Documentation Status

**Contents**

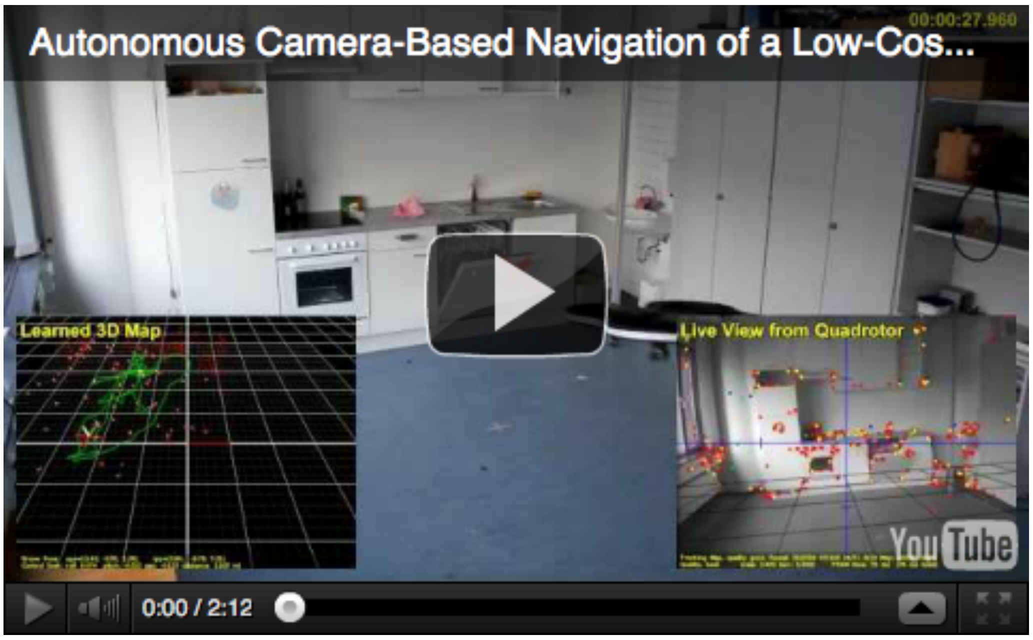
## 1. Overview

This package contains the implementation corresponding to the following publications:

- Camera-Based Navigation of a Low-Cost Quadrocopter (J. Engel, J. Sturm, D. Cremers)
- Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing (J. Engel, J. Sturm, D. Cremers)

Abstract— In this paper, we describe a system that enables a low-cost quadrocopter coupled with a ground-based laptop to navigate autonomously in previously unknown and GPS- denied environments. Our system consists of three components: a monocular SLAM system, an extended Kalman filter for data fusion and state estimation and a PID controller to generate steering commands. Next to a working system, the main contribution of this paper is a novel, closed-form solution to estimate the absolute scale of the generated visual map from inertial and altitude measurements. In an extensive set of experiments, we demonstrate that our system is able to navigate in previously unknown environments at absolute scale without requiring artificial markers or external sensors. Furthermore, we show (1) its robustness to temporary loss of visual tracking and significant delays in the communication process, (2) the elimination of odometry drift as a result of the visual SLAM system and (3) accurate, scale-aware pose estimation and navigation.

This is a video of the AR.Drone 1.0 flying autonomously, using this package:



The code works for both the AR.Drone 1.0 and 2.0, the default-parameters however are optimized for the AR.Drone 2.0 by now. You can find more details on my research on my website.

This Package builds on the well known monocular SLAM framework PTAM presented by Klein & Murray in their paper at ISMAR07. Please study the original PTAM website and the corresponding paper for more information on this part of the software. Also, be aware of the license that comes with it.

## 2. Nodes

- drone_stateestimation: Drone State-estimation, including PTAM & Visualization.
- drone_autopilot: Drone Controller, requires drone_stateestimation
- drone_gui: GUI for controlling the drone (with a Joystick or KB) and for controlling the autopilot and the stateestimation node.

