Note: This tutorial assumes that you have completed the previous tutorials: ROS Tutorials (/ROS/Tutorials).

It is appreciated that problems/questions regarding this tutorial are asked on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, versions of your OS & ROS, and also add appropriate tags.

1. How to Calibrate a Monocular Camera

Description: This tutorial cover using the camera_calibration (/camera_calibration)'s `cameracalibrator.py` node to calibrate a monocular camera with a raw image over ROS.

Keywords: monocular, camera, calibrate

Tutorial Level: BEGINNER

Conteúdo

1.1 Before Starting

Make sure that you have the following:

- a large checkerboard (/camera_calibration/Tutorials/MonocularCalibration?action=AttachFile&do=view&target=check-108.pdf) with known dimensions. This tutorial uses a 8x6 checkerboard with 108mm squares. Calibration uses the interior vertex points of the checkerboard, so an "8x6" board is nine squares wide and seven high, like the example below.
- a well lit 5m x 5m area clear of obstructions and check board patterns
- a monocular camera publishing images over ROS

1.2 Compiling

Start by getting the dependencies and compiling the driver.

```
$ rosdep install camera_calibration
$ rosmake camera_calibration
```

1.3 Camera Publishing

Make sure that your monocular camera is publishing images over ROS. Let's list the topics to check that the images are published:

```
$ rostopic list
```

This will show you all the topics published, check to see that there is an `image_raw` topic. The default topics provided by most ROS camera drivers are:

```
/camera/camera_info
/camera/image_raw
```

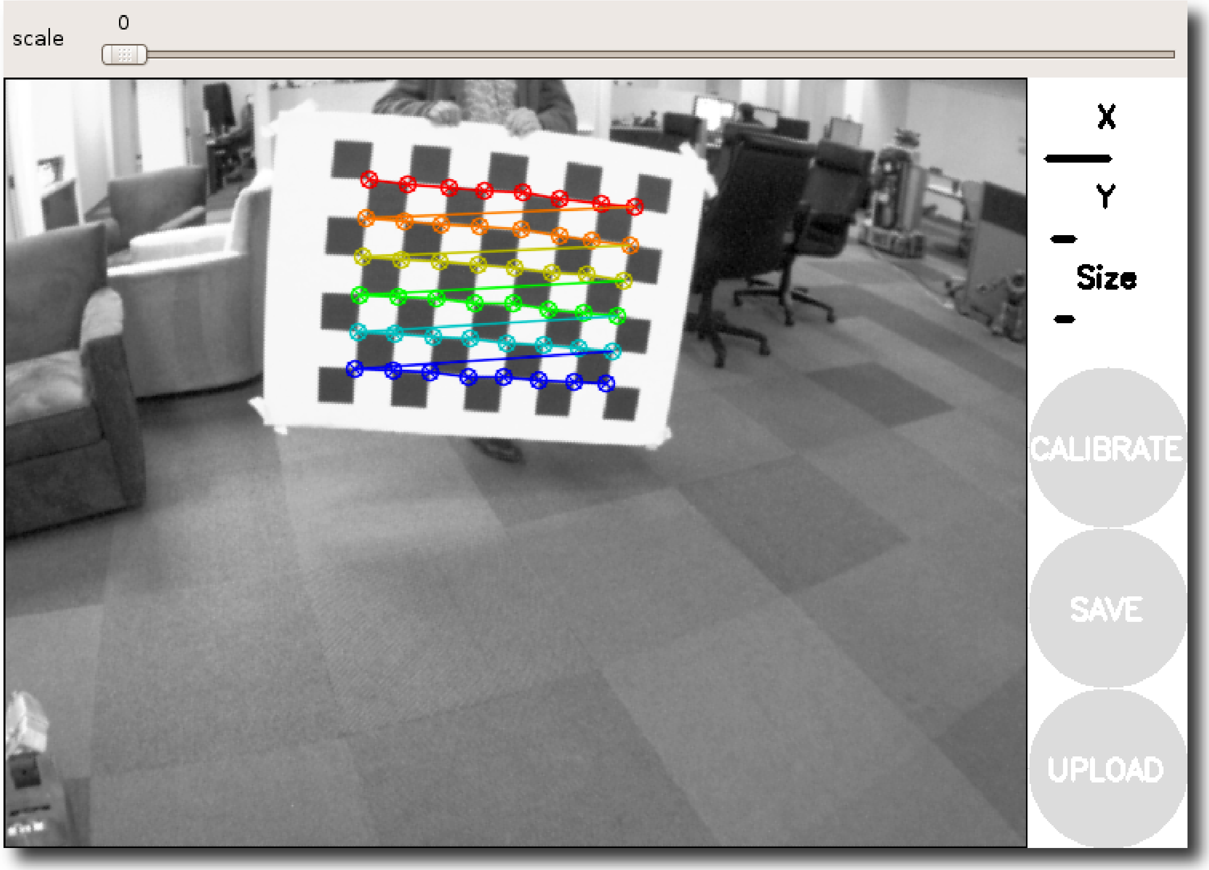
If you have multiple cameras or are running the driver in its own namespace, your topic names may differ.

