Differential Flatness

Project 1-2 Debrief

Student solutions pushed the staff out of the top-10 (about 40 seconds).

Fastest on-time submission about 24 seconds, no one faster than 14 seconds.

If you got down to one minute, you probably made all the right “big” choices.

Optimization can go two ways…

1. More beautiful with each step.

"Perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away.“

-- Antoine de Saint Exupéry, Aviator and Author

2. More inscrutable with each step.

“Programs must be written for people to read, and only incidentally for machines to execute.”

-- Abelson and Sussman, "Structure and Interpretation of Computer Programs"

"Premature optimization is the root of all evil.“

-- Donald Knuth

(context: Knuth wasn’t excusing sloth; he was writing about *how to build fast programs.)*

Lessons to Carry Forward

Code with good comments is much more likely to be correct.

- When it is not correct, it is much easier to find the error.

  (Just look for a place where the code doesn't do what the comment says it does.)
- Yes, I really do write the comments before I write the code.

Ways to make your code "self-commenting."

Good variable names.

- Correct terminology.
- Verbosity in proportion to scope.

Write (sub)-functions.

- Clearly identify inputs and outputs of sections of code.
- Independently testable.

Lab Experiments

Differences between simulation and experiment?

Share your data now, before someone drops their laptop in a swimming pool.

We are guests in the lab space. It can be a cluttered mess, but the robots and equipment are precious to someone.

Today's Topic:

How do we extend our results for points

$$x^{(n)} = u$$

linear

to robots that look like

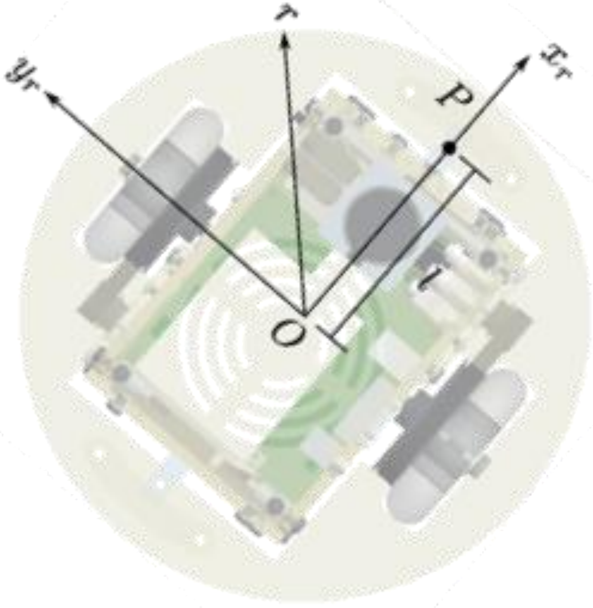
$$\dot{x} = f(x) + g(x)u \text{ ?}$$

nonlinear, control affine

maybe underactuated

Aside: why might this be difficult?

Example: Kinematic Planar Cart



Equations of Motion

Equations of Motion and Inputs

$$\dot{X} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad \dot{X} = g(X)u$$

Define an "output" of interest.

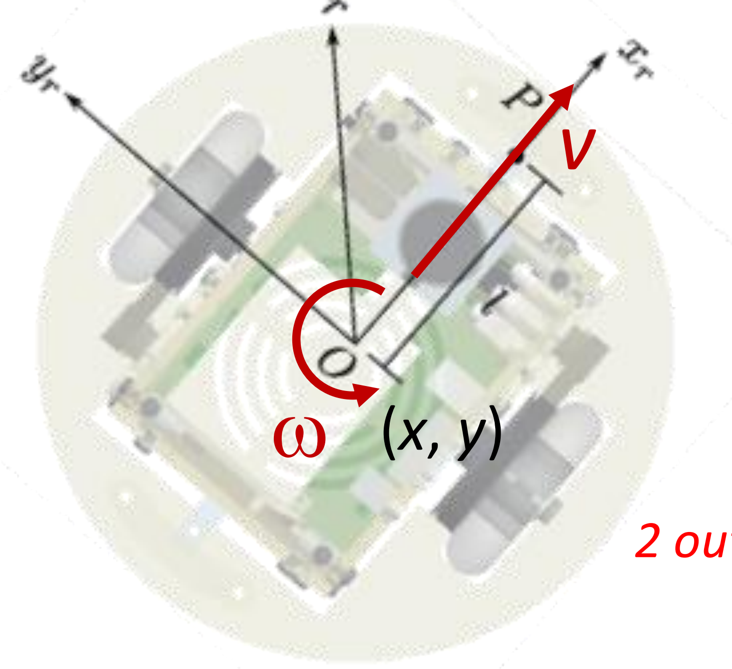
$$y = h(x) = \begin{bmatrix} x \\ y \end{bmatrix}$$

How does the output change?

$$\dot{y} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$\mathcal{L}_g h$ is singular

2 inputs, 3 degrees of freedom



2 outputs (x,y)

Is it possible to control the two outputs with two inputs?

