Controls for Multi-Rotor Vehicles
... from Model-Based to Learning-Enabled Approaches

Prof. Angela Schoellig, University of Toronto

TRADR Summer School on Autonomous Micro Aerial Vehicles
August 25, 2015
What is Controls?
What is Controls?

It’s MAGIC!
Armageddon
@ the Flying Machine Arena

April 2011
What is Controls?

Controls enables a machine to achieve a task without human interaction. Despite disturbances.

⇒ Self-regulating system.
How is this relevant for flying robots?
Motor Turn Rates

Motor Controller

Motion

Measured Turn Rates
QUADROTOR CONTROL

Body Turn Rate
+ Thrust

Motor Turn Rates

Gyroscopes
(+Accelerometer)

Rate Ctrl

Motor Ctrl

Measured Turn Rates

Motion
QUADROTOR CONTROL

Angles + Thrust

Body Turn Rate + Thrust

Motor Turn Rates

Motion

Angle Ctrl

Rate Ctrl

Motor Ctrl

Angle Measurements/Estimates

Gyroscopes (+Accelerometer)

Measured Turn Rates
Allows us to focus on the high-level task.
How does it fit together?
OVERVIEW

Aerial Manipulation

Motion Planning

Controls

Simultaneous Localization & Mapping

Nonlinear State Estimation

3D Reconstruction

Sensors

Vision

Perception

Action
OVERVIEW

Motion Planning

Controls

ACTION

Simultaneous Localization & Mapping

Nonlinear State Estimation

PERCEPTION

3D Reconstruction

Sensors

Vision

OVERVIEW
OVERVIEW

Motion Planning → Controls

PERCEPTION

GPS

ACTION
My goal for today!
GOAL

Prepare you to design your own advanced controllers.
OUTLINE

I. Model-Based Control
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. Learning-Enabled Control
   - Task-Dependent Learning
   - Task-Independent and Safe Learning

III. Summary
GOAL OF CONTROLS:
Want the error to go exponentially to zero as function of time.

\[ e := x_d - x \]

Example:
\[ \dot{e} + ke = 0 \implies e(t) = e_0 \exp(-kt), \quad k > 0. \]
GOAL OF CONTROLS:
Want the error to go exponentially to zero as function of time.

\[ e := x_d - x \]

Example:
\[ \dot{e} + ke = 0 \implies e(t) = e_o \exp(-kt), \quad k > 0. \]

Can be higher-order, but coefficients must be non-negative.
Example:

\[ \ddot{x} = u, \quad u := k_p (x_d - x) + k_v (\dot{x}_d - \dot{x}) + \ddot{x}_d \]

\[ \Rightarrow \quad \ddot{e} + k_v \dot{e} + k_p e = 0 \]

\[ \Rightarrow \quad \ddot{e} + 2\xi \omega_n \dot{e} + \omega_n^2 e = 0 \]

Intuitive parameterization:
- Damping ratio: \( \xi \in [0.7..1] \)
- Natural frequency, related to rise time (10-90%): \( \omega_n \sim \frac{t_r}{1.8} \)
MODEL-FREE VS. MODEL-BASED CONTROL

Plant: \[ m\ddot{x} + b\dot{x} + kx = u \]

Model-free: \[ u = k_pe + k_v\dot{e} + \int e\,dt + \ddot{x}_d \]

Advantages? Disadvantages?

• No model needed.
• Performance depends on model parameters.
• Need to tune gains to maximize performance.
MODEL-FREE VS. MODEL-BASED CONTROL

Plant: \[ m\ddot{x} + b\dot{x} + kx = u \]

Model-based:

\[ u = m(\ddot{x}_d + k_p e + k_v \dot{e}) + b\dot{x} + kx \]

\[ \Rightarrow \dot{e} + 2\xi \omega_n \dot{e} + \omega_n^2 e = 0 \]

Advantages? Disadvantages?

• Model needed. Model parameter errors?
• Model-based part: cancels dynamics of the system.
• Model-independent part: design/tune independent of the model.
MODEL-FREE VS. MODEL-BASED CONTROL

Tracking error bounded...

Perfect model

Imperfect model, 10% errors in parameters
SUMMARY

Model-free:

• No model needed.
• Performance depends on model parameters. Re-tune often...
• Need to tune gains to maximize performance.

Advantages? Disadvantages?