1.4 Running the Calibration Node

To start the calibration you will need to load the image topics that will be calibrated:

```
$ rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.108 image:=/camera/image_raw camera:=/camera
```

This will open up the calibration window which will highlight the checkerboard:



If it does not open up the window try the following parameter:

```
--no-service-check
```

If you can't see any colored dots make sure you count the interior vertex points, not the squares!

1.4.1 Dual Checkerboards

New in D

Starting in Diamondback, you will be able to use multiple size checkerboards to calibrate a camera.

To use multiple checkerboards, give multiple `--size` and `--square` options for additional boards. Make sure the boards have different dimensions, so the calibration system can tell them apart.

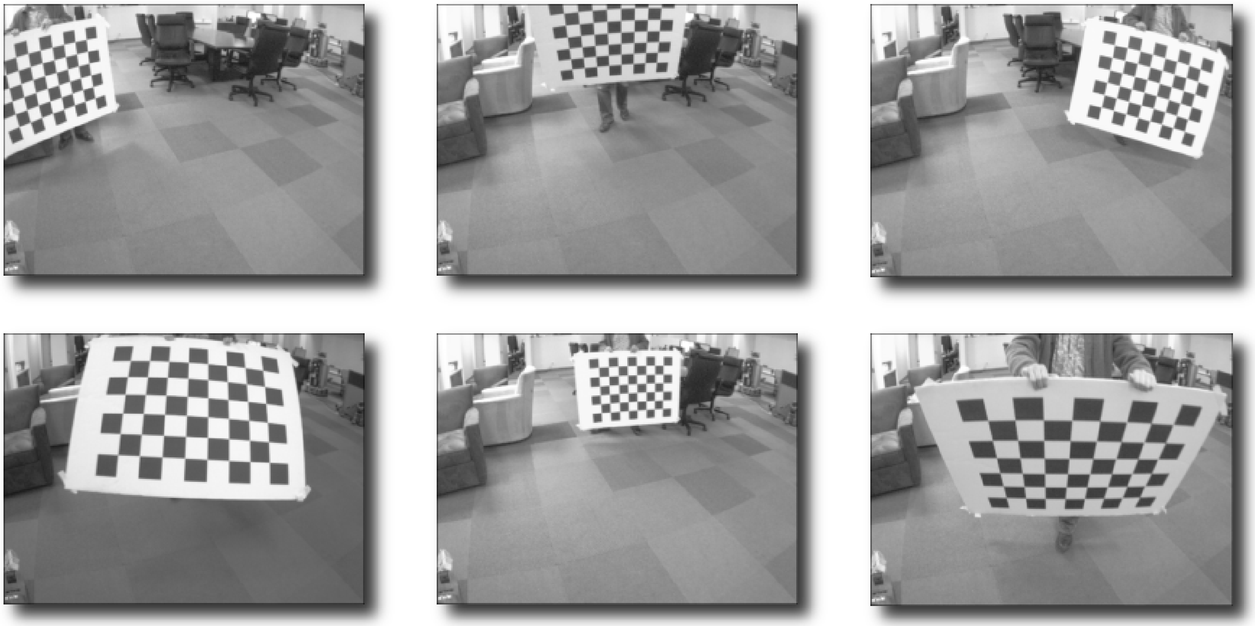
1.5 Moving the Checkerboard

In order to get a good calibration you will need to move the checkerboard around in the camera frame such that:

- checkerboard on the camera's left, right, top and bottom of field of view
  - X bar - left/right in field of view

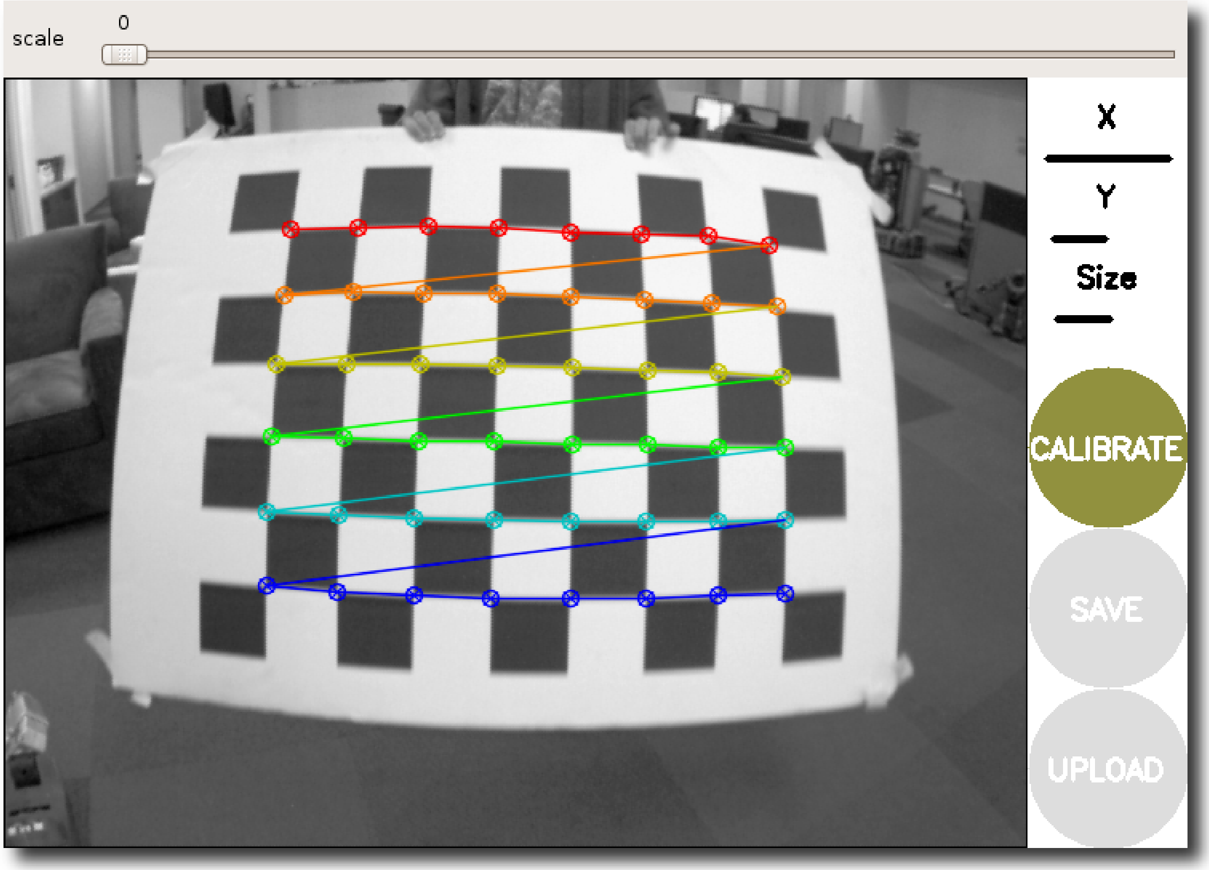
    - Y bar - top/bottom in field of view
    - Size bar - toward/away and tilt from the camera
- checkerboard filling the whole field of view
- checkerboard tilted to the left, right, top and bottom (Skew)

At each step, hold the checkerboard still until the image is highlighted in the calibration window.



