Introduction to the Unmanned Aerial Platform in the MRS Lab

From control theory to practical experiments

Tomas Baca[1], Petr Stibinger[1], Robert Penicka[1]

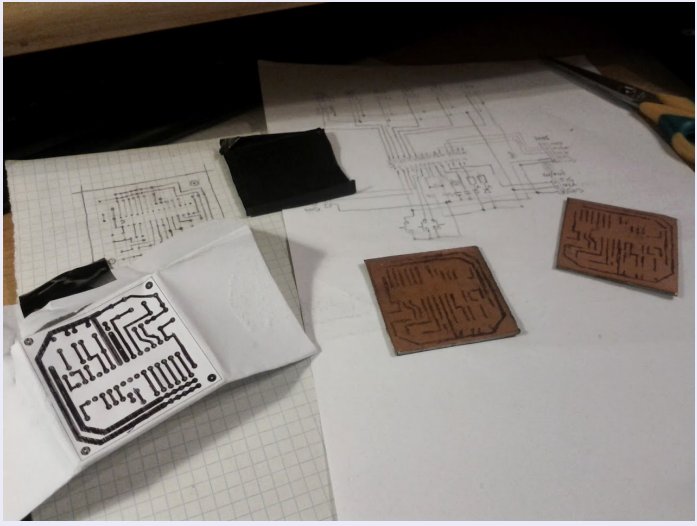
[1]Multi-Robot Systems group, Faculty of Electrical Engineering
Czech Technical University in Prague

tomas.baca@fel.cvut.cz

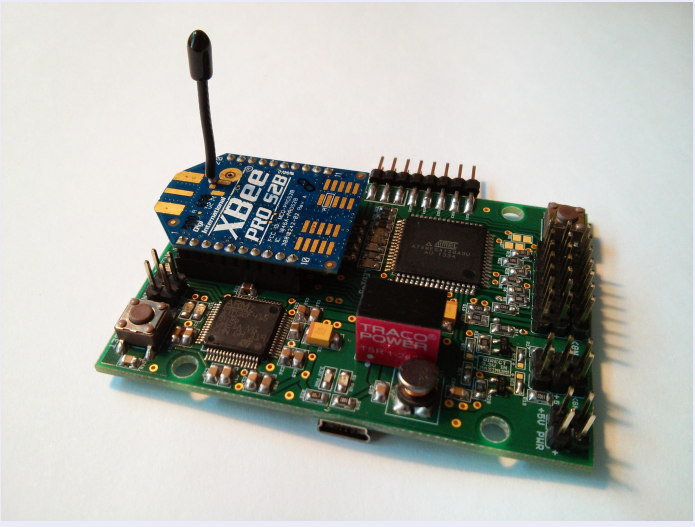
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE | MRS MULTI-ROBOT SYSTEMS GROUP

Outline

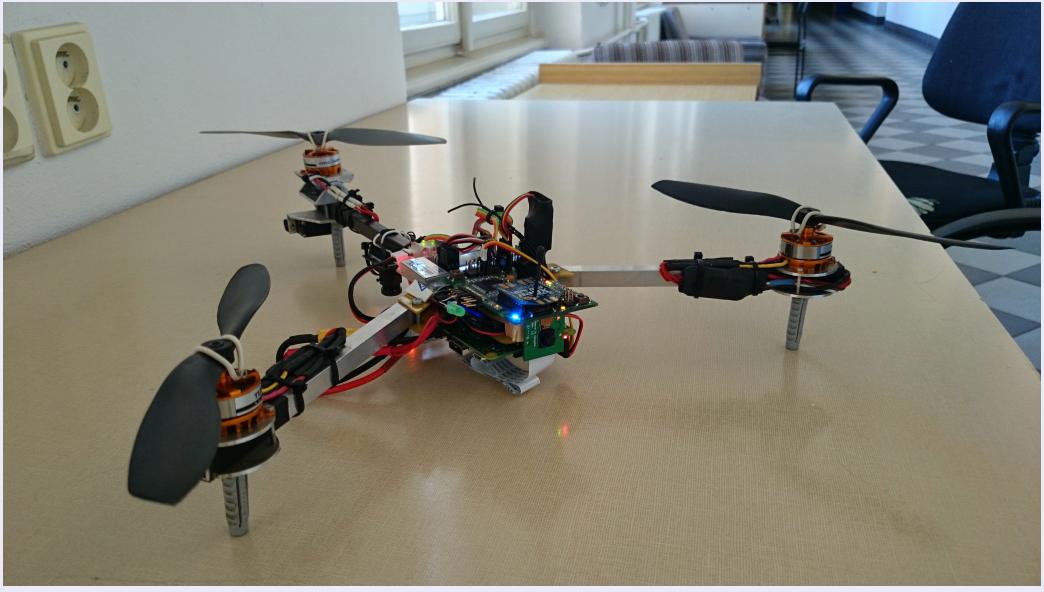
≈ 2012, designing custom PCBs for controllers



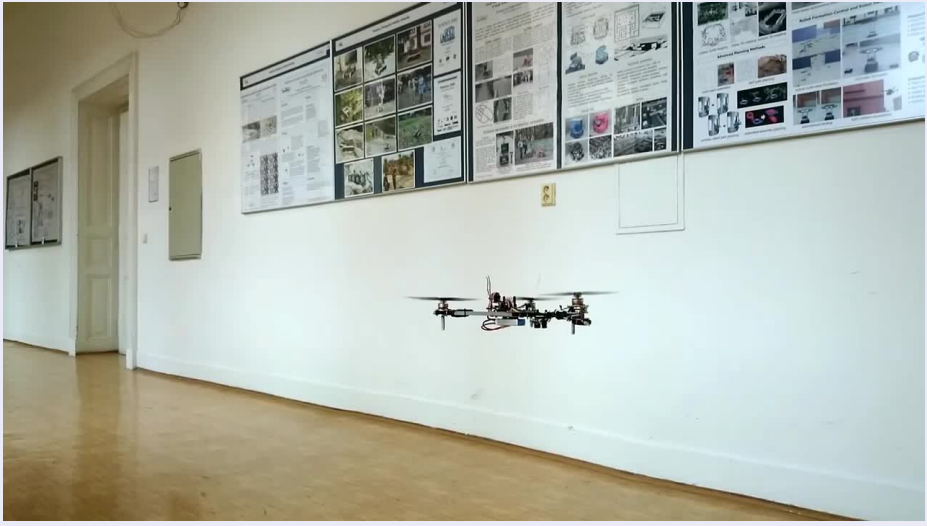
≈ 2014, Model Predictive Control on embedded hardware



