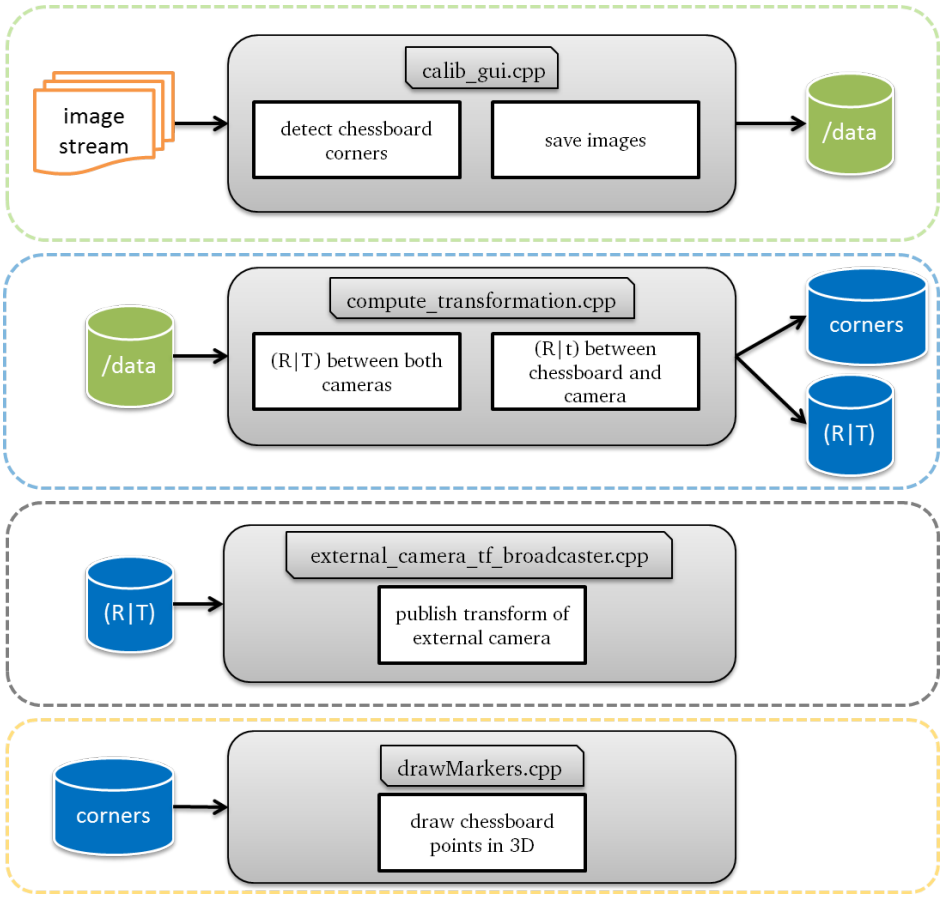
Preview

1.3.2 Nodes

The 6 nodes need some preparation to work correctly. The settings concern in particular the image file names and their input/output paths.

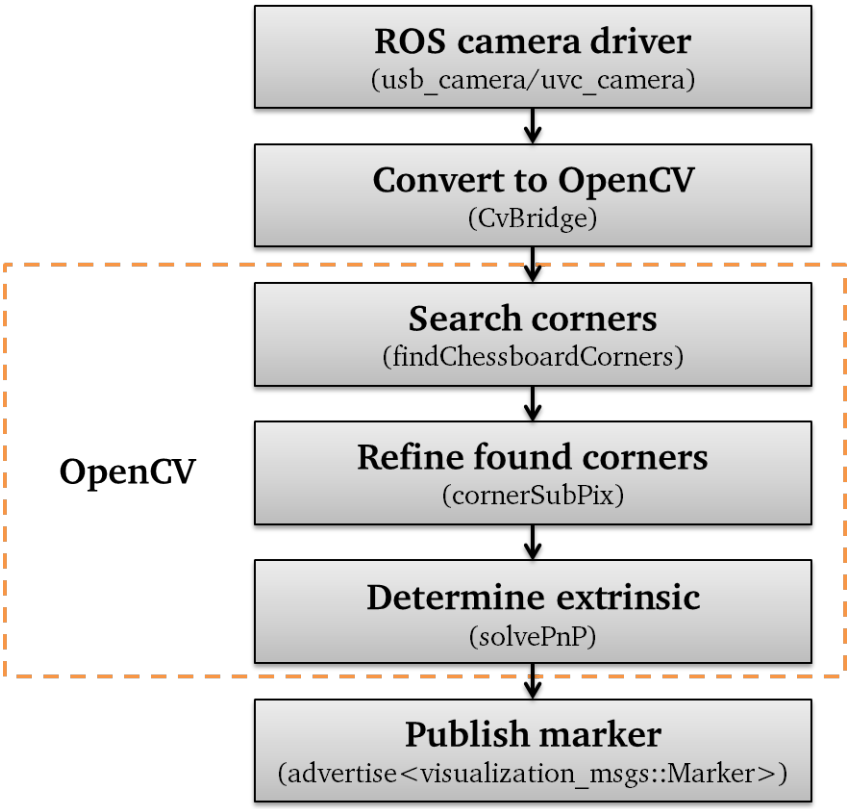
- calib_gui
- check_for_chessboard
- compute_single_camera_pose
- compute_transformation
- marker_broadcaster
- external_camera_tf_broadcaster

The following figure shows the package on high level of abstraction



1.3.3 check_for_chessboard

This node subscribes to ONE image_topic and checks with OpenCV for a chessboard/circleboard. The detected corners are published as marker points and can be visualized with RVIZ in 3D. The process is illustrated in the following figure.



Note: ROS: image publisher Opencv

ROS: marker broadcaster

Preview

How to use

Edit the settings in the check_for_chessboard.cpp file:

1. Edit the paths to the camera_info files, containing the intrinsic parameter of the cameras

```
// 1. Paths to the camera_info files, containing the intrinsic parameter of the cameras
static const string ARDRONE_FRONT_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/cam
static const string ARDRONE_BOTTOM_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/ca
static const string ARDRONE_EXTERNAL_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/
```

2. Choose a camera: 0=external,1=front,2=bottom

```
int caseSwitch = 0;
```

3. Choose a board: chessboard=0 or circleboard=1

```
int caseSwitchBoard = 0;
```

4. Edit the image topic to subscribe to if neccesary

```
    // 4. Image topic to subscribe to
    static const string CAMERA1_TOPIC = "/ardrone/image_raw";
    static const string CAMERA2_TOPIC = "/camera_external/image_raw";
```

5. Chessboard characteristics; squareSize in meter

```
    // 5. Chessboard characteristics; squareSize in meter
    double squareSize = 0.0359375;
    cv::Size chessboardSize = cv::Size(8, 6);
```

Note: chessboadSize - the parameters are NOT the quantity of the squares rather the quantity of the inner corner points

Then run the camera driver to publish an image_raw topic

```
roslaunch external_camera_tf camera_external.launch
```

Finally, run the node:

```
rosrun external_camera_tf check_for_chessboard
```

1.3.4 calib_gui

This node subscribes to image_raw topics from two cameras and detects a chessboard. The images can be saved only if the chessboard is detected by both cameras.

Preview

How to use

1. Edit topics this node subscribes to

```
// 1. Topics this node subscribes to
 static const string CAMERA1_TOPIC = "/ardrone/image_raw";
 static const string CAMERA2_TOPIC = "/camera_external/image_raw";
```

2. Edit Size of the checkerboard

```
// 2. Size of the checkerboard
 cv::Size checkerboardSize = cv::Size(8, 6);
```

3. Edit the two folder paths where the images will be saved on your computer

```
void buttonCallback(int state, void* userdata)
{
// save current image to folder
char file_name1[100];
char file_name2[100];
sprintf(file_name1, "/home/ille/catkin_ws/src/external_camera_tf/data/images/bottom/bottom_cam%d.jpg"
sprintf(file_name2, "/home/ille/catkin_ws/src/external_camera_tf/data/images/external/external_cam%d.
      fileIndex2++);

      // saves the images if data is okay
      if (src2.data && src1.data )
      {
```

```
        imwrite(file_name1, src1);
        imwrite(file_name2, src2);
        }
}
```

Warning: Care to leave the %d in the path name

Run this node when both cameras are publishing their topic. For the standard case this will be:

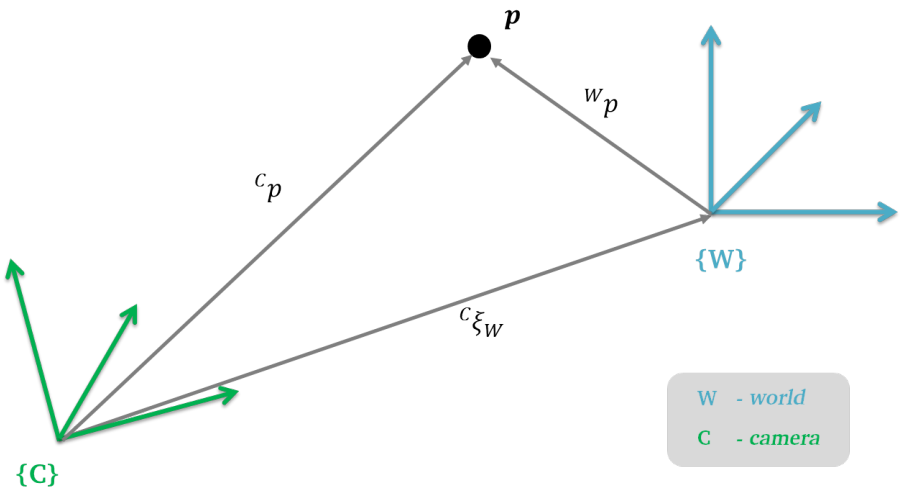
```
roslaunch external_camera_tf camera_external.launch
rosrun ardrone_autonomy ardrone_driver
rosrun external_camera_tf calib_gui
```

Ctrl+P or right-Click on one of the images and hit on "Display properties window" to show the save button. Hit save when both images show a detected chessboard with the colors in the same direction.

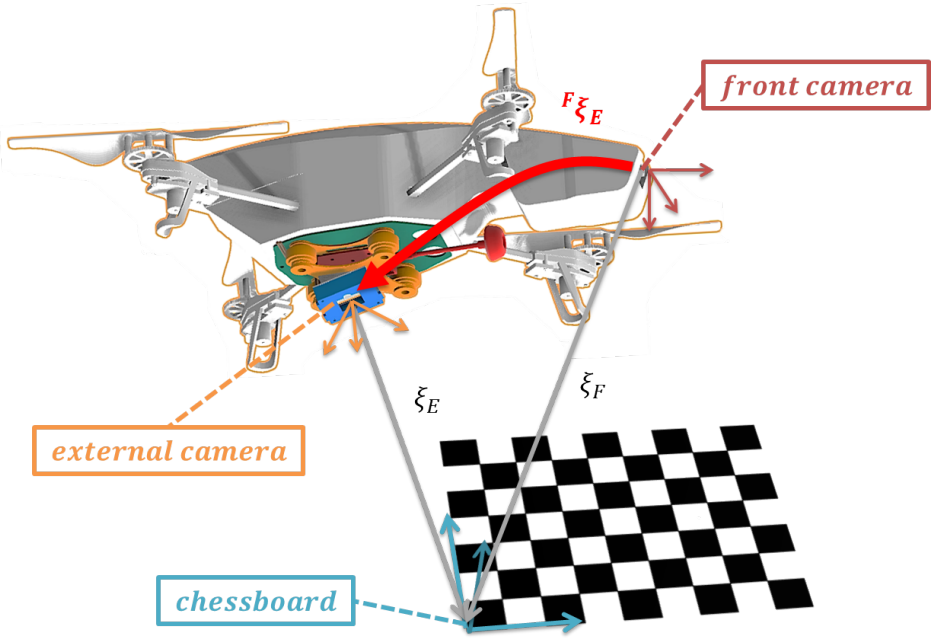
1.3.5 compute_transformation

This node is executed once and needs some preparation to work correctly. First task of the function is to import the saved images of the front/bottom and external camera from the /data folder and calculate the rotation and translation between the chessboard and the camera for every single image. With known poses for each camera in respect to the chessboard, this node computes the transform between the cameras as depicted in the following figures:

Point p in camera coordinate frame and world coordinate frame



Pose(red) between two cameras



The uknown pose between the cameras is given by

$$\xi_F = {}^F\xi_E \oplus \xi_E$$

$${}^F\xi_E = \xi_F \ominus \xi_E$$

With the inverse

$$T^{-1} = \begin{pmatrix} R & t \\ 0_{1\times3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} R^T & -R^T t \\ 0_{1\times3} & 1 \end{pmatrix}$$

and

$$T_1 T_2 = \begin{pmatrix} R_1 & t_1 \\ 0_{1\times3} & 1 \end{pmatrix} \begin{pmatrix} R_2 & t_2 \\ 0_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} R_1 R_2 & t_1 + R_1 t_2 \\ 0_{1\times3} & 1 \end{pmatrix}$$

we get the final equation to determine the pose

$$T_1 T_2^{-1} = \begin{pmatrix} R_1 & t_1 \\ 0_{1\times3} & 1 \end{pmatrix} \begin{pmatrix} {R_2}^T & -R_2^T t_2 \\ 0_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} R_1 {R_2}^T & t_1 - R_1 R_2^T t_2 \\ 0_{1\times3} & 1 \end{pmatrix}$$

Preview

How to use

1. Edit the paths to the camera_info files, containing the intrinsic parameter of the cameras

```
// 1. Paths to the camera_info files
 static const string ARDRONE_FRONT_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/camera_
 static const string ARDRONE_BOTTOM_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/camera
 static const string ARDRONE_EXTERNAL_CAMERA = "/home/ille/catkin_ws/src/external_camera_tf/data/came
```

2. Edit Size and squareSize of the checkerboard

```
// 2. Size and squareSize(in meter) of the checkerboard
 double squareSize = 0.0359375;
 cv::Size chessboardSize = cv::Size(8, 6);
```

3. Chose which cameras are used

```
// 3. Switch-case for camera intrinsics: external+front=0; external+bottom=1
 int caseSwitch = 0;
```

4. Edit imported image files

```
/* 4. Import Settings:
*      IMAGE_SOURCE1 = the path to the images
*      IMAGE_BASE_NAME1 = base name of the images, without the number
*      IMAGE_TYPE1 = type of the image
*/

static const string IMAGE_SOURCE1 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/front/";
static const string IMAGE_BASE_NAME1 = "front_cam";
static const string IMAGE_TYPE1 = ".jpg";

static const string IMAGE_SOURCE2 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external
static const string IMAGE_BASE_NAME2 = "external_cam";
static const string IMAGE_TYPE2 = ".jpg";
```

Warning: Be careful with the Import Settings, most errors happen because of missing / or wrong path

5. Chose where the Output parameters are saved

```
/* 5. Output:
 *      PARAMETERS1 = R|t , rotation and translation for camera1
 *      PARAMETERS2 = R|t , rotation and translation for camera2
 *      POSE = R|T, rotation and translation between camera1 and camera2. camera1 is the "origin"
 *      WORLD_POINTS1 = detected corner points in 3D (X,Y,Z) from the camera1 point of view
 *      WORLD_POINTS2 = detected corner points in 3D (X,Y,Z) from the camera2 point of view
 */

static const string SINGLE_POSE1 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/front/sin
```

```
static const string WORLD_POINTS1 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/front/po

static const string SINGLE_POSE2 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external/
static const string WORLD_POINTS2 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external

static const string POSE = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external/pose.yam
```

If all 5 settings are correct, just run this node with

```
rosrun external_camera_tf compute_transformation
```

1.3.6 external_camera_tf_broadcaster

A simple tf_broadcaster for the external camera.

Preview

See first part of the check_for_chessboard preview.

How to use

Copy & paste R and T from pose.yaml that was determined by compute_transformation.cpp

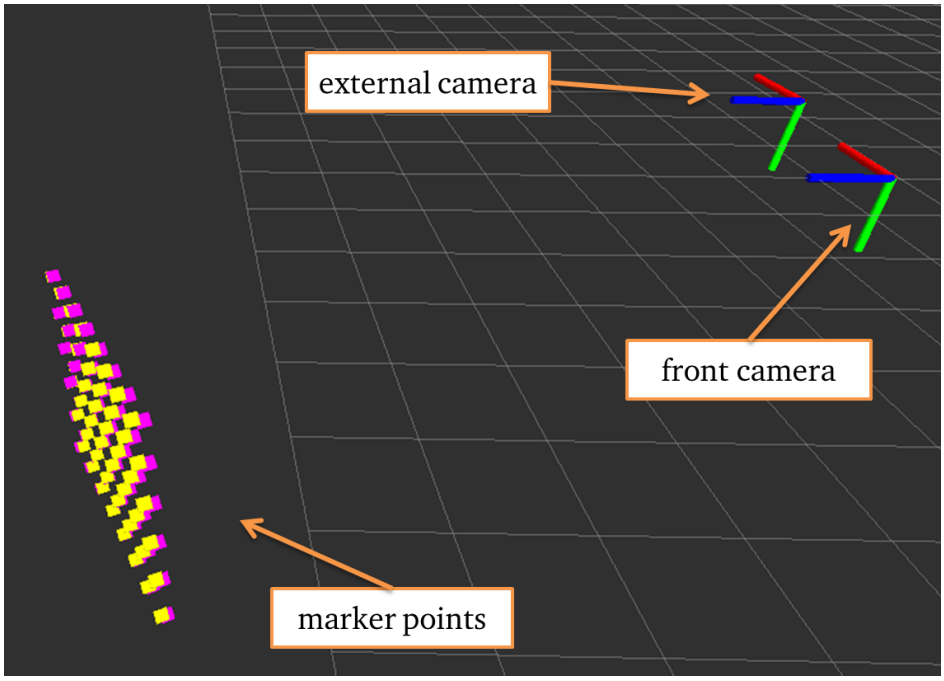
```
// copy & paste R and T from pose.yaml
tf::Matrix3x3 R = tf::Matrix3x3( 9.978979937188068e-01, 2.6072910076787169e-03,
                                 -6.4751809688806963e-02, -6.1357256215213542e-03,
                                 9.9850294377488946e-01, -5.4352774942752063e-02,
                                 6.4513159087674535e-02, 5.4635824387313209e-02,
                                 9.9642008159111639e-01 );

tf::Vector3 T = tf::Vector3(1.7364865838495314e-01, -2.3857123863684243e-02,
                            3.4440008732199434e-03);
```

Note: Based on: ROS tf broadcaster

1.3.7 marker_broadcaster

Imports the found corner points in the camera point of view for both cameras and broadcastes the points using Marker points from RVIZ.



Note: ROS: marker broadcaster

Preview

How to use

1. Select an image pair

```
// 1. Chessboard points of the image pair to broadcast
 static const string IMAGE = "image3";
```

2. Edit CAMERA1 parameters from */ardrone_base_frontcam* to */ardrone_base_bottomcam* when using the bottom camera

```
// 2. Camera IDs, from ardrone_driver and external_camera_tf_broadcaster
 static const string CAMERA1 = "/ardrone_base_frontcam";
 static const string CAMERA2 = "/ardrone_base_externalcam";
```

3. Edit the path to the file with the objectpoints(*WORLD_POINTS1* and *WORLD_POINTS2*)

```
// 3. Path to the file with the objectpoints
 static const string OBJ_POINTS1 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/front/po
 static const string OBJ_POINTS2 = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external/
```

1.3.8 compute_single_camera_tf

This node determines the rotation (as a 3x3 matrix and a 3x1 Rodrigues vector) and translation vector to a chessboard from a the camera point of view and saves the result in a .yaml file.

Preview

How to use

1. Edit the path to the camera_info file

```
// 1. Path to the camera_info
 static const string CAMERA_INFO = "/home/ille/catkin_ws/src/external_camera_tf/data/camera_info/ardr
```

2. Edit the path where the image is imported from

```
// 2. Path where ONE image is imported from
 static const string IMAGE_SOURCE = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external
 static const string IMAGE_BASE_NAME = "external_cam0";
 static const string IMAGE_TYPE = ".jpg";
```

3. Path where the results are saved

```
// 3. Save directory of the desired parameters (R|t) <=> (rotation | translation)
 static const string IMAGE_SINK = "/home/ille/catkin_ws/src/external_camera_tf/data/images/external/s
```

1.4 License

The code is subject to BSD license