Introduction to the Unmanned Aerial Platform in the MRS Lab
From control theory to practical experiments

Tomas Baca¹, Robert Penicka¹, Vojtech Spurny¹

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

tomas.baca@fel.cvut.cz
Outline

1 A little bit of history

2 UAV Control pipeline
   - control theory point of view
   - implementation point of view

3 Linux technologies
   - terminal, shell, TMUX, Vim, ROS

4 MRS simulation stack
   - Gazebo/ROS, spawn

5 Summer School seminar
   - the MTSPN problem
   - running the planner, w/ and w/o ROS
   - running the simulation
A little bit of history

≈ 2012, designing custom PCBs for controllers
A little bit of history

≈ 2014, Model Predictive Control on embedded hardware
A little bit of history

≈ 2014, Model Predictive Control on embedded hardware
A little bit of history

Embedded Model Predictive Control

Video: https://youtu.be/AXI_rkQRBaE
A little bit of history

Embedded Model Predictive Control

Video: https://youtu.be/9Bpm4J31CgE
A little bit of history

- Custom and purpose-built hardware
A little bit of history

- Custom and purpose-built hardware
- Embedded software
A little bit of history

- Custom and purpose-built hardware
- Embedded software
- Results:
  - Embedded MPC [Baca et al., 2016]
  - Multi-robot publications: [Spurny et al., 2016], [Saska et al, 2016]
A little bit of history

- Custom and purpose-built hardware
- Embedded software
- Results:
  - Embedded MPC [Baca et al., 2016]
  - Multi-robot publications: [Spurny et al., 2016], [Saska et al, 2016]
- ...
A little bit of history

- Custom and purpose-built hardware
- Embedded software

Results:
- Embedded MPC [Baca et al., 2016]
- Multi-robot publications: [Spurny et al., 2016], [Saska et al, 2016]

... 2015

Not Scalable! Bottlenecks everywhere: Embedded programming sucks, simulations are difficult, system is hard to maintain, connecting sensors and peripheries is limited...
A little bit of history

- Custom and purpose-built hardware
- Embedded software
- Results:
  - Embedded MPC [Baca et al., 2016]
  - Multi-robot publications: [Spurny et al., 2016], [Saska et al, 2016]
- ... 2015
- Not Scalable! Bottlenecks everywhere: Embedded programming *sucks*, simulations are difficult, system is hard to maintain, connecting sensors and peripheries is limited...
A typical pipeline structure revolves around a Linux computer.
Kalman Filter \(\approx 10\) lines in Matlab

\[
\begin{align*}
\hat{x}[t] &\leftarrow A\hat{x}[t-1] + Bu[t-1] \\
\hat{\Sigma}[t] &\leftarrow A\hat{\Sigma}[t-1]A^T + R
\end{align*}
\]

\[
\begin{align*}
K[t] &\leftarrow \hat{\Sigma}[t]W^T\left(W\hat{\Sigma}[t]W^T + M\right)^{-1} \\
\hat{x}[t] &\leftarrow \hat{x}[t] + K[t]\left(z[t] - W\hat{x}[t]\right) \\
\hat{\Sigma}[t] &\leftarrow \left(I - K[t]W\right)\hat{\Sigma}[t]
\end{align*}
\]
Kalman Filter \( \approx 10 \) lines in Matlab

**Prediction phase**

\[
\begin{align*}
\hat{x}_{[t]} & \leftarrow A\hat{x}_{[t-1]} + Bu_{[t-1]} \\
\hat{\Sigma}_{[t]} & \leftarrow A\hat{\Sigma}_{[t-1]}A^T + R
\end{align*}
\]

**Correction phase**

\[
\begin{align*}
K_{[t]} & \leftarrow \hat{\Sigma}_{[t]}W^T\left(W\hat{\Sigma}_{[t]}W^T + M\right)^{-1} \\
\hat{x}_{[t]} & \leftarrow \hat{x}_{[t]} + K_{[t]}(z_{[t]} - W\hat{x}_{[t]}) \\
\hat{\Sigma}_{[t]} & \leftarrow (I - K_{[t]}W)\hat{\Sigma}_{[t]}
\end{align*}
\]

Kalman Filter in practice

- \( \geq 1000 \) lines of code
- Real-world fusion & estimation \( \geq 5000 \) lines of code
- see uav_core/mrs_odometry
From theory to practical experiments

SO(3) controller $\approx$ 10 lines in Matlab

\[
\begin{align*}
\mathbf{m} &= -k_r \mathbf{e}_r - k_\omega \mathbf{e}_\omega + \omega \times \mathbf{j}\omega - \\
&\quad \ldots - \mathbf{j}(\hat{\omega}^t \mathbf{r}_c \omega_c - \mathbf{r}_c \omega_c) \\
\mathbf{f} &= - (-k_x \mathbf{e}_r - k_{ib} \mathbf{r} \int_0^t \mathbf{r}(\tau)^t \mathbf{e}_r d\tau - \\
&\quad \ldots - k_{iw} \int_0^t \mathbf{e}_r d\tau - k_v \mathbf{e}_v - m\mathbf{g} \mathbf{e}_3 + m\ddot{\mathbf{x}}_d) \cdot \mathbf{r} \mathbf{e}_3,
\end{align*}
\]
From theory to practical experiments