≈ 2014, Model Predictive Control on embedded hardware

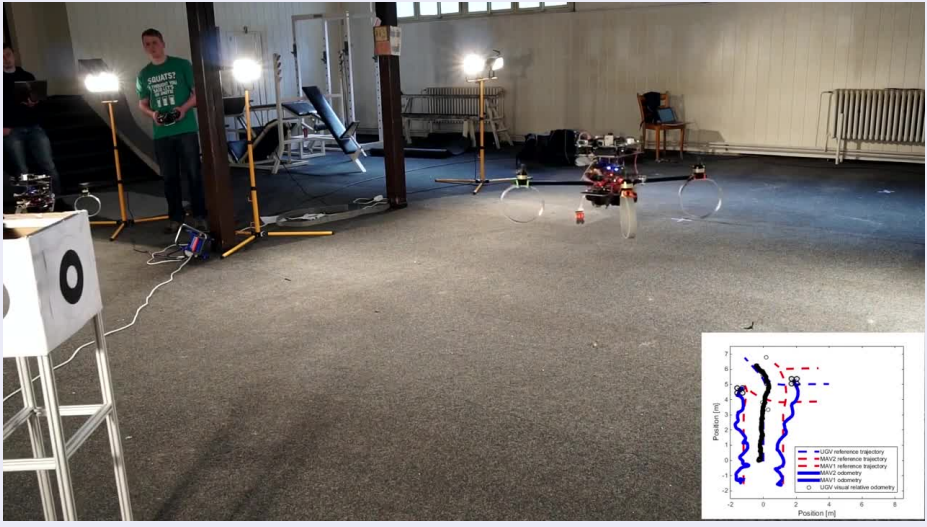


Embedded Model Predictive Control



Video: `http://youtu.be/AXI_rkQRBaE`

Embedded Model Predictive Control



Video: `http://youtu.be/9Bpm4J31CgE`

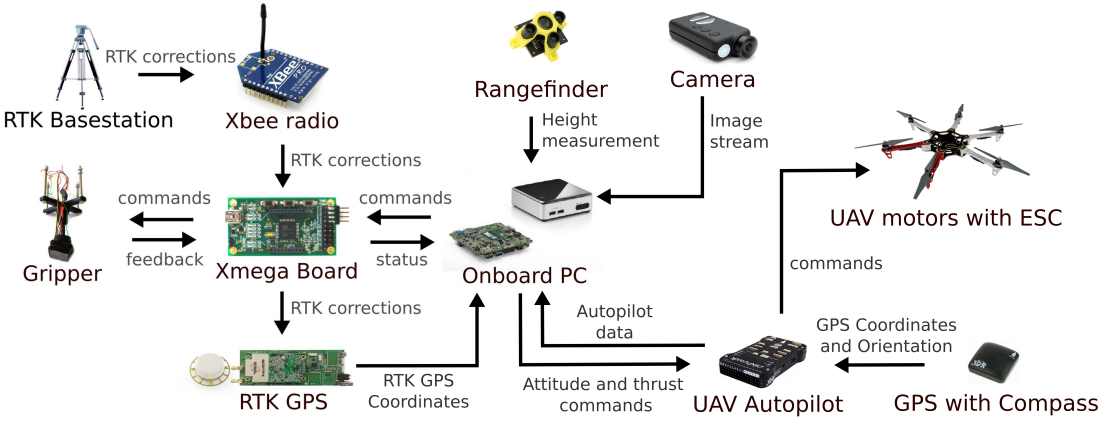
- Custom and purpose-built hardware

- Custom and purpose-built hardware
- Embedded software

- Custom and purpose-built hardware
- Embedded software
- Results: [Baca et al., 2016], [Spurny et al., 2016], [Saska et al, 2016], [Chudoba et al., 2016]

- Custom and purpose-built hardware
- Embedded software
- Results: [Baca et al., 2016], [Spurny et al., 2016], [Saska et al, 2016], [Chudoba et al., 2016]
- Not Scalable! Bottlenecks everywhere: Embedded programming *cumbersome*, simulating is difficult, system is hard-to-maintain, connecting sensors and peripheries is limited...

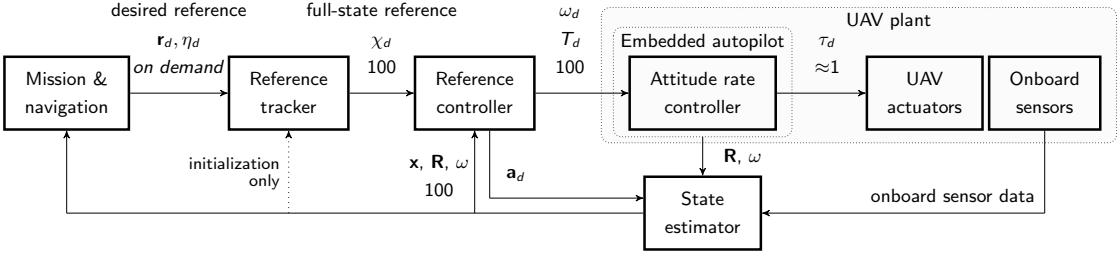
A typical pipeline structure revolves around a Linux computer.



ROS – Robot Operating System

- middleware allowing communication between programs
- integrates with C++, Python, Bash and Zshell
- makes the transition from *Matlab* to reality bearable
- supported by sensor manufacturers
- integration through the Linux terminal
- out of the box: time and clock management, logging, recording onboard data, visualization and plotting, parameter loading, static and dynamic transformations, etc.
- integrates to robotic simulators: Gazebo, Coppelia (V-REP)

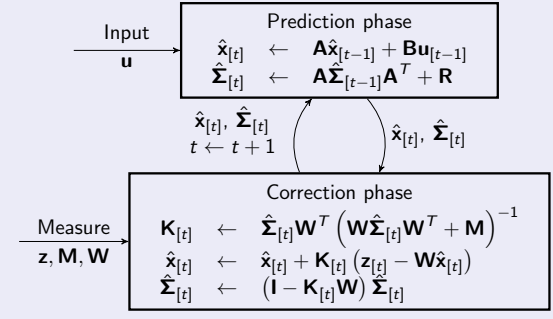
Control theory point of view



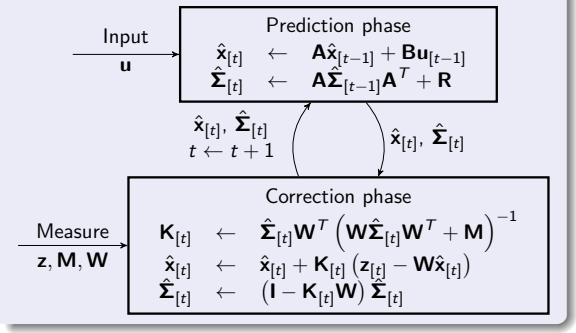
[Baca et al., 2020, arXiv:2008.08050]

The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles, preprint arXiv:2008.08050, 2020

Linear Kalman Filter ≈ 10 lines



Linear Kalman Filter ≈ 10 lines



Linear Kalman Filter in practice

- $\geq$ 1000 lines of `C++` code
- Real-world fusion & estimation
  $\geq$ 10 000 lines of `C++` code

Geometric tracking controller $\approx$ 10 lines

$$\mathbf{f}_d = -m_e \mathbf{k}_p \circ \mathbf{e}_{pe} \mathbf{k}_v \circ \mathbf{e}_v + m_e \ddot{\mathbf{r}}_{de} g \hat{\mathbf{e}}_3 - \mathbf{d}_w \circ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} - \mathbf{d}_b \circ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},$$

$$\mathbf{e}_R = \frac{1}{2} \left(\mathbf{R}_d^\mathsf{T} \mathbf{R} - \mathbf{R}^\mathsf{T} \mathbf{R}_d \right),$$

$$\omega_d = -\mathbf{k}_R \circ \mathbf{e}_R + \omega_j - \omega_c,$$

Geometric tracking controller ≈ 10 lines

$$\mathbf{f}_d = -m_e\mathbf{k}_p \circ \mathbf{e}_{pe}\mathbf{k}_v \circ \mathbf{e}_v + m_e\ddot{\mathbf{r}}_{de}g\hat{\mathbf{e}}_3 - \mathbf{d}_w \circ \begin{bmatrix}1\\1\\0\end{bmatrix} - \mathbf{d}_b \circ \begin{bmatrix}1\\1\\0\end{bmatrix},$$

$$\mathbf{e}_R = \frac{1}{2}\left(\mathbf{R}_d^\mathsf{T}\mathbf{R} - \mathbf{R}^\mathsf{T}\mathbf{R}_d\right),$$

$$\omega_d = -\mathbf{k}_R \circ \mathbf{e}_R + \omega_j - \omega_c,$$

MPC controller ≈ 100 lines

$$\min_{\mathbf{u}_{[1:n]}} \quad \frac{1}{2}\sum_{i=1}^{n-1}\left(\mathbf{e}_{[i]}^\mathsf{T}\mathbf{Q}\mathbf{e}_{[i]}\right) + \mathbf{e}_{[n]}^\mathsf{T}\mathbf{S}\mathbf{e}_{[n]}$$

$$\begin{aligned}
\text{s.t. } \mathbf{x}_{m[i]} &= \mathbf{A}_m\mathbf{x}_{m[i-1]} + \mathbf{B}_m\mathbf{u}_{[i]}, && \forall i \in \{1,\dots,n\}\\
\mathbf{x}_{m[i]} &\leq \mathbf{x}_{\max}, && \forall i \in \{1,\dots,n\}\\
\mathbf{x}_{m[i]} &\geq -\mathbf{x}_{\max}, && \forall i \in \{1,\dots,n\}\\
\mathbf{u}_{[i]} - \mathbf{u}_{[i-1]} &\leq \dot{\mathbf{u}}_{\max}\Delta t, && \forall i \in \{2,\dots,n\}\\
\mathbf{u}_{[i]} - \mathbf{u}_{[i-1]} &\geq -\dot{\mathbf{u}}_{\max}\Delta t, && \forall i \in \{2,\dots,n\}
\end{aligned}$$

Geometric tracking controller ≈ 10 lines

$$\mathbf{f}_d = -m_e\mathbf{k}_p \circ \mathbf{e}_{pe}\mathbf{k}_v \circ \mathbf{e}_v + m_e\ddot{\mathbf{r}}_{de}g\hat{\mathbf{e}}_3 - \mathbf{d}_w \circ \begin{bmatrix}1\\1\\0\end{bmatrix} - \mathbf{d}_b \circ \begin{bmatrix}1\\1\\0\end{bmatrix},$$

$$\mathbf{e}_R = \frac{1}{2}\left(\mathbf{R}_d^\mathsf{T}\mathbf{R} - \mathbf{R}^\mathsf{T}\mathbf{R}_d\right),$$

$$\omega_d = -\mathbf{k}_R \circ \mathbf{e}_R + \omega_j - \omega_c,$$

MPC controller ≈ 100 lines

$$\min_{\mathbf{u}_{[1:n]}} \quad \frac{1}{2}\sum_{i=1}^{n-1}\left(\mathbf{e}_{[i]}^\mathsf{T}\mathbf{Q}\mathbf{e}_{[i]}\right) + \mathbf{e}_{[n]}^\mathsf{T}\mathbf{S}\mathbf{e}_{[n]}$$

$$\begin{aligned}
\text{s.t. } \mathbf{x}_{m[i]} &= \mathbf{A}_m\mathbf{x}_{m[i-1]} + \mathbf{B}_m\mathbf{u}_{[i]}, && \forall i \in \{1,\dots,n\}\\
\mathbf{x}_{m[i]} &\leq \mathbf{x}_{\max}, && \forall i \in \{1,\dots,n\}\\
\mathbf{x}_{m[i]} &\geq -\mathbf{x}_{\max}, && \forall i \in \{1,\dots,n\}\\
\mathbf{u}_{[i]} - \mathbf{u}_{[i-1]} &\leq \dot{\mathbf{u}}_{\max}\Delta t, && \forall i \in \{2,\dots,n\}\\
\mathbf{u}_{[i]} - \mathbf{u}_{[i-1]} &\geq -\dot{\mathbf{u}}_{\max}\Delta t, && \forall i \in \{2,\dots,n\}
\end{aligned}$$

UAV Control in practice

- $\geq$ 10 000 lines of C++ code

What needs to be solved outside of Matlab's sandbox?

- crashes are expensive, so don't crash
- control references might not be feasible
- sensors can get disconnected during the flight
- takeoff and landing: **the most tricky part of the flight**
- mass (thus the model) can change during the flight
- controllers can be poorly tuned... handle instabilities
- acceleration and speed depend on the available sensors
- people are fallible, don't let them crash the drones
- not all states of UAV are allowed, even though controllers can reach them (upside down)

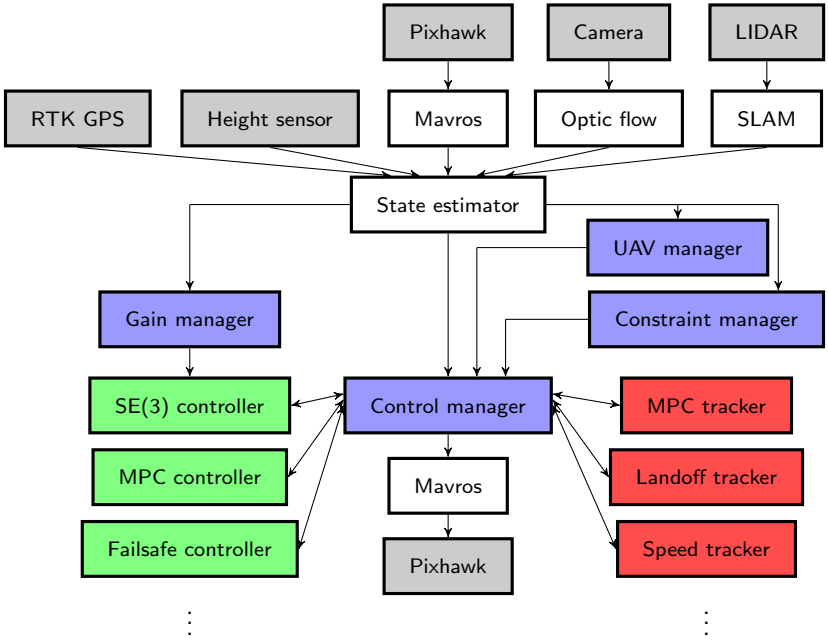
The failsafe core

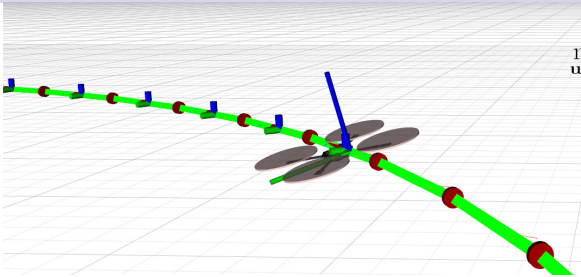
```
if (goingToCrash())
    dont();
```

The MRS UAV System

- open-source system for multirotor UAVs
- *relatively* smooth transition from simulations to real world
- complete onboard execution
- outdoor/indoor/lab/real-world environment
- emphasis on safety and autonomy during experiments
- painless experiments from takeoff to landing
- extensible and powerful simulation
- relies on the PixHawk embedded flight controller
- safe for development and testing of new controllers

Control – implementation point of view





$$\min_{\mathbf{u}_{[1:n]}} \quad \mathrm{V}\left(\mathbf{x}_{[1:n]}, \mathbf{u}_{[1:n]}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{e}_{[i]}^{T}\mathbf{Q}\mathbf{e}_{[i]} + \mathbf{u}_{[i]}^{T}\mathbf{P}\mathbf{u}_{[i]}\right)$$

$$\begin{aligned} \text{s.t. } \mathbf{x}_{[t]} &= \mathbf{A}\mathbf{x}_{[t-1]} + \mathbf{B}\mathbf{u}_{[t]}, && \forall t \in \{0,\ldots,n\} \\ \mathbf{x}_{[t]} &\leq \mathbf{x}_{\max[t]}, && \forall t \in \{1,\ldots,n\} \\ \mathbf{x}_{[t]} &\geq \mathbf{x}_{\min[t]}, && \forall t \in \{1,\ldots,n\} \end{aligned}$$

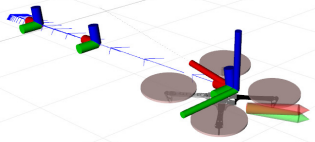
- receive high-level references from a navigation software — a single **position** & **heading** reference, a **trajectory**, several **position** & **heading** sampled with constant
- generate reference for controllers — desired $\dot{x}, \ddot{x}, \dddot{x}, \ddddot{x}$, η, $\dot{\eta}$
- are provided with a set of state constraints that should be satisfied at all times.
- loaded as plugins to the *ControlManager*

Available trackers

MPC tracker, Landoff tracker, Joy tracker, Speed tracker, ...

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_trackers`



$$\mathbf{f}_d = \overbrace{-\mathbf{k}_p \circ \mathbf{e}_p}^{\text{position feedback}} + \overbrace{-\mathbf{k}_v \circ \mathbf{e}_v}^{\text{velocity feedback}} + \overbrace{m_e \ddot{\mathbf{r}}_d}^{\text{reference feedforward}} + \underbrace{-m_e g \hat{\mathbf{e}}_3}_{\text{gravity compensation}} + \underbrace{-\mathbf{d}_w \circ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}}_{\text{world disturbance compensation}} + \underbrace{-\mathbf{d}_b \circ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}}_{\text{body disturbance compensation}}$$

- receive a current UAV state estimate
- receive the all-state reference from reference trackers
- control the states of the UAV by outputting a desired angular rate (or desired orientation) and desired collective thrust
- loaded as plugins to the *ControlManager*

Available controllers

SE(3) controller, MPC controller, Emergency controller, Failsafe controller

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_controllers`

MRS UAV System — Control Manager

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE | MRS MULTI-ROBOT SYSTEMS GROUP

- integrates the control and tracking part of the MRS UAV pipeline
- dynamically loads **controllers** and **trackers** plugins
- subscribes to the UAV state estimate and provides it to all loaded trackers and controllers
- outputs the attitude rate command the Pixhawk flight controller
- provides constraints for trackers through a common interface
- allows in-mid-air switching of feedback controllers and reference trackers
- provides an interface between to UAV and a user
- **provides safety features**

Safety features — automatically triggered

- emergency hover
- emergency landing
- feedforward failsafe landing

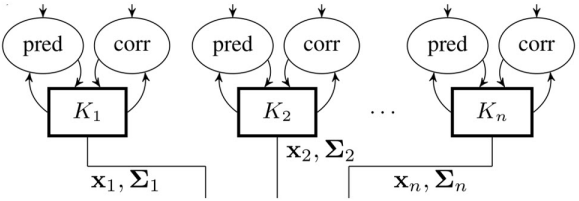
[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_managers`