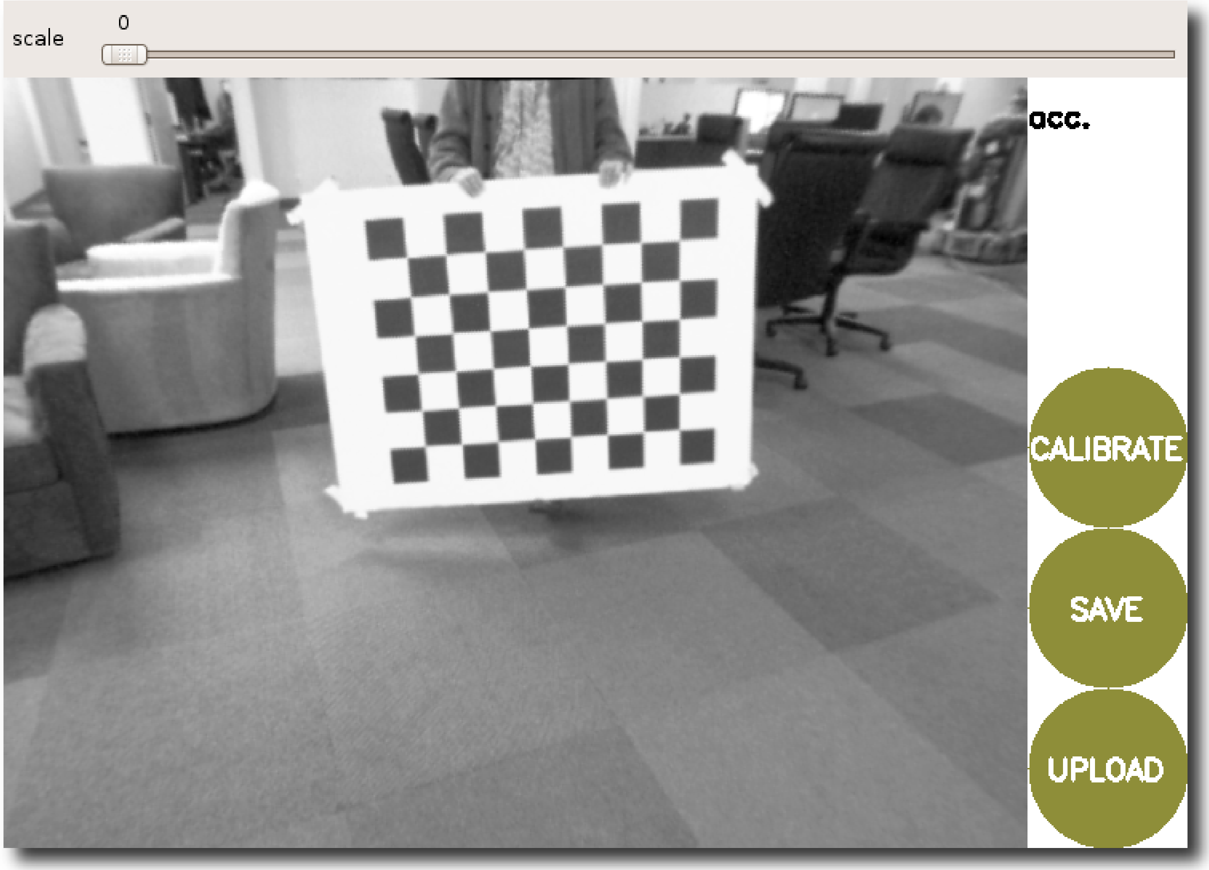
As you move the checkerboard around you will see three bars on the calibration sidebar increase in length. When the **CALIBRATE** button lights, you have enough data for calibration and can click **CALIBRATE** to see the results.

Calibration can take about a minute. The windows might be greyed out but just wait, it is working.



1.6 Calibration Results

After the calibration is complete you will see the calibration results in the terminal and the calibrated image in the calibration window:



A successful calibration will result in real-world straight edges appearing straight in the corrected image.

A failed calibration usually results in blank or unrecognizable images, or images that do not preserve straight edges.

After a successful calibration, you can use the slider at the top of the calibration window to change the size of the rectified image. A scale of 0.0 means that the image is sized so that all pixels in the rectified image are valid. The rectified image has no border, but some pixels from the original image are discarded. A scale of 1.0 means that all pixels in the original image are visible, but the rectified image has black borders where there are no input pixels in the original image.

```
D =  [-0.33758562758914146, 0.11161239414304096, -0.00021819272592442094,
-3.029195446330518e-05]
K =  [430.21554970319971, 0.0, 306.6913434743704, 0.0, 430.53169252696676,
227.22480030078816, 0.0, 0.0, 1.0]
R =  [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P =  [1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
# oST version 5.0 parameters


[image]

width
640

height
480

[narrow_stereo/left]

camera matrix
430.215550 0.000000 306.691343
0.000000 430.531693 227.224800
0.000000 0.000000 1.000000

distortion
-0.337586 0.111612 -0.000218 -0.000030 0.0000

rectification
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000

projection
1.000000 0.000000 0.000000 0.000000
0.000000 1.000000 0.000000 0.000000
0.000000 0.000000 1.000000 0.000000
```

If you are satisfied with the calibration, click **COMMIT** to send the calibration parameters to the camera for permanent storage. The GUI exits and you should see "writing calibration data to ..." in the console.

1.7 Creating a yml file

The Camera Calibration Parser (/camera_calibration_parsers) helps you to create a yml file, which you can load with nearly all ros camera driver using the *camera_info_url* parameter.

1.8 Rectifying an image

Simply loading a calibration file does not rectify the image. For rectification, use the image_proc package (/image_proc).

Wiki: camera_calibration/Tutorials/MonocularCalibration (última edição 2015-03-23 21:04:57 efectuada por bradknox (/bradknox))