## 3. Quickstart

### 3.1 Installation

1. Install the ardrone_autonomy package: *At the moment there are some minor changes to this package required, the easiest way is to check it out from this repository, which contains a slightly older, but well-tested and already modified version*. To do this, run:

```
# cd into ros root dir
roscd

# clone repository
git clone git://github.com/tum-vision/ardrone_autonomy.git
ardrone_autonomy

# add to ros path (if required)
```

```
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:`pwd`/ardrone_autonomy

# build SDK (might require your confirmation to install some system
libraries)
cd ardrone_autonomy
./build_sdk.sh

# build package
rosmake
```

2. Install tum_ardrone package:

```
# cd into ros root dir
roscd

# clone repository
git clone git://github.com/tum-vision/tum_ardrone.git tum_ardrone

# add to ros path (if required)
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:`pwd`/tum_ardrone

# build package (may take up to 10 minutes)
rosmake tum_ardrone
```

## 3.2 Run

```
# run driver
rosrun ardrone_autonomy ardrone_driver

# run stateestimation node
rosrun tum_ardrone drone_stateestimation

# run autopilot node
rosrun tum_ardrone drone_autopilot

# run gui node
rosrun tum_ardrone drone_gui
```

## 3.3 Use

#### Manual Keyboard control

- focus drone_gui window
- press ESC to activate KB control
- fly around with KB (see here for key assignments)

#### Manual Joystick control

assuming a plugged-in PS3 six-axis controller, with set rights.

- `rosrun joy joy_node`
- press PS button on controller to activate it

- fly around (see here for button assignments)

**Autopilot**

- place the drone on the ground, with enough open space around it. There should be some structure with enough keypoints in front of it, ideally at a distance of 2m to 10m.
- read Tips for good PTAM and Scale Estimation Performance
- load contents of file *initDemo.txt* (left, below the big text field in the GUI)
- click *Clear and Send* (best to click *Reset* first). => drone will takeoff & initialize PTAM, then fly a small figure (1m up, 1m down, 1x1m horizontal square).
- you can *interrupt* the figure anytime by interactively setting a relative target: click on video (relative to current position); see here. **First fly up at least 1m to facilitate a good scale estimate, do not start e.g. by flying horizontally over uneven terrain.**
- always have a finger on ESC or on the joystick for emergency-keyboard control.

## 4. Troubleshooting

**Drone does not start**

- battery empty? (drone does not start if battery < ~18%)
- drone in emergency state? (if so, the four LED's are red. click Emergency to reset).

**Cannot control drone**

- selected correct control source? maybe re-select.

**Drone flies unstable using the autopilot**

- adjust / reduce control parameters using dynamic_reconfigure (see here). First, try reducing `aggressiveness`, which should make the drone fly slower, but also more stable. If that does not help, reduce the differential weights, in particular for roll/pitch control (`Kp_rp`). This may be necessary in particular if PTAM's pose estimates are inaccurate (e.g. because all keypoints are very far away).

**drone_stateestimation: PTAM initialization fails: "IMU Baseline smaller than 5cm, try again: XXX"**

- In order to initialize PTAM, the stateestimation node needs some metric scale information (i.e. horizontal velocity estimates or ultrasound altimeter measurements). These measurements however are only sent while flying (at least for the 2.0 drone), hence PTAM can only run properly / be initialized while flying. You may also record a flight and then replay it, if you want to test without actually flying.

**drone_stateestimation: XXXs between two consecutive navinfos. This system requires Navinfo at 200Hz...**

- usually, with navdata_debug set to 0, the drone sends a new navdata package every 5ms. The above message means that there was no navdata received for XXXs, which, if it happens continuously, leads to a poor state estimate and flight performance. Possible reasons: If the publish frequency on /ardrone/navdata (displayed on UI window) is around 50, this means the driver's loop_rate is set incorrectly; if it is around 15, navdata_demo is set to 1. Another reason might be a bad network connection / high pings, try restarting the drone or going somewhere with less WLan clutter.

#### Drone broken

- buy a new one.

## 5. Notes

- the channel */tum_ardrone/com* is used for communication between the nodes. to see what's going on internally, it is generally a good idea to echo this channel.

## 6. Known Bugs & Issues

- as the software was originally developed for the Ar.Drone 1.0, the pressure sensor and magnetometer are not used.
- drone_stateestimation crashes occasionally when PTAM init fails (crash occurs in PTAM code). Happens in particular if there is no baseline in between the first two keyframes, hardly ever happens in praxis.

## 7. License

The major part of this software package - that is everything except PTAM - is licensed under the GNU General Public License Version 3 (GPLv3), see http://www.gnu.org/licenses/gpl.html. PTAM (comprised of all files in /src/stateestimation/PTAM) has it's own licence, see http://www.robots.ox.ac.uk/~gk/PTAM/download.html. This licence in particular prohibits commercial use of the software.

Wiki: tum_ardrone (last edited 2013-02-12 11:40:44 by JakobEngel)