- high-level non-essential routines
- takeoff state machine
- landing state machine
- return to home state machine
- flight timer
- max thrust checking
- max height checking

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_managers`



- fusion of available sensors
- estimation of UAV translational states
- bank of estimators
- allows switching estimators in mid-flight
- automatically switches when state estimator becomes unreliable

Available estimators

Optic flow, optic flow & GPS, GPS, RTK GPS, VIO, Visual Servoing, 2D SLAM, 3D SLAM, Visual SLAM, Visual Servoing & optic flow, 2D ICP

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_odometry`

Constraint & Gain managers

- dynamics constraint & gain scheduling
- automatically switches with every switch of a state estimator

Dynamics constraints

- horizontal: speed, acceleration, jerk, snap
- vertical ascending: speed, acceleration, jerk, snap
- vertical descending speed, acceleration, jerk, snap
- heading: speed, acceleration, jerk, snap
- attitude rate: roll, pitch, yaw rate
- attitude: max tilt

SE(3) controller gains

- translation: position and velocity feedback
- rotation: attitude feedback
- mass estimator
- body & world disturbance estimators

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_uav_managers`

MRS UAV System — Simulation

- 3 UAV types: DJI f450, DJI f550, Tarot 650

Available sensors

- RGB camera: MatrixVision Bluefox, mobius
- Stereo Cameras: Intel Realsense
- 1D LiDARs: Terabee Teraranger, Gamin Lite
- 2D LiDARs: Scanse Sweep, Garmin RPLidar
- 3D LiDARs: Velodyne Puck, Ouster
- Thermal cameras, UV cameras for UVDAR

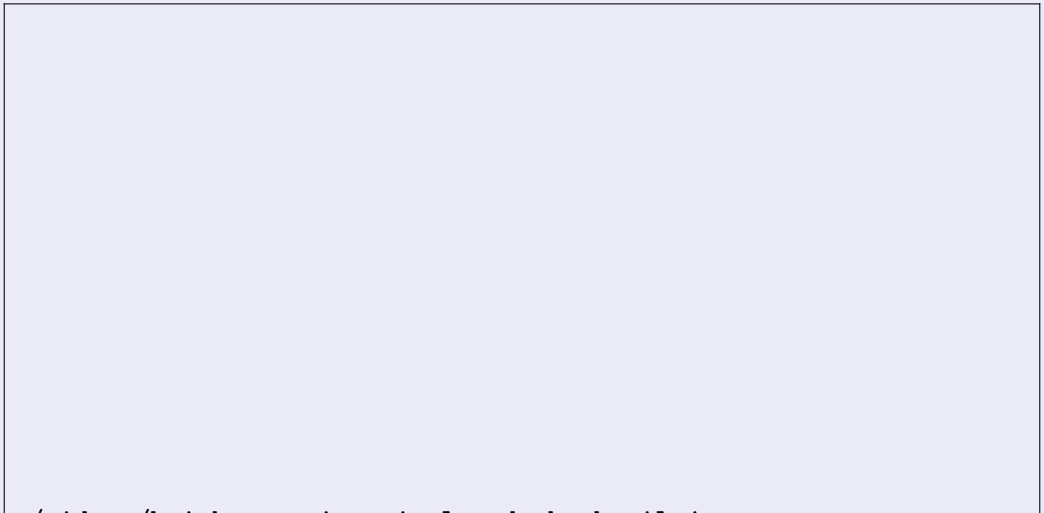
Available actuation

- magnetic gripper
- parachute
- water gun

[Baca et al., 2020, arXiv:2008.08050]

`http://github.com/ctu-mrs/mrs_simulation`

Realistic simulations of UAV grasping a brick



Real world UAV grasping a brick

The MRS UAV System paper preprint — arXiv:2008.0805

The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles

Tomas Baca[a,*], Matej Petrlik[a], Matous Vrba[a], Vojtech Spurny[a], Robert Penicka[a], Daniel Hert[a] and Martin Saska[a]

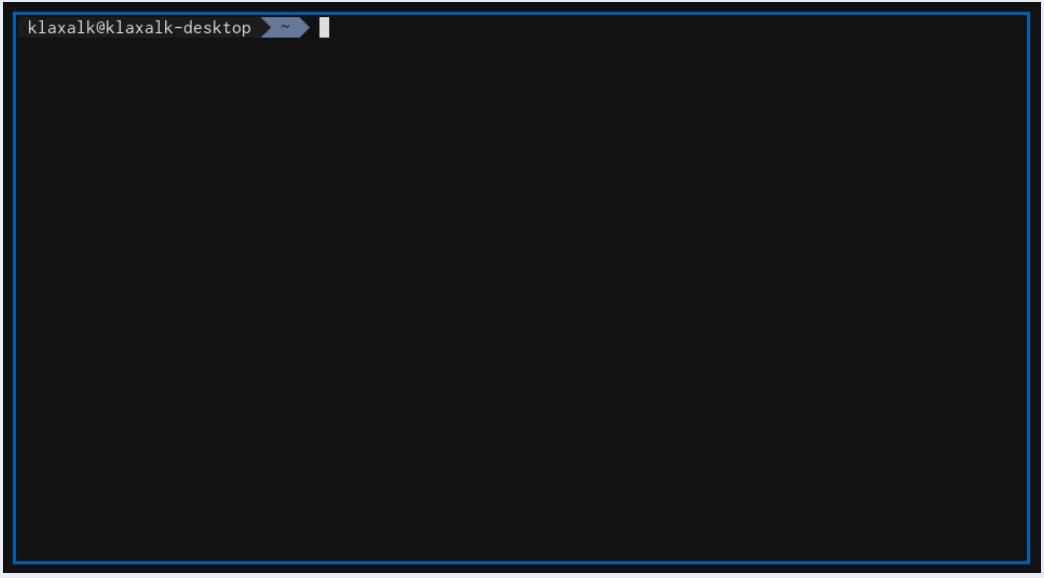
ARTICLE INFO

Keywords:
Unmanned Aerial Systems
Automatic control
Educational robotics

ABSTRACT

We present a multirotor Unmanned Aerial Vehicle (UAV) control and estimation system for supporting replicable research through realistic simulations and real-world experiments. We propose a unique multi-frame localization paradigm for estimating the states of a UAV in various frames of reference using multiple sensors simultaneously. The system enables complex missions in GNSS and GNSS-denied environments, including outdoor-indoor transitions and the execution of redundant estimators for backing up unreliable localization sources. Two feedback control designs are presented: one for precise and aggressive maneuvers, and the other for stable and smooth flight with a noisy state estimate. The proposed control and estimation pipeline are constructed without using the Euler/Tait-Bryan angle representation of orientation in 3D. Instead, we rely on rotation matrices and a novel heading-based convention to represent the one free rotational degree-of-freedom in 3D of a standard multirotor helicopter. We provide an actively maintained and well-documented open-source implementation (http://github.com/ctu-mrs/mrs_uav_system), including realistic simulation of UAVs, sensors, and localization systems. The proposed system is the product of years of applied research on multi-robot systems, aerial swarms, aerial manipulation, motion planning, and remote sensing. All our results have been supported by real-world system deployment that subsequently shaped the system into the form presented here. In addition, the system was utilized during the participation of our team from the Czech Technical University in Prague in the prestigious MBZIRC 2017 and 2020 robotics competitions, and also in the DARPA Subterranean challenge. Each time, our team was able to secure top places among the best competitors from all over the world. On each occasion, the competitions and challenges has motivated the team to improve the system and to gain a great amount of high-quality experience within tight deadlines.