• Model needed. Model errors?
• Model-based part: cancels dynamics of the system.
• Model-independent part: design/tune independent of the model.
I. Model-Based Control
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. Learning-Enabled Control
   - Task-Dependent Learning
   - Task-Independent and Safe Learning

III. Summary
\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z}
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
-g
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
c
\end{bmatrix}
\]
Maps from body to inertial frame.

\[c = \frac{(f_1 + f_2 + f_3 + f_4)}{m}\]
MODEL

\[
J \begin{bmatrix}
\dot{p} \\
\dot{q} \\
\dot{r}
\end{bmatrix} =
\begin{bmatrix}
l(f_2 - f_4) \\
l(f_3 - f_1) \\
\kappa(f_1 - f_2 + f_3 - f_4)
\end{bmatrix} - \begin{bmatrix}
p \\
q \\
r
\end{bmatrix} \times J \begin{bmatrix}
p \\
q \\
r
\end{bmatrix}
\]

\[
\dot{R} = R \begin{bmatrix}
0 & -r & q \\
r & 0 & -p \\
-q & p & 0
\end{bmatrix}
\]
I. **Model-Based Control**
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. **Learning-Enabled Control**
   - Task-Dependent Learning
   - Task-Independent and Safe Learning

III. **Summary**
PART 1: VERTICAL CONTROL

\[
\begin{bmatrix}
\ddot{x} \\
\ddot{y} \\
\ddot{z}
\end{bmatrix} = 
\begin{bmatrix}
0 \\
0 \\
-g
\end{bmatrix} + \mathbf{R} 
\begin{bmatrix}
0 \\
0 \\
c
\end{bmatrix}
\]

\[c = (f_1 + f_2 + f_3 + f_4)/m\]

\[
c_d = \frac{1}{R_{33}} \left( \omega^2_{n,z} (z_d - z) + 2\xi_z \omega_{n,z} (\dot{z}_d - \dot{z}) + \ddot{z}_d + g \right)
\]
PART 1: LATERAL CONTROL

Define similarly for lateral direction:
\[ a_x = \omega^2_{n,x} (x_d - x) + 2\xi_x \omega_{n,x} (\dot{x}_d - \dot{x}) + \ddot{x}_d \]
\[ a_y = \ldots \]

Transform into desired turn rates:
\[
\begin{bmatrix}
  a_x \\
  a_y
\end{bmatrix} =
\begin{bmatrix}
  R_{13,d} \\
  R_{23,d}
\end{bmatrix} C_d
\]
\[
\dot{R}_{13,d} = \frac{1}{\tau_{13}} (R_{13,d} - R_{13})
\]
\[
\dot{R}_{23,d} = \frac{1}{\tau_{23}} (R_{23,d} - R_{23})
\]

\[
\begin{bmatrix}
  p_d \\
  q_d
\end{bmatrix} = \frac{1}{R_{33}} \begin{bmatrix}
  R_{21} & -R_{11} \\
  R_{22} & -R_{12}
\end{bmatrix} \begin{bmatrix}
  \dot{R}_{13,d} \\
  \dot{R}_{23,d}
\end{bmatrix}
\]
OVERALL CONTROL

OFF-BOARD

Horizontal Controller $\alpha_x$

Reduced Attitude Controller $\alpha_y$

Yaw Controller

$\alpha_x, \alpha_y$

Vertical Controller $C_d$

$P_d, Q_d, r_d$

Body Rate Controller $f_{1..4,d}$

Motor Controller

ON-BOARD

Body rate controller:

$$J \begin{bmatrix} \frac{1}{\tau_p} (p_d - p) \\ \frac{1}{\tau_q} (q_d - q) \\ \frac{1}{\tau_r} (r_d - r) \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} l(f_{2,d} - f_{4,d}) \\ l(f_{3,d} - f_{1,d}) \\ \kappa(f_{1,d} - f_{2,d} + f_{3,d} - f_{4,d}) \end{bmatrix}$$

$$\left(f_{1,d} + f_{2,d} + f_{3,d} + f_{4,d}\right) = mc_d$$
OUTLINE

I. Model-Based Control
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. Learning-Enabled Control
   - Task-Dependent Learning
   - Task-Independent and Safe Learning

III. Summary
OTHER APPROACHES

• **Linear controller based on linearized system** [Gurdan, et al. 2007][Boubadallah, 2007][Hoffman et al., 2008]

• **LQR**

• **Backstepping**

• **Exact linearization**, differentially flat system [Mellinger]

• **L1 adaptive control**
OUTLINE

I. Model-Based Control
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. Learning-Enabled Control
    - Task-Dependent Learning
    - Task-Independent and Safe Learning

III. Summary
LIMITATIONS

Latency

- 13.19 ms command rate
- 5 ms motion capture
- 1.25 ms onboard loop

Latency (ms)
LIMITATIONS

Latency

Predict the vehicle position at the time the input arrives at the vehicle.