**SO(3) controller** $\approx$ 10 lines in Matlab

\[
m = -k_r e_r - k_\omega e_\omega + \omega \times j \omega - \ldots
\]

\[\ldots - j (\dot{\omega} r^t r c \omega c - r^t r c \dot{\omega} c)\]

\[
f = - (-k_x e_r - k_{ib} r \int_0^t r(\tau)^t e_r d\tau - \ldots
\]

\[\ldots - k_{iw} \int_0^t e_r d\tau - k_v e_v - m g e_3 + m \ddot{x}_d) \cdot r e_3,
\]

**MPC tracker** $\approx$ 20 lines in Matlab

\[
\min_{u[t], x[t]} \ V (x, u) = \frac{1}{2} \sum_{i=1}^{m-1} \left( e_{[i]}^T Q_{e[i]} + u_{[i]}^T P_{u[i]} \right)
\]

s.t.  \[
\begin{align*}
x_{[t+1]} &= A x_{[t]} + B u_{[t]}, & \forall t \in \{0, \ldots, m - 1\} \\
x_{[t]} &\leq x_{\text{max}[t]}, & \forall t \in \{1, \ldots, m\} \\
x_{[t]} &\geq x_{\text{min}[t]}, & \forall t \in \{1, \ldots, m\}
\end{align*}
\]
From theory to practical experiments

**SO(3) controller** ≈ 10 lines in Matlab

\[
m = - k_r e_r - k_\omega e_\omega + \omega \times j \omega - \ldots
\]

\[
... - j (\dot{\omega} r^t c \omega_c - r^t r_c \dot{\omega}_c)
\]

\[
f = -( - k_x e_r - k_{ib} \int_0^t r(\tau)^t e_r d\tau - \ldots
\]

\[
... - k_{iw} \int_0^t e_r d\tau - k_v e_v - m g e_3 + m \ddot{x}_d) \cdot r e_3,
\]

**MPC tracker** ≈ 20 lines in Matlab

\[
\min_{u[t], x[t]} \quad V(x, u) = \frac{1}{2} \sum_{i=1}^{m-1} (e_{[i]}^T Q e_{[i]} + u_{[i]}^T P u_{[i]})
\]

s.t. \(x_{[t+1]} = A x_{[t]} + B u_{[t]}\), \quad \forall t \in \{0, \ldots, m - 1\}

\(x_{[t]} \leq x_{\text{max}}_{[t]}\), \quad \forall t \in \{1, \ldots, m\}

\(x_{[t]} \geq x_{\text{min}}_{[t]}\), \quad \forall t \in \{1, \ldots, m\}

**Control in practice**

\(\geq 10,000\) lines of 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 depends 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

```cpp
if (goingToCrash())
    dont();
```
ROS – Robot Operating System

- Middleware allowing communication between programs
- Integrates with Cpp, Python, Bash and Zsh
- Makes the transition from Matlab to reality bearable
- Supported by sensor manufacturers (lots of ROS drivers)
- Integration through the terminal (important, works over ssh)
- Handles the difficult stuff that nobody wants to program: time and clock management, logging, recording onboard data, basic visualization, parameter loading, static and dynamic transformations, etc.
- Integrates to robotic simulators: Gazebo, V-REP

ROS.org
Control – implementation PoV
Control – implementation PoV

- Odometry
  - Rplidar
  - Garmin
  - Mavros interface
    - Mavros
    - Pixhawk
  - Optic flow
    - Auto exposure
      - Bluefox
  - Realsense
  - ORB slam
Control – implementation PoV

UAV manager

Gain manager

SO(3) controller
MPC controller
Failsafe controller
NSF controller
Attitude controller

Control manager

Constraint manager

Mavros
Pixhawk

Landoff Tracker
MPC tracker
Joy Tracker
Matlab Tracker
CSV tracker
Control – implementation PoV

Controllers (ROS Plugins)
- SO(3) controller
- MPC controller
- Failsafe controller
- NSF controller
- Attitude controller

Gain manager

Control manager
- Landoff Tracker
- MPC tracker
- Joy Tracker
- Matlab Tracker
- CSV tracker