A Linux terminal



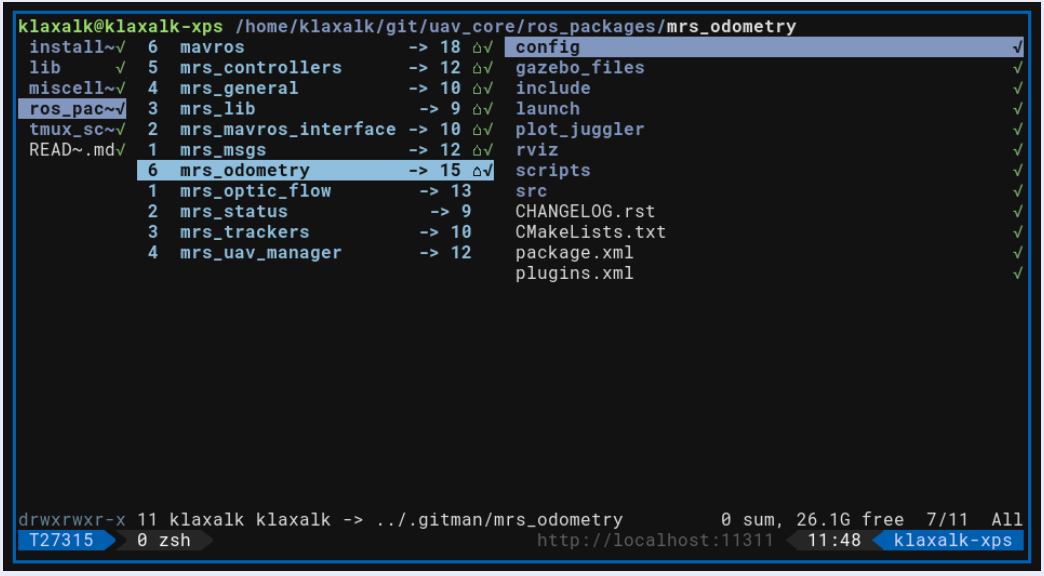
Tmux in the Linux terminal

```
klaxalk@klaxalk-desktop ~/git ls
bibliography     mrs-presentation  rob_dissertation            uav_core              xray-decoder
dissertation     notes             simulation                  uav_modules
linux-setup      papers            summer_school_seminar_task  vzlusat-presentation
mrs_cheatsheet   radiation         tmp                         waypoint_flier
klaxalk@klaxalk-desktop ~/git
```

```
klaxalk@klaxalk-desktop ~                    klaxalk@klaxalk-desktop ~
```

```
T31936  0 zsh  1 zsh  2 zsh                  http://localhost:11311  14:41  klaxalk-desktop
```

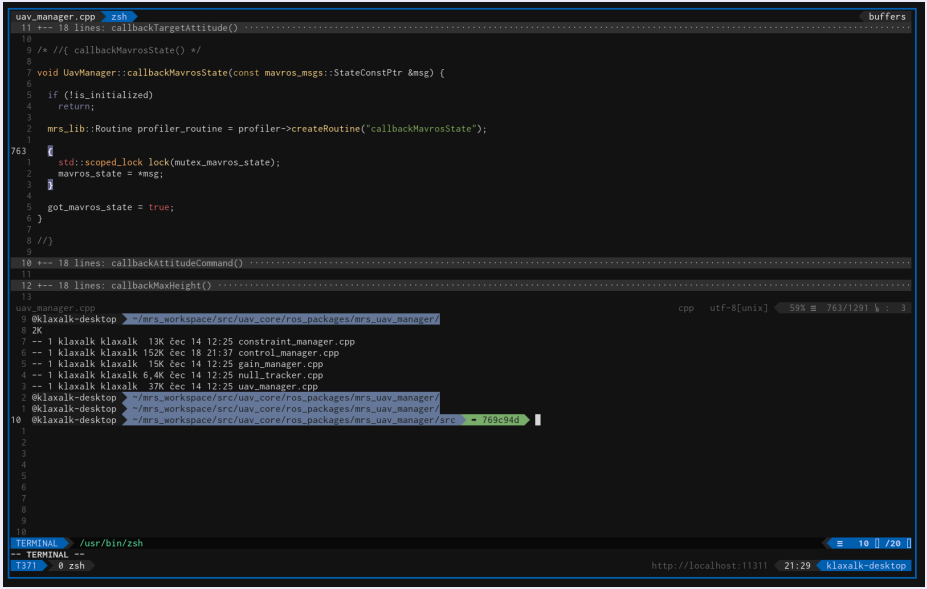
Ranger


klaxalk@klaxalk-xps /home/klaxalk/git/uav_core/ros_packages/mrs_odometry
install~√ 6 mavros -> 18 ⌂√
lib √ 5 mrs_controllers -> 12 ⌂√
miscell~√ 4 mrs_general -> 10 ⌂√
ros_pac~√ 3 mrs_lib -> 9 ⌂√
tmux_sc~√ 2 mrs_mavros_interface -> 10 ⌂√
READ~.md√ 1 mrs_msgs -> 12 ⌂√
6 mrs_odometry -> 15 ⌂√
1 mrs_optic_flow -> 13
2 mrs_status -> 9
3 mrs_trackers -> 10
4 mrs_uav_manager -> 12
config √
gazebo_files √
include √
launch √
plot_juggler √
rviz √
scripts √
src √
CHANGELOG.rst √
CMakeLists.txt √
package.xml √
plugins.xml √
drwxrwxr-x 11 klaxalk klaxalk -> ../.gitman/mrs_odometry 0 sum, 26.1G free 7/11 All
T27315 0 zsh http://localhost:11311 11:48 klaxalk-xps

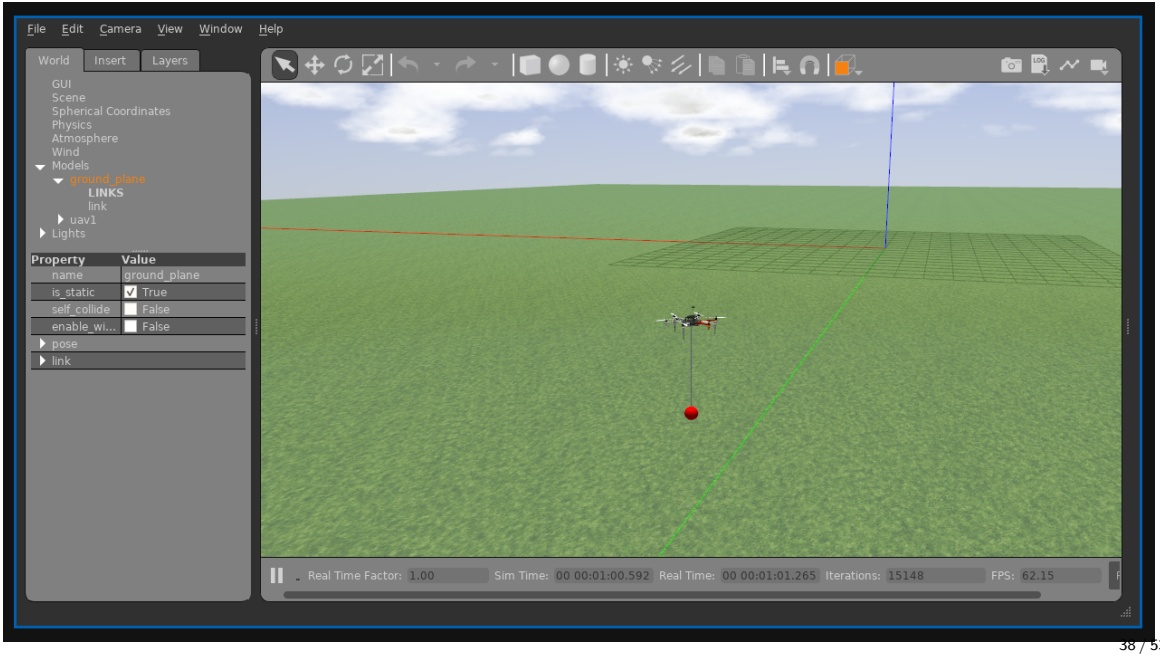
Vim in (Tmux in (Linux Terminal)

```
uav_manager.cpp+                                                    buffers
  11
  10   if (!is_initialized)
   9     return;
   8
   7   mrs_lib::Routine profiler_routine = profiler->createRoutine("callbackMavrosState");
   6
   5   {
   4     std::scoped_lock lock(mutex_mavros_state);
   3     mavros_state = *msg;
   2   }
   1
768    mavros_state.
   1                  armed                                  m _armed_type
   2   got_mavros_s   connected                              m _connected_type
   3 }                guided                                 m _guided_type
   4                  header                                 m _header_type
   5 //}              mode                                   m _mode_type
   6                  operator=( State_<allocator<void> > && ) f State_<allocator<void> > &
   7 +-- 18 lines:    system_status                          m _system_status_type
   8                  State_::                               s
   9 +-- 18 lines:    ~State_()                              f void
  10
INSERT COMPL  769c94d <]   callbackMavrosState() < cpp  utf-8[unix]  59% ≡ 768/1293 [] : 16
-- INSERT --
T31936  0 zsh                                  http://localhost:11311  14:42  klaxalk-desktop
```

Terminal in (Vim in (Tmux in (Linux Terminal))



Gazebo/ROS simulator



Spawning drones to the simulator

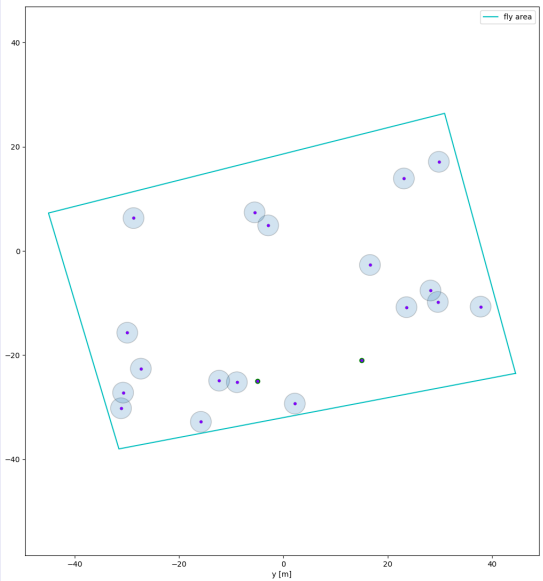

```
klaxalk@klaxalk-desktop ~ spawn --help                                    [90/106]
usage: Spawn vehicles for team. [-h] [--f450]
                                [--gazebo-ros-master-uri GAZEBO_ROS_MASTER_URI]
                                [--mavlink-address MAVLINK_ADDRESS]
                                [--generate-launch-file] [--run] [--delete]
                                [--file FILE] [--enable-bluefox-camera]
                                [--enable-bluefox-camera-reverse]
                                [--enable-bluefox-wall] [--enable-whycon-box]
                                [--enable-mobius-camera-down]
                                [--enable-mobius-camera-front]
                                [--enable-mobius-camera-back-left]
                                [--enable-mobius-camera-back-right]
                                [--enable-realsense-front-pitched]
                                [--enable-realsense-down]
                                [--enable-realsense-top]
                                [--enable-realsense-front]
                                [--enable-realistic-realsense]
                                [--enable-ground-truth] [--enable-rangefinder]
                                [--enable-teraranger]
                                [--enable-rangefinder-up] [--enable-gripper]
                                [--enable-scanse] [--enable-rplidar]
                                [--enable-teraranger-tower-evo] [--visualize]
                                [--gps-indoor-jamming] [--enable-pendulum]
                                [--enable-timepix] [--enable-velodyne]
                                [--enable-ball-holder] [--enable-uv-leds]
                                [--led-frequencies LED_FREQUENCIES LED_FREQUENCIES]
                                [--enable-uv-camera]
                                [--uv-camera-calibration-file UV_CAMERA_CALIBRATION_FILE]
                                [--debug]
                                [VEHICLE_ID [VEHICLE_ID ...]]

T16257 0 zsh                          http://localhost:11311 23:13 klaxalk-desktop
```

Now is the time to show the Live demo.

1st law of robotics: never give a robotic demo!

A sample map of points



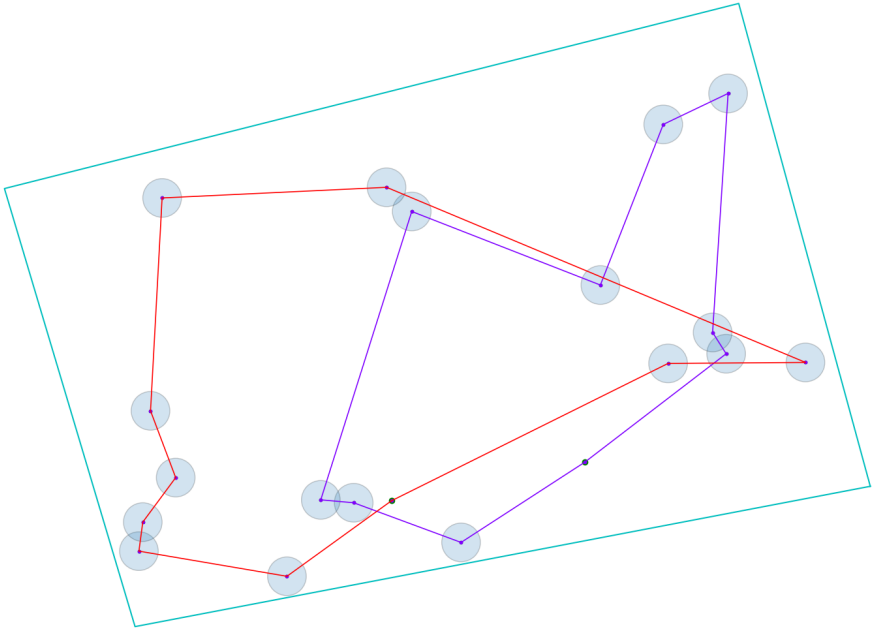
Multiple (Dubins) Traveling Salesman Problems (with Neighborhoods)

- 2 m neighborhood around each point
- each point's neighborhood has to be visited by any of the drones
- drones' min mutual distance: 5 m
- faster total time wins

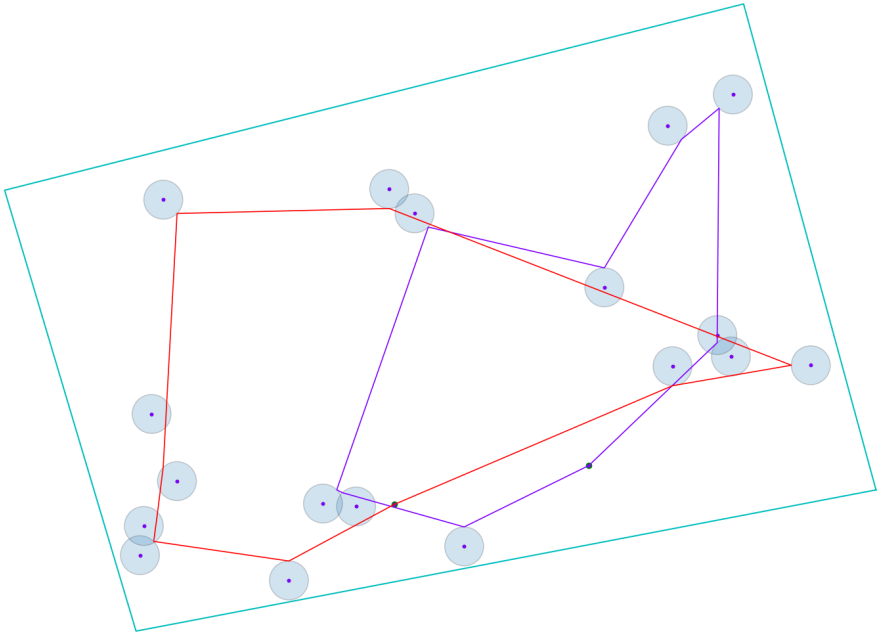
UAV's constraints

- trajectory sampled at 5 Hz
- max. speed 7 $\mathrm{m/s}$
- max. acceleration 2.5 $\mathrm{m/s^2}$
- UAV's start $=$ finish

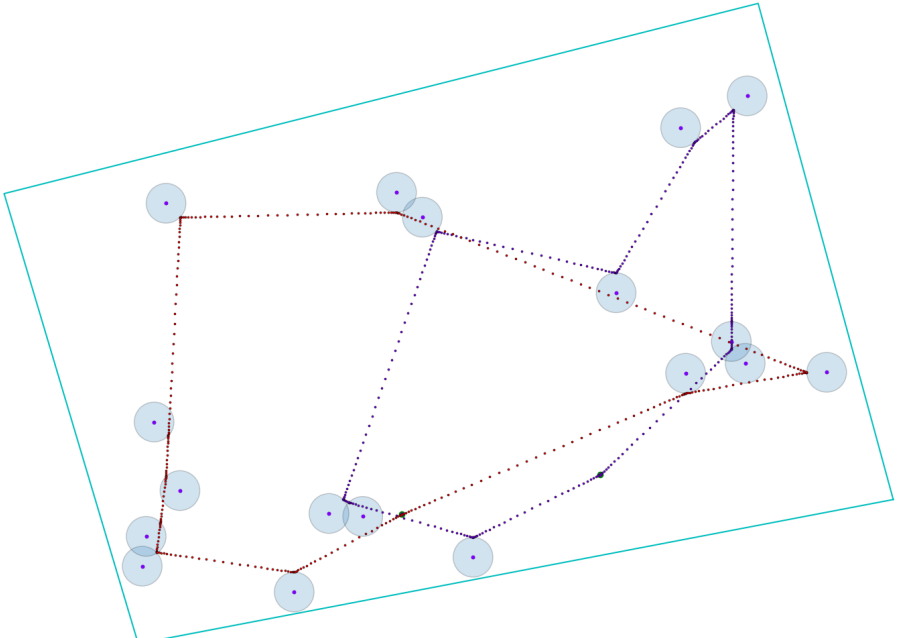
An example solution of MTSP



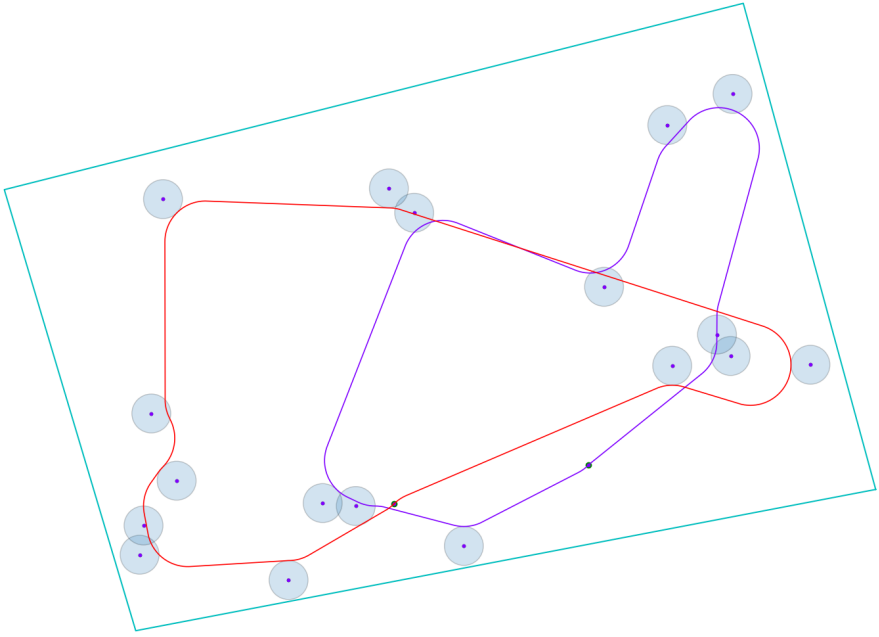
An example solution with neighborhoods



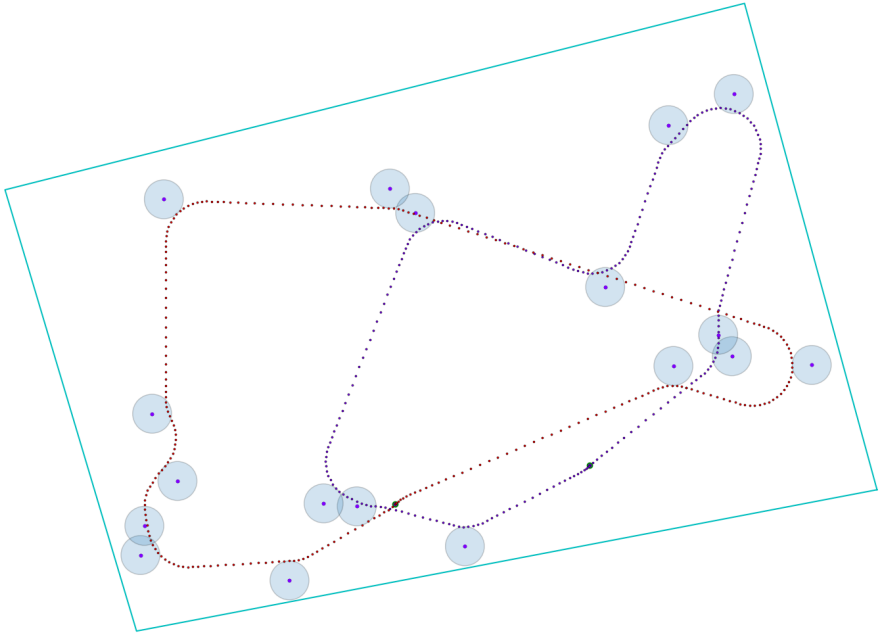
Sampling segments for the UAV's tracker



Using the Dubins vehicle model



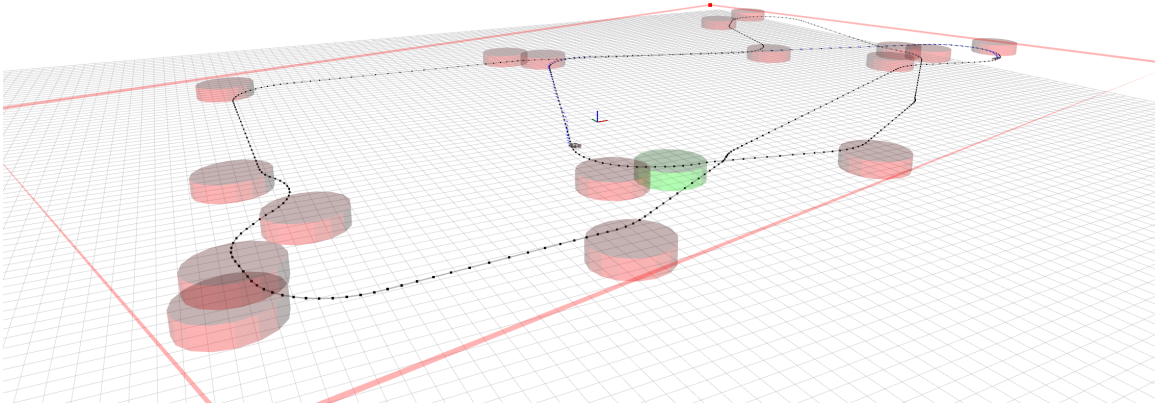
Sampling the path for the UAV's tracker



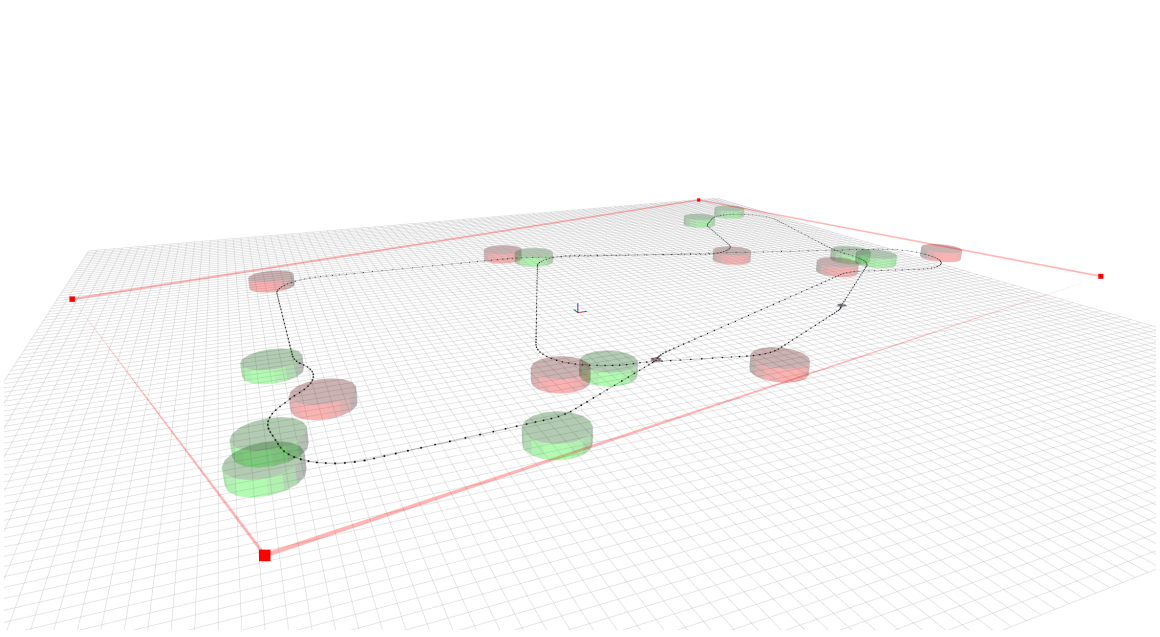
Now is the time to show the Live demo.

Message to future me: “How did the last one go? ;-)”

Plotting the trajectory in RVIZ



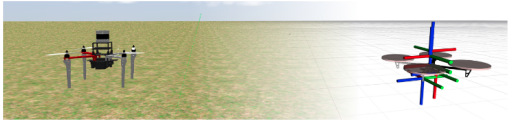
Showing Results in RVIZ



FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE | MRS MULTI-ROBOT SYSTEMS GROUP

http://ctu-mrs.github.io

Transformations



As a robotic system, the MRS system makes use of many coordinate frames refererence frames. Transformations within the MRS system are maintained by the tf2 ros package and Transformer from the mrs_lib package, which is a tf2 wrapper with several functions simplifying the work with transformations. The transformations within the MRS system are important not only for tracking the relationship among particular coordinate frames, but they are used also to enable sending commands in various reference frames independently on current control frame.

MRS Trackers commands with specified *frame_id*

The MRS UAV system provides topics and services to command a UAV in any coordinate frame with a known transformation to current control frame without need to transform the desired reference explicitly. The coordinate frame in which the desired reference is provided has to be specified as a `'frame_id'` in a header of the message. The frame_id of the coordinate frame is formed by part of the coordinate frame name without the name of UAV (e.g., `frame_id = 'fcu_untilted'` for `'<uav_name>/fcu_untilted'`).

Without specifying the *frame_id*, the frame, which is currently used for control, will be used.

If an invalid *frame_id* is given, the reference is not used.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE
MRS MULTI-ROBOT SYSTEMS GROUP

http://github.com/ctu-mrs/mrs_cheatsheet

MRS lab ROS platform Cheat Sheet

by Tomas Baca @ Multi-robot Systems (MRS), v1.0.0

Ubuntu terminal - GNU/Linux basics

Hitting ⟨Tab⟩ **autocompletes commands, filenames, etc.**

New terminal	Ctrl+Alt+t
Need help	append `--help` after command
Need more help!	:$ man [command]
Change directory	:$ cd [path]
Path symbolic links	. – current directory .. – previous directory ~ – home directory (also $HOME) / – root directory
create a file	:$ touch [path]
remove a file	:$ rm [path]
move (also rename) a file	:$ mv [from] [to]
copy a file	:$ cp [from] [to]
print a file	:$ cat [path]
edit a file	:$ vim [path], :$ nano [path]
set a variable	:$ VARIABLE="dog", VARIABLE=3.0
print a variable	:$ echo "the content is: $VARIABLE"
run a script or executable	:$ ./script.sh, ./program
output redirection	> – to a file (rewrite) >> – to a file (append) | – pipe to another command
redirect to /dev/null	> /dev/null 2>&1

Would You Like to Know More? http://google.com

TMUX - Terminal multiplexer

Run tmux	:$ tmux
List all sessions	:$ tmux ls
Attach to a session	:$ tmux a -t [session name]
New window (tab)	Ctrl+t
New vertical split	Ctrl+s
New horizontal split	Ctrl+d
Moving through windows (tabs)	Shift+→, Shift+←
Moving through panes (splits)	Alt+→, Alt+←, Alt+↑, Alt+↓
prefix	Ctrl+a
Killing window	**prefix** x, :$ exit, :$:q
Killing session	**prefix** k
Detach from session	**prefix** d
Enter vim mode (scrolling, copying)	F2, **prefix** [

Would You Like to Know More? https://github.com/klazalk/linux-setup/wiki/tmux

Vim – a modern pluginizable text processor

Vim is not a joke. Although you might not know how to exit it (yet), it is a very powerful tool. Our vim is filled with features, including code snippets, code completion (ROS aware), code formatting, syntax highlighting and tmux integration. Its control is completely mouse-less and it is fully usable over ssh, which makes it great for remote editing on a drone. Moreover, its modal editing paradigm is very intuitive. Lastly, when you learn how to control vim, you also learn to control other tools such as *Linux manual pages*, *ranger*, *less* and much more. Even gmail uses vim-like controls natively. Run :$ **vimtutor** to start learning vim using an interactive "file tutorial". Here are some simple commands:

switch to insert mode	i	jump a word/Word forwards	w/W
return to normal mode	ESC	jump a word/Word backwards	b/B
cut a line to clipboard	dd	change current word/Word	ciw/ciW
paste a clipboard	p	delete 3 lines down	3dj
open a command line	:	substitute *dog* for *cat*	:%s/dog/cat/g
save	:w	move cursor left/down/up/right	h/j/k/l
quit	:q	delete every line containing *dog*	:%g/dog/normal dd

Would You Like to Know More? https://www.tutorialspoint.com/vim/

Git version control system

Git is a distributed version control system. Repositories are equal, some are just used as a "server" (called **remote**). Git uses branches to isolate ongoing work on the same project. Branches can be merged to combine the work back into a single piece. Changes in the files should be **commited**. Only commit "runnable" code.

Cloning a repository	over ssh :$ git clone git@mrs.felk.cvut.cz:usr/usr_core over https :$ git clone https://github.com/klazalk/linux-setup
Update origin state	:$ git fetch
Update current branch from remote	:$ git pull
Update current branch to remote	:$ git push
Add files for commit	:$ git add [file]
Commit changes	:$ git commit -m "commit message"
Checkout a branch	:$ git checkout [branch name]
Create a branch	:$ git checkout -b [branch name]
unstage the file	:$ git reset [file name]
undo all uncommited changes	:$ git reset --hard
remove all new unstaged files	:$ git clean -fd
Merge a branch	:$ git merge [branch name]
Rebase on a branch	:$ git rebase [branch name]
refactor branch history	:$ git filter-branch [lot of args]
show status	:$ git status
show log	:$ git log
show better log	:$ glog – is an alias for :$ git log with more arguments
show super forest log	:$ flog – uses `./scripts/git-forest.sh`

Would You Like to Know More? https://try.github.io/

.bashrc – Bash configuration

When a new terminal is opened and an instance of bash is launched, the ~/.bashrc file is *sourced* (executed while its leftover variables, functions and aliases stay in the context). We use .bashrc heavily for setting context for ROS and our development environment. .bashrc sources ROS setup scripts, which are also generated by each workspace. If you change this file, source it (or open a new terminal) to activate the changes: :$ source ~/.bashrc or just :$ sb. Here is an example of what should not be missing in the bottom of a healthy .bashrc file:

```
# THIS IS FOR VIM
export CTAGS_SOURCE_DIR="-R "/mrs_workspace -R "/workspace"
export CTAGS_ONCE_SOURCE_DIR="-R "/opt/ros/melodic/include"
export ROS_WORKSPACE="~/mrs_workspace ~/workspace"

export GIT_PATH=$HOME/git

source /opt/ros/melodic/setup.bash
# SOURCE ONLY ONE WORKSPACE AT A TIME
source ~/workspace/devel/setup.bash
# source ~/other_workspace/devel/setup.bash

# NEEDED FOR SIMULATION
source /usr/share/gazebo/setup.sh
source $GIT_PATH/simulation/install/share/simulation/setup.sh
export UAV_MASS=3.0

# SWAP LOCALHOST FOR A HOSTNAME OF A REMOTE ROS SERVER
export ROS_MASTER_URI=http://localhost:11311

# SET TO FALSE IF YOU DON'T LIKE TMUX
export RUN_TMUX=true

source $GIT_PATH/linux-setup/appconfig/bash/dotbashrc
```

Links

MRS UAV System	`github.com/mrs_uav_system`
MRS UAV System wiki	`ctu-mrs.github.io`
MRS cheat sheet	`github.com/ctu-mrs/mrs_cheatsheet`
Summer School Seminar Task #1	`github.com/ctu-mrs/uvdar_leader_follower`
Summer School Seminar Task #2	`github.com/ctu-mrs/mtsp_planning_task`
This presentation	`github.com/ctu-mrs/summer_school_presentation`
Tomas's Linux setup Repository	`github.com/klaxalk/linux-setup`

References

Baca, T and Loianno, G and Saska, M
Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles
2016 IEEE International Conference on Methods and Models in Automation in Robotics

Chudoba, J and Kulish, M and Saska, M and Baca, T and Preucil, L
Exploration and Mapping Technique Suited for Visual-features Based Localization of MAVs
Journal of Intelligent Robotics Systems, 2016

Spurny, V and Baca, T and Saska, M
Complex manoeuvres of heterogeneous MAV-UGV formations using a model predictive control
2016 IEEE International Conference on Methods and Models in Automation in Robotics

Saska, M and Baca, T and Thomas, J and Chudoba, J and Preucil, L and Krajnik, T and Faigl, J and Loianno, G and Kumar, V
System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization
Autonomous Robots, 2016

Baca, T and Hert, D and Loianno, G and Saska, M and Kumar, V
Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles
2018 IEEE/RSJ International Conference on Intelligent Robots and Systems

Baca, T and Petrlik, M and Vrba, M and Spurny, V and Penicka, R and Hert, D and Saska, M
The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles
http://arxiv.org/abs/2008.08050