![Graph showing pitch angle over time with measured and predicted lines, indicating a latency of 30 ms.](image)
Latency

Circle motion at 4 m/s.
Offsets
Calibrate during hover.
LIMITATIONS

Aggressive Maneuvers

1. Triple flip with a quadrotor.

2. Time-optimized slalom.

3. Fast path following with a ground vehicle.
MOTIVATING EXAMPLE

Without ILC
1.0 m/s
EXPLANATION

- Unmodelled dynamics
- Unknown external disturbances (e.g., environment conditions such as surface material, topography or weather)

Model inaccuracies limit achievable performance!
Learning/adaptation enables safe, high-performance motions in uncontrolled, unknown or changing environments.
RESEARCH FOCUS

Prior information
- Which motions are feasible?
- How to plan collision-free motions?

+ Current sensor measurement
- How to guide the vehicle along a desired path?

+ Past experiment data
- Can the performance be improved by leveraging past data?

Towards robotics applications.
FRAMEWORK

Update the input and/or controller

LEARNING

Improve the controls performance by learning from data.
I. Model-Based Control
   - Model-Free Vs. Model-Based Control
   - Quadrotor Model
   - Position Control Approach
   - Other Approaches
   - What Can Go Wrong?

II. Learning-Enabled Control
   - Task-Dependent Learning
   - Task-Independent and Safe Learning

III. Summary
1. TASK-DEPENDENT LEARNING

**Task** Executing a *given* motion.

**Data Incorporation** Adaptation of input parameters.

1. TRIPLE FLIPS

A Priori Knowledge  First-principles model, input constraints, parameterized input trajectory
1 | APPROACH

A Priori Knowledge  First-principles model, input constraints, parameterized input trajectory

![Diagram showing the approach](image)
Algorithm Policy gradient method

1. First principles model
2. Initial parameter set $P^0$
3. Correction matrix $J^{-1}$
4. Perform flip
5. Final error $E^i$
6. Correction $P^{i+1} = P^i - \gamma J^{-1}E^i$
APPROACH

Final State Error

Collective Acceleration Parameters

Stage Duration Parameters

Iteration

(m,m/s,rad)

2

0

-2

-4

Final State Error

 Collective Acceleration Parameters

 Stage Duration Parameters

Iteration

(g)

2.25

2.2

2.15

2.1

2.05

2.0

2.3

2.25

2.2

2.05

2.0

 Stage Duration Parameters

Iteration

(s)

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1.0

1.1

1.5

model

Iter 1

Iter 70
3 | TASK-DEPENDENT LEARNING

**Task** Executing a *given* motion.

**Data Incorporation** Adaptation of full discretized input trajectory.

### 2. FINITE-TIME TRAJECTORY

![Slalom racing example.](image)

**INPUT** (Desired position) → **CONTROL** → **OUTPUT** (Measured position)

![Graph showing desired and actual y-position over x-position.](graph)
3 | APPROACH

1. Extract system model from numeric simulation
2. Estimate model error/systematic offset along trajectory
3. Update input trajectory
**APPROACH**

**A Priori Knowledge** First-principles model, input and state constraints, desired output trajectory

Prerequisites:

- Coarse model \( \mathcal{D} : U \rightarrow (Y, C) \)
- Desired output trajectory \( (Y^*, C^*) = \mathcal{D}(U^*) \)

Linear mapping from numeric simulation of coarse model:

\[
\begin{align*}
y &= Fu, \\
c &= Lu \\
u &= U - U^*, \\
y &= Y - Y^*, \\
c &= C - C^*
\end{align*}
\]
Algorithm Optimization-based Iterative Learning

Iteration-Domain Model:
- For each trial $j, j \in \{1, 2, \ldots\}$,

$$y_j = F u_j + d_j + \mu_j$$

$$d_{j+1} = d_j + \omega_j$$

Disturbance estimate: Kalman filter in the iteration domain
- From $j$th execution get $y_j$ and estimate $\hat{d}_{j+1}$

Input update: minimize expected tracking error

$$\min_{u_{j+1}} \left\| F u_{j+1} + \hat{d}_{j+1} \right\|_p$$

subject to

$$L u_{j+1} \preceq c_{max}$$

$d_j$ – unknown recurring disturbance

$\mu_j, \omega_j$ – trial-uncorrelated, zero-mean Gaussian noise

Convex optimization
3 | RESULT

![Graph showing the results of iterations 1, 2, 3, 10-15 with labels for each iteration and the desired path.](image-url)

- Iteration 1
- Iteration 2
- Iteration 3
- Desired Path

- Iteration 10-15

**x-position [m]**

**y-position [m]**
TASK-DEPENDENT LEARNING

**Task** Executing a *given* motion.

**Data Incorporation** Adaptation of input parameters.

If task changes, learning is started from scratch!
I. Model-Based Control

- Model-Free Vs. Model-Based Control
- Quadrotor Model
- Position Control Approach
- Other Approaches
- What Can Go Wrong?

II. Learning-Enabled Control

- Task-Dependent Learning
- Task-Independent and Safe Learning

III. Summary
Task-Independent Learning

**Task** Executing a *set of* motions.

**Procedure** *Continuous* operation.

**Data Incorporation** Adaptation of *system model* and *feedback controller*.

Learning-based Model Predictive Control


Prof. Tim Barfoot

Chris Ostafew
State-space model with state- and input-dependent disturbance model:

\[
x_{k+1} = f(x_k, u_k) + g(a_k)
\]

\[
a_k = (x_k, v_{k-1}, u_k, u_{k-1})
\]

Using a Gaussian Process to estimate the disturbance function.
LEARNING-BASED MODEL PREDICTIVE CONTROL

GP-based Disturbance Model

Nonlinear Model Predictive Control

Mobile Robot

$x_d$  $u$  $x$
Teach Pass
TASK-INDEPENDENT LEARNING

- Disturbance modelled as function of state and input using a Gaussian Process.
- Learning data can be transferred from one task to another.
- Uncertainty estimate is not considered, safety during learning not guaranteed.
SAFE, TASK-INDEPENDENT LEARNING

**Procedure**  *Continuous* operation.

**Data Incorporation**  Adaptation of *system model* and *feedback controller*.

**ROBUST LEARNING CONTROL**

- Guarantee stability while improving performance [1]


Robust control

• Specify prior uncertainty in model
• Guarantee stability and performance for all possible models

Online learning

• Learn from online data
• Improve the model
THE MISSING LINK

<table>
<thead>
<tr>
<th></th>
<th>Robust Control</th>
<th>Online learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models uncertainty</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Guarantees stability</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Improves online</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Nominal model**
- **True model**
- **Set of possible models**

Online learning

- **Set of possible models**
- **True model**
- **Nominal model**
**APPROACH**

\[
x_{k+1} = f(x_k, u_k) + g(x_k, u_k) \\
\text{a priori model} \quad \text{to be learned}
\]

\[
y_k = Cx_k + \omega_k,
\]

- **Gaussian Process:** Online learning
- **Robust Control:** Guaranteed stability / performance

Nominal model
\[\approx f(x_k, u_k) + \mathbb{E}[g(x_k, u_k)]\]

True model
\[\bullet\]

Set of possible models \[\approx \text{Var}[g(x_k, u_k)]\]
SAFE, TASK-INDEPENDENT LEARNING

- **Combined** Gaussian Process learning with Linear Robust Control
- Enables **controller performance to improve online** while providing **stability guarantees**
DEVELOPMENT

Specific task, adaptation of a few input parameters only

General task, full input trajectory adaptation

Model learning, anytime learning.

Learning with safety guarantees.

True model
- Nominal model

Institute for Aerospace Studies
UNIVERSITY OF TORONTO

Angela Schoellig
SAFE, TASK-INDEPENDENT LEARNING

... more to come!
MY GROUP

#1 university in Canada, top 20 worldwide.
Founded in 1827.
Interesting outdoor flight opportunities, official flight licenses easy to get!
LEARNING HELPS US TO ACHIEVE...

- High speed
- High accuracy
- Energy efficiency
- Excellence
- Safe for the human
- ... the robot
- ... the environment

Excellence

Safety
THANK YOU

For follow-up discussions, please contact me:

Angela P. Schoellig
web:  www.schoellig.name
email:  schoellig@utias.utoronto.ca

FOLLOW US!
EXERCISE

You get the task to fly the Parrot AR.Drone autonomously in an indoor motion capture system.

• **Measurements:**
  Full vehicle state

• **Inputs to be computed:**
  Roll, pitch (ZYX Euler angles), rate around body z-axis, z velocity

⇒ Start with `quadsim_user_interface.m`
⇒ Fill out `DSLcontroller.m`, `desiredstate.m`, `parameters.m`
⇒ Do not change given parameters.
CHALLENGE

Fly a circle of 4m/s and 1m radius (e.g. sin(4t) ).

- Calculate your tracking error

⇒ Start with `quadsim_user_interface.m`
⇒ Fill out `DSLcontroller.m, desiredstate.m, parameters.m`
⇒ Do not change given parameters.