Gain manager

UAV manager

Constraint manager

Mavros

Pixhawk
Control – implementation PoV

UAV manager

Gain manager

SO(3) controller

MPC controller

Failsafe controller

NSF controller

Attitude controller

Control manager

Constraint manager

Mavros

Pixhawk

Landoff Tracker

MPC tracker

Joy Tracker

Matlab Tracker

CSV tracker

Trackers (ROS Plugins)
Software tools at our disposal

“Every smith needs a hammer”,

unnamed researcher.
A Linux terminal
Tmux – Terminal multiplexer

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 ~

T31936 0 zsh 1 zsh 2 zsh

http://localhost:11311 14:41 klaxalk-desktop
### Ranger

<table>
<thead>
<tr>
<th>Path</th>
<th>Status</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>install~√ 6 mavros</td>
<td></td>
<td>18 △√</td>
</tr>
<tr>
<td>lib √ 5 mrs_controllers</td>
<td></td>
<td>12 △√</td>
</tr>
<tr>
<td>miscell~√ 4 mrs_general</td>
<td></td>
<td>10 △√</td>
</tr>
<tr>
<td>ros_pac~√ 3 mrs_lib</td>
<td></td>
<td>9 △√</td>
</tr>
<tr>
<td>tmux_sc~√ 2 mrs_mavros_interface</td>
<td></td>
<td>10 △√</td>
</tr>
<tr>
<td>READ~.md~√ 1 mrs_msgs</td>
<td></td>
<td>12 △√</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>klaxalk@klaxalk-xps /home/klaxalk/git/uav_core/ros_packages/mrs_odometry</td>
<td>config</td>
<td></td>
</tr>
<tr>
<td>install~√ 6 mrs_odometry</td>
<td></td>
<td>15 △√</td>
</tr>
<tr>
<td>1 mrs_optic_flow</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>2 mrs_status</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>3 mrs_trackers</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4 mrs_uav_manager</td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

**Permissions:**

- `drwxrwxr-x` 11 klaxalk klaxalk -> ../.gitman/mrs_odometry

**System Information:**

- 0 sum, 26.1G free 7/11 All
- T27315 0 zsh
Vim – modal, modular, modern

Vim in (Tmux in (Linux Terminal))

```
if (!is_initialized)
    return;

mrs_lib::Routine profiler_routine = profiler->createRoutine("callbackMavrosState");
{
    std::scoped_lock lock(mutex_mavros_state);
    mavros_state = *msg;
}
```

```
operator=( State_<allocator<void> > & & ) f State_<allocator<void> > & &
```

```
INSERT COMPL  769c94d  callbackMavrosState()  cpp  utf-8[unix]  59%  768/1293  []  16
```
Terminals and command-line environments are crucial for developers and system administrators. Vim, with its powerful functionality and customization options, is a popular choice among programmers. When integrated with tools like Tmux in a Linux terminal, it offers enhanced productivity and efficiency.

```cpp
#include <ros/ros.h>

int main(int argc, char **argv)
{
    ros::init(argc, argv, "my_node");
    ros::NodeHandle n;
    ros::Rate rate(10.0);
    // Code here...
    return 0;
}
```

This snippet shows a simple ROS node written in C++, illustrating how Vim can be used for coding and editing such scripts. The terminal layout includes Vim, Tmux, and a ROS node, demonstrating the workflow of a developer working on a ROS project.
Gazebo/ROS simulator
Spawning drones to the simulator

```
klaxalk@klaxalk-desktop ~$ spawn --help

usage: Spawn vehicles for team. [-h] [--f450]
    [--gazebo-ros-master-uri GAZEBO_ROS_MASTER_URI]
    [--maylink-address MAYLINK_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-scense] [--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]
    [-u-v-camera-calibration-file UV_CAMERA_CALIBRATION_FILE]
    [--debug]
    [VEHICLE_ID [VEHICLE_ID ...]]
```
Now is the time to show the Live demo.

1st law of robotics: never give a robotic demo!
Seminars’ assignment – MTSPN

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 m/s
- max. acceleration 2.5 m/s²
- 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
Home

Last edited by Matěj Petrlik 2 weeks ago

First steps (Beginner user)

We encourage everyone one to familiarize with the following topics since it will make your start with UAVs much easier. If you find some information missing, ask us personally or email us to tomas.baca@fel.cvut.cz or vojtech.spurny@fel.cvut.cz.

Suggested reading for newcomers

Are you new here? Start with Suggested reading for newcomers. It will help you with further work.

File structure

After you had installed everything by our automated install script, it is time to learn about what appeared in your file system. Learn about the file structure of our development environment.

How to compile?

Do you have everything downloaded and cloned? The next step is to learn, how to compile all our workspaces. Follow to how to compile page.

How to start the simulation

Is everything compiled and ready to run? How to start the simulation page will help you with the next step.
Links

- MRS wiki –
  https://mrs.felk.cvut.cz/gitlab/uav/uav_core/wikis/home

- MRS cheat sheet –
  https://github.com/klaxalk/mrs-cheatsheet

- Summer School Seminar Task –
  https://mrs.felk.cvut.cz/gitlab/summer-school/seminar_task

- This presentation –
  https://github.com/klaxalk/mrs-presentation

- Tomas’s Linux setup Repository –
  https://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

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

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