Kalibr camera calibration notes

Background

Camera calibration tells us about the intrinsic and extrinsic parameters of a camera.

The intrinsic parameters include the focal length and the principle point. The extrinsic parameters include the relationship between the camera and some frame of reference which for VIO is normally an IMU.

If we are using a fisheye camera we often have to include an additional distortion model. A distortion mode helps to correct for optical effects such as barrel or pincushion distortion which become relevant when using certain types of lenses.

To calibrate cameras we will use the open source software Kalibr. Kalibr provides a set of scripts which can be used to calibrate single and multi-camera setups. It can also calibrate camera to IMU. For more resources please visit there wiki as they have several very useful pages around how to run each script as well as a video detailing the calibration process.

The basic idea of calibration for the VIO copter is that we require the camera calibration and camera to IMU calibration. To do this we will take a video of a thing called an Alice grid while recording camera images and IMU measurements. We will then combine these into a ROS bag and finally run the Kalibr scripts.

Recording the Data

To record the data we will use the record_vio.py script. The first thing we will need to do is to ssh into the raspberry pi. This can be done using the following command:

```
$ ssh -X pi@raspberrypi.local
```

And then entering the password 'raspberry'. We will then cd to vio_recording/VIO-tools/ on the terminal connected to the raspberry pi. To run the record_vio.py file use the following command on the pi terminal:

```
$ python3 record_vio.py
```

Or

```
$ python3 record_vio.py -i
```

Where the -i is added if the recording is going to be made indoors. This will record a file named by the current date and time when the recording is made. We will then want to transfer this file back to our local laptop for processing. To do this we use the command

```
$ scp -r pi@raspberrypi.local:~/data_location_on_pi /location_on_laptop_to_store_at
```

For tips on recording a good data set I would recommend viewing the video on Kalibr's wiki however some basic tips include:

- Stimulate all IMU directions individually

- Ensure that all sections of the camera view the April grid
- Try to have slow a steady movements that don't blur the camera.

Processing the data

Once we have the data we need to process it using Kalibr. Kalibr requires a Ros Bag for its data so we need to convert the recorded data into a Ros Bag. To do this we can use the mav2ROS.py script. To run this use the following format.

```
$ python mav2ROS.py mav_imu.csv frames/ cam.csv
```

Once a ROS bag is generated we can run the Kalibr scripts. Kalibr works best on ubuntu 16. It has apparently been updated for ubuntu 20.4 however I have had limited success with this. To ensure that the Kalibr files are added to the path run:

```
$ source ~/kalibr_workspace/devel/setup.bash
```

We will now run the script to calibrate the camera. This requires 4 arguments:

- --bag ~ this is the ROS bag name
    - --topics ~This is the name of the topics of the camera pics in the ROS bag. If mav2ROS was uses then this will just be /cam
- --models ~ This is the model we want to use for the camera. There are two parts to this model, the actual camera model and the distortion model. The camera model is the type of projection model we want to use. In general either pinhole or omni should be used here. We also have the distortion model which details the way the lens distorts light. For a fisheye camera equidistant or radial-tangential is an appropriate choice.
- --target ~ this is the calibration target we are using. I would recommend using an April grid target. The file for this can be downloaded of the Kalibr website.

So, the camera calibration command will look something like:

```
$ kalibr_calibrate_cameras --bag mav_vio.bag --topics /cam --models pinhole-equi --target April_6x6.yaml
```

This will give you a report and a file detailing the results called something like camchain-mav_vio.yaml. We can now run the camera to IMU calibration however first we must create the IMU parameters file. The format f this file is important. The file will need a format similar to that detailed below:

```
#Accelerometers
accelerometer_noise_density: 1.86e-03   #Noise density (continuous-time)
accelerometer_random_walk:   4.33e-04   #Bias random walk

#Gyroscopes
gyroscope_noise_density:     1.87e-04   #Noise density (continuous-time)
gyroscope_random_walk:       2.66e-05   #Bias random walk

rostopic:                    /imu       #the IMU ROS topic
```

```
update_rate:                 200.0      #Hz (for discretization of the
values above)
```

Once we have this saved as a .yaml we can run the camera to IMU calibration. The format for this will be something like:

```
$ kalibr_calibrate_imu_camera --bag mav_vio.bag --cam camchain-mav_vio.yaml --imu
imu_param.yaml --target April_6x6.yaml
```

This will again produce a report with graphs and a file detailing the results.

A good camera calibration will have the following characteristics:

- Low reprojection error (~0.5 pixels) with even distribution
- Good coverage of the whole space. This can be seen I the graph on the left of page 3 of the report.

A good camera to IMU calibration will have:

- Translational aspects that are logical
- A rotation element that is logical when converted to Euler angles as well as obeys the rule R*R.T=I