Equations of Motion

We can't control $\dot{y}$ directly using those inputs.

What does the next derivative look like?

$$\dot{X} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$y = h(x) = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\dot{y} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

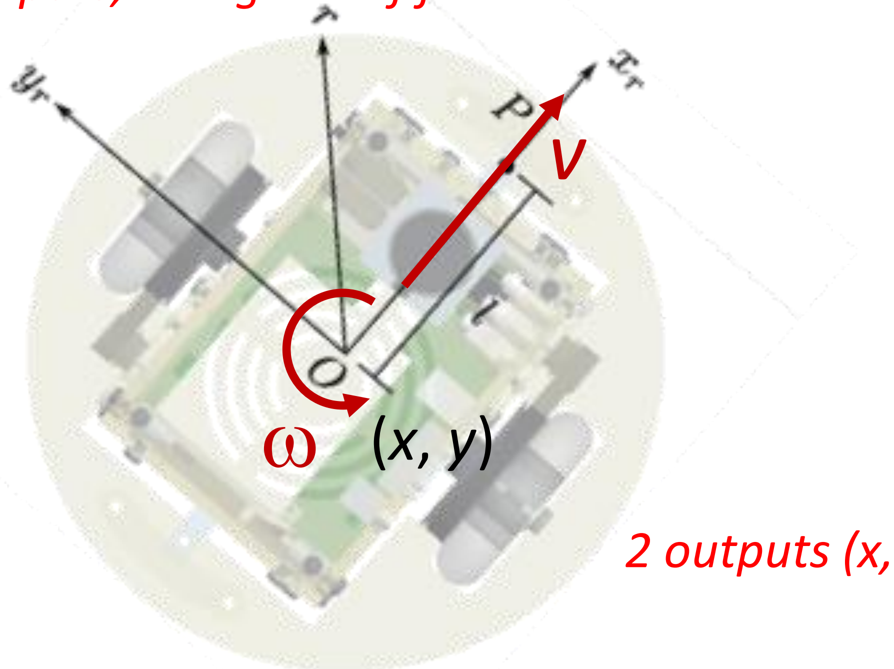
$$\ddot{y} = \begin{bmatrix} -\sin\theta \; v\omega + \cos\theta \; \dot{v} \\ \cos\theta \; v\omega + \sin\theta \; \dot{v} \end{bmatrix}$$

Pick new inputs $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix}$

New state equation

$$X = \begin{bmatrix} x \\ y \\ v \\ \theta \end{bmatrix} \quad \dot{X} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ u_1 \\ u_2 \end{bmatrix}$$

2 inputs, 3 degrees of freedom



2 outputs (x,y)

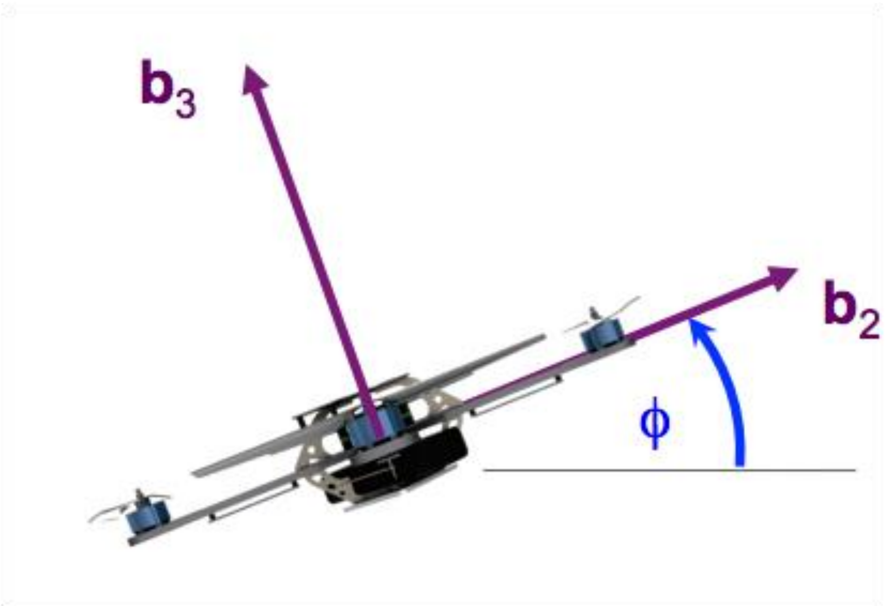
Output as a function of new inputs

$$\ddot{y} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix}$$

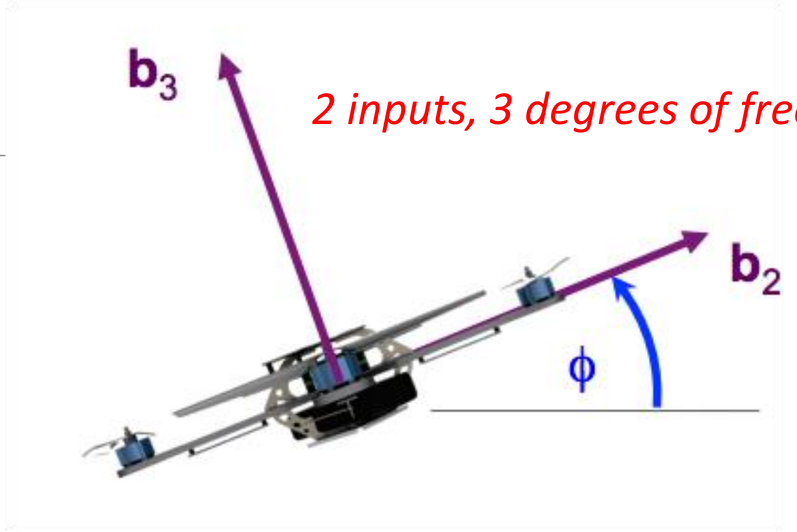
$\mathcal{L}_g\mathcal{L}_f h$ is nonsingular

except when …?

Example: Planar Quadrotor



Equation of Motion



governing equations:

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

state:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix}$$

inputs:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

state space form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}$$

also "control affine"

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Controlling Position as an "Output"

How can we control the quadrotor position?

Choose the position as an "output" $\boldsymbol{y}$ for analysis.

Does the input $\boldsymbol{u}$ influence the output $\boldsymbol{y}$?

$$\boldsymbol{y} = h(\boldsymbol{x}) = \begin{bmatrix} y \\ z \end{bmatrix}$$

No.

Does the input $\boldsymbol{u}$ influence $\dot{\boldsymbol{y}}$?

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{y} \\ \dot{z} \end{bmatrix}$$

No.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Controlling Position as an "Output"

Does the input $\boldsymbol{u}$ influence $\ddot{\boldsymbol{y}}$?

$$\ddot{\boldsymbol{y}} = \begin{bmatrix} \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} u_1 \sin\phi \\ -g + \frac{1}{m} u_1 \cos\phi \end{bmatrix}$$

Input u_1 appears, but not u_2.
We can't generate arbitrary $\ddot{\boldsymbol{y}}$.

Does the input $\boldsymbol{u}$ influence $\dddot{\boldsymbol{y}}$?

$$\dddot{\boldsymbol{y}} = \begin{bmatrix} \dddot{y} \\ \dddot{z} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-\dot{u}_1 \sin\phi - \dot{\phi} u_1 \cos\phi) \\ \frac{1}{m}(\dot{u}_1 \cos\phi - \dot{\phi} u_1 \sin\phi) \end{bmatrix}$$

Same result.
We can't generate arbitrary $\dddot{\boldsymbol{y}}$.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Does the input $\boldsymbol{u}$ influence $\boldsymbol{y}^{(iv)}$?

$$\boldsymbol{y}^{(\mathrm{iv})} = \begin{bmatrix} y^{(\mathrm{iv})} \\ z^{(\mathrm{iv})} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-\ddot{u}_1 \sin\phi - 2\dot{\phi}\dot{u}_1 \cos\phi + \dot{\phi}^2 u_1 \sin\phi - \ddot{\phi} u_1 \cos\phi) \\ \frac{1}{m}(\ddot{u}_1 \cos\phi - 2\dot{\phi}\dot{u}_1 \sin\phi - \dot{\phi}^2 u_1 \cos\phi - \ddot{\phi} u_1 \sin\phi) \end{bmatrix}$$

Needs a closer look.

Controlling Position as an "Output"

From the dynamics, substitute $\ddot{\phi} = \frac{1}{I_{xx}} u_2$

finally u_2!

$$\boldsymbol{y}^{(\mathrm{iv})} = \begin{bmatrix} y^{(\mathrm{iv})} \\ z^{(\mathrm{iv})} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-\ddot{u}_1 \sin\phi - 2\dot{\phi}\dot{u}_1 \cos\phi + \dot{\phi}^2 u_1 \sin\phi - \frac{1}{I_{xx}} u_1 u_2 \cos\phi) \\ \frac{1}{m}(\ddot{u}_1 \cos\phi - 2\dot{\phi}\dot{u}_1 \sin\phi - \dot{\phi}^2 u_1 \cos\phi - \frac{1}{I_{xx}} u_1 u_2 \sin\phi) \end{bmatrix}$$

… and now have derivatives of u_1

Since $\ddot{u}_1$ drives $\dot{u}_1$ and u_1, we can think of $\{\ddot{u}_1, u_2\}$ as new inputs and form an "extended state."

new input

$$\bar{\boldsymbol{u}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix} = \begin{bmatrix} \ddot{u}_1 \\ u_2 \end{bmatrix}$$

new extended state

$$\bar{\boldsymbol{x}} = \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ u_1 \\ \dot{u}_1 \end{bmatrix}$$

lower derivatives of our new input appear in the extended state

Extended State Equation

Now re-write the control affine state space system for the new state.

new input

$$\bar{\boldsymbol{u}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix} = \begin{bmatrix} \ddot{u}_1 \\ u_2 \end{bmatrix}$$

new state

$$\bar{\boldsymbol{x}} = \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ u_1 \\ \dot{u}_1 \end{bmatrix}$$

original eqn.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

extended state space system

$$\dot{\bar{\boldsymbol{x}}} = \bar{f}(\bar{\boldsymbol{x}}) + \bar{g}(\bar{\boldsymbol{x}})\bar{\boldsymbol{u}}$$

$$\dot{\bar{\boldsymbol{x}}} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \frac{-1}{m} u_1 \sin\phi \\ -g + \frac{1}{m} u_1 \cos\phi \\ 0 \\ \dot{u}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{I_{xx}} \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix}$$

The Relative Degree is 4

Using our new variables, we can pull the input terms out of the snap equation from before.

$$\underbrace{\boxed{\begin{bmatrix} y^{(\mathrm{iv})} \\ z^{(\mathrm{iv})} \end{bmatrix}}}_{h^{(\mathrm{iv})}} = \frac{1}{m}\begin{bmatrix} -2\dot{\phi}\dot{u}_1 \cos\phi + \dot{\phi}^2 u_1 \sin\phi \\ -2\dot{\phi}\dot{u}_1 \sin\phi - \dot{\phi}^2 u_1 \cos\phi \end{bmatrix} + \underbrace{\boxed{\frac{1}{m}\begin{bmatrix} -\sin\phi & -\frac{1}{I_{xx}} u_1 \cos\phi \\ \cos\phi & -\frac{1}{I_{xx}} u_1 \sin\phi \end{bmatrix}}}_{\mathcal{L}_{\bar{g}}\mathcal{L}_{\bar{f}}^3 h} \underbrace{\boxed{\begin{bmatrix} \ddot{u}_1 \\ u_2 \end{bmatrix}}}_{\bar{u}}$$

Under what conditions is $\mathcal{L}_{\bar{g}}\mathcal{L}_{\bar{f}}^3 h$ full rank? What is the consequence?

Differential Flatness

A Differentially Flat System

A system with n states and m inputs

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$$

$\boldsymbol{x} \in \mathbb{R}^n$, states (possibly coordinates for some interesting manifold)

$\boldsymbol{u} \in \mathbb{R}^m$, inputs

is *differentially flat* if there exists a *flat output* $\boldsymbol{y} \in \mathbb{R}^m$ such that:

- The number of flat outputs equals the number of inputs.
- The flat outputs and their derivatives are independent.
- The outputs $\boldsymbol{y}$ can be written as a smooth function of $\boldsymbol{x}$,$\boldsymbol{u}$ and derivatives.
- Conversely $\boldsymbol{x}$,$\boldsymbol{u}$ can be written in terms of the output $\boldsymbol{y}$ and derivatives.

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \dots, \boldsymbol{u}^{(p)})$$

$$\boldsymbol{x} = \varphi(\boldsymbol{y}, \dot{\boldsymbol{y}}, \dots, \boldsymbol{y}^{(q)})$$

$$\boldsymbol{u} = \psi(\boldsymbol{y}, \dot{\boldsymbol{y}}, \dots, \boldsymbol{y}^{(q)})$$

Example: Planar Quadrotor

The planar quadrotor is a differentially flat system

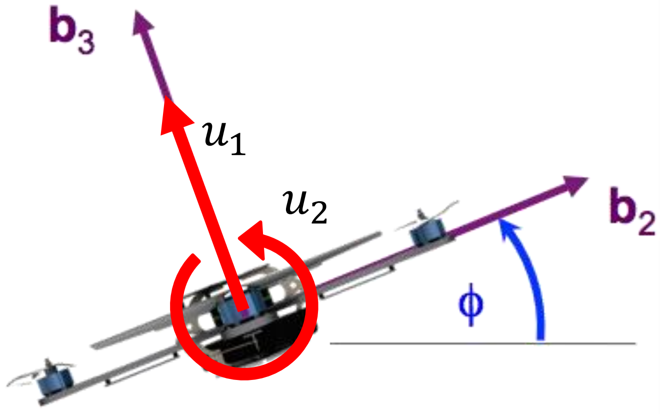
- All state variables and the inputs can be written as smooth functions of *flat outputs* and their derivatives

flat outputs: y, z

$$\begin{bmatrix} y \\ z \\ \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \\ \dddot{y} \\ \dddot{z} \\ y^{(iv)} \\ z^{(iv)} \end{bmatrix} \xleftrightarrow{\text{diffeomorphism}} \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \hdashline u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ u_2 \end{bmatrix}$$

state: $y, z, \phi, \dot{y}, \dot{z}, \dot{\phi}$

inputs (and derivatives): $u_1, \dot{u}_1, \ddot{u}_1, u_2$



Useful For: Trajectory Planning

<u>Every</u> curve $t \to \boldsymbol{y}(t)$ smooth enough corresponds to a valid trajectory.

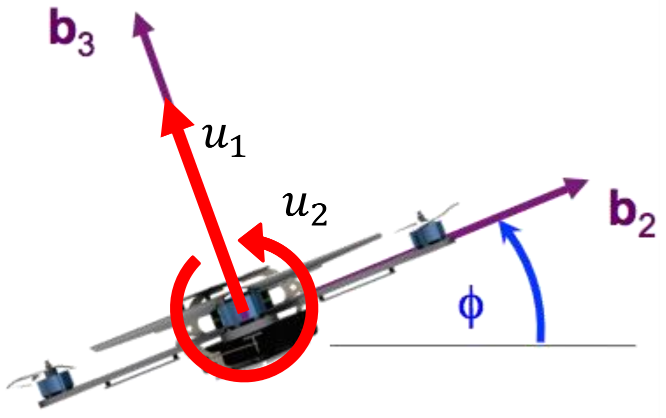
Planning in the flat outputs is easy.

- Draw a smooth enough curve in $\mathbb{R}^n$.

Planning in state space is hard.

- Trajectories must obey ODE.
- Maybe on some manifold not $\mathbb{R}^n$?
- Maybe nonholonomic constraints?
- Maybe underactuated.

$$\begin{bmatrix} \left.\begin{matrix} y \\ z \end{matrix}\right\}\text{flat outputs} \\ \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \\ \dddot{y} \\ \dddot{z} \\ y^{(iv)} \\ z^{(iv)} \end{bmatrix} \overset{\text{diffeomorphism}}{\longleftrightarrow} \begin{bmatrix} \left.\begin{matrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{matrix}\right\}\text{state} \\ \hdashline \left.\begin{matrix} u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ u_2 \end{matrix}\right\}\text{inputs (and derivatives)} \end{bmatrix}$$



3-D Quadrotor

Inputs

$u_1, \mathbf{u}_2$

State

$(\mathbf{x}, \dot{\mathbf{x}})$

$$u_1 = \sum_{i=1}^{4} F_i$$

$$\mathbf{u}_2 = L \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ \mu & -\mu & \mu & -\mu \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

flat outputs $\mathbf{y}$

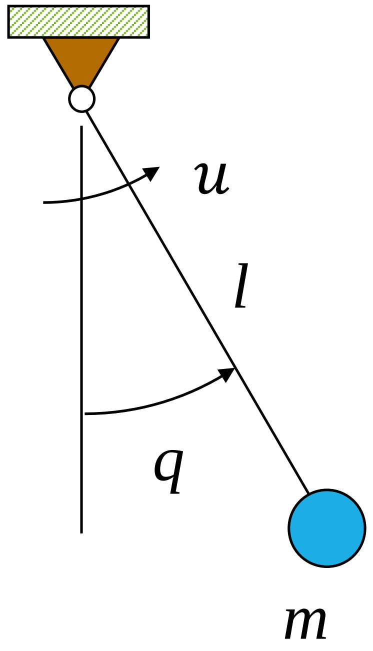
$$\begin{array}{r} \\ \\ \\ \textit{jerk} \\ \textit{snap} \\ \textit{yaw} \\ \\ \\ \end{array} \begin{bmatrix} \boxed{\mathbf{r}} \\ \mathbf{v} \\ \mathbf{a} \\ \mathbf{j} \\ \mathbf{s} \\ \boxed{\psi} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} \longleftrightarrow \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \\ u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$$

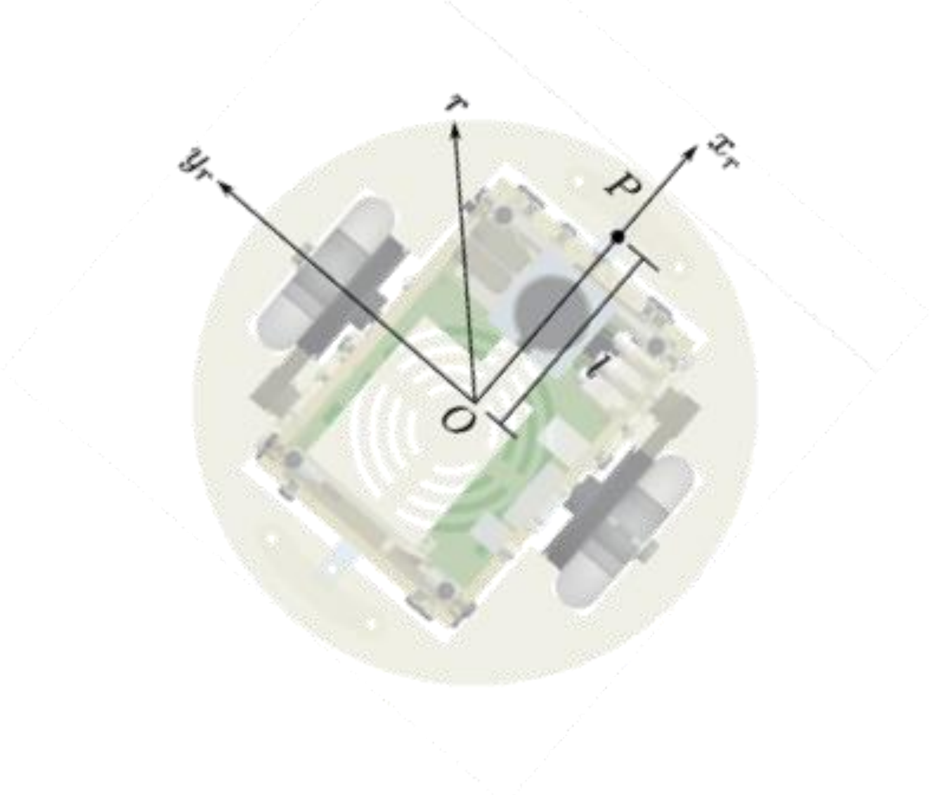
configuration coordinates for SE(3)

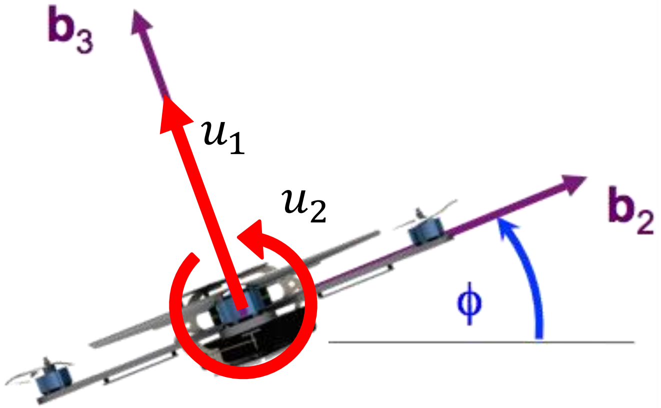
[Mellinger and Kumar, ICRA 2011]

Useful For: Control Design

If the system is differentially flat, then we can always derive a controller using dynamic feedback linearization.

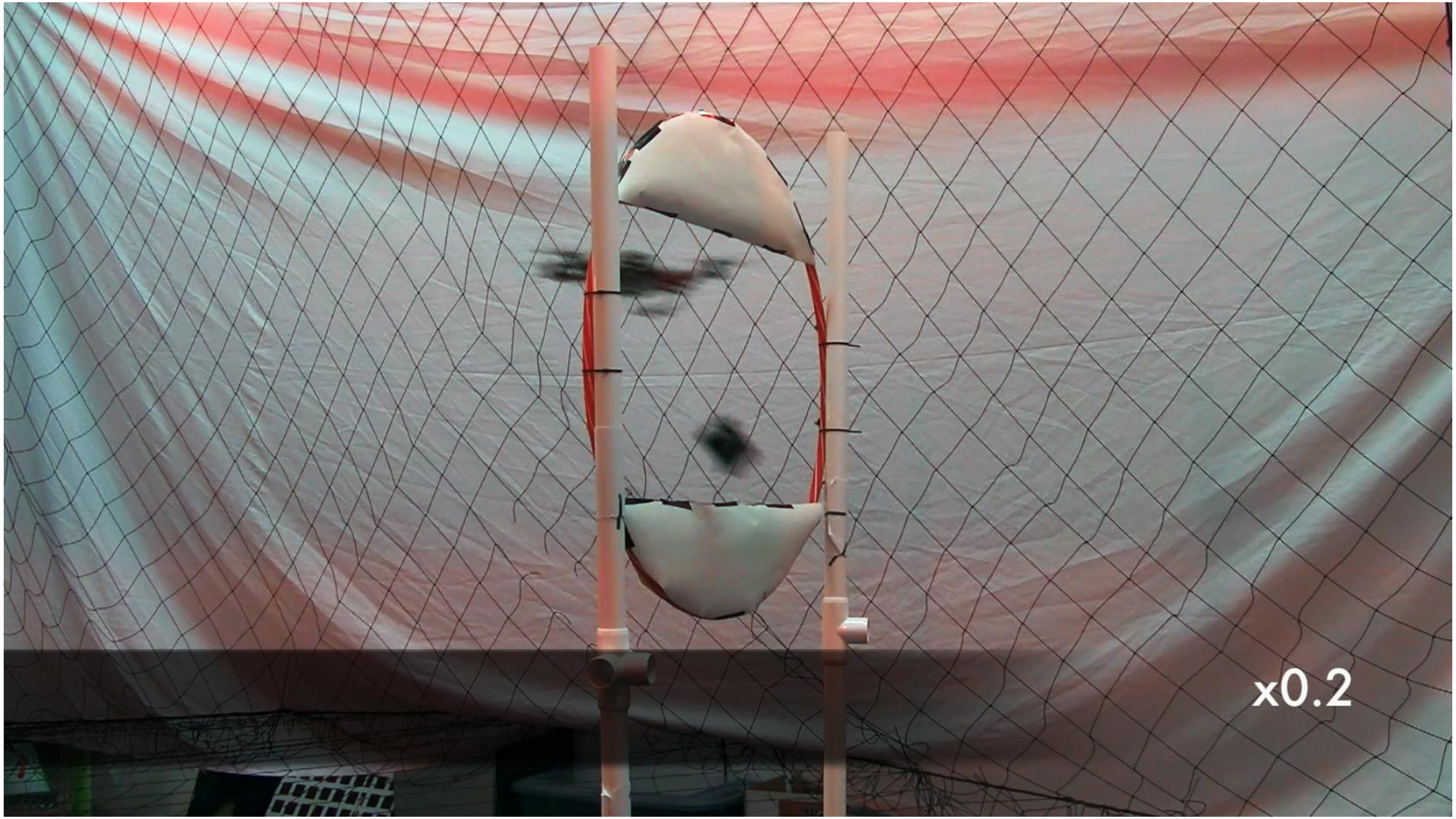






Suspended Payload

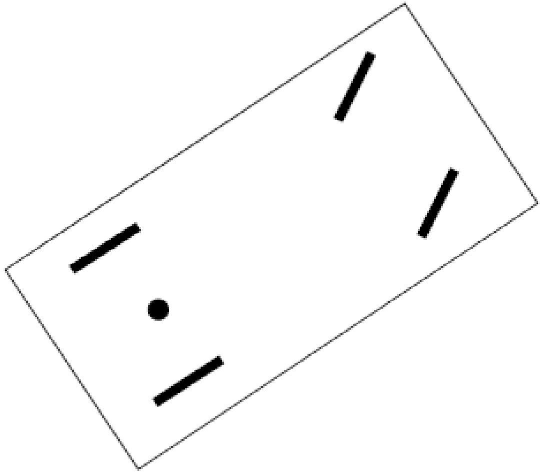
Sarah Tang

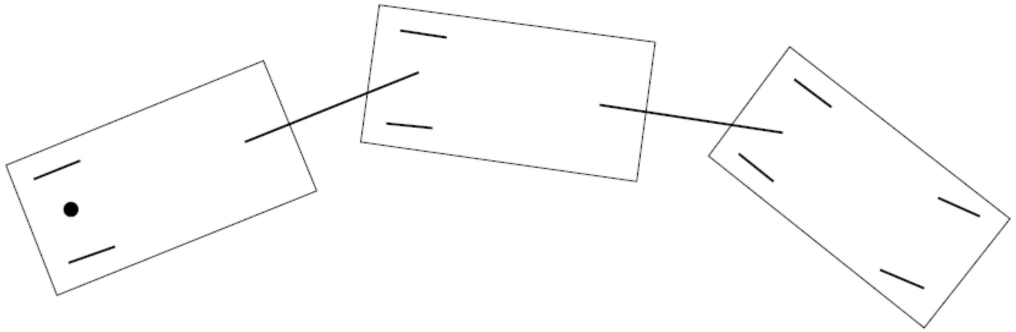


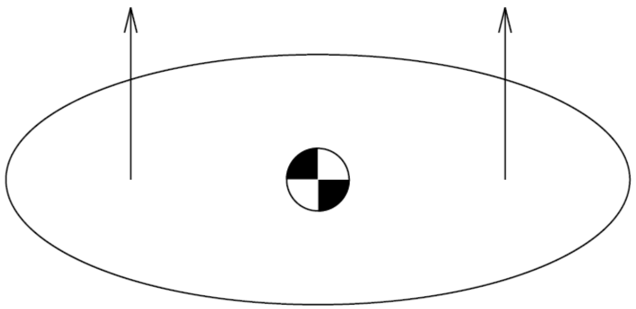
How to Identify A Flat Output

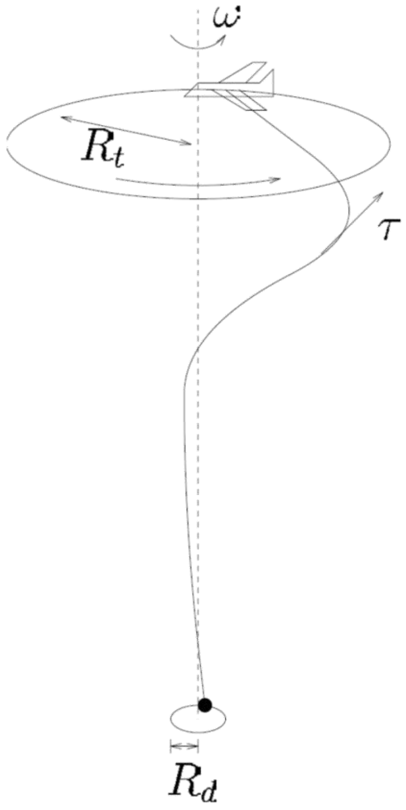
No general method; constructive methods exist only for some special cases.

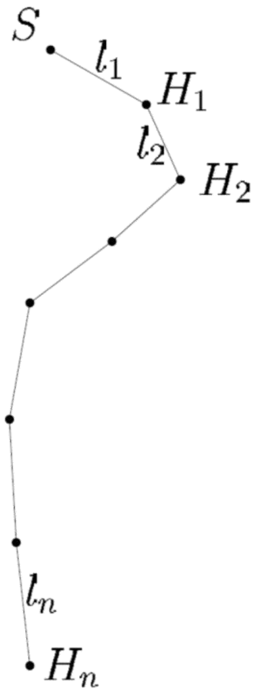
See Murray 1995 for a sampling of results for robotics / mechanical systems.











Martin, Murray, Rouchon 2003

Project 1-3 Brief

Out shortly, but you can guess what it is.

***** *lots of room for improvement*

P1-2	Map	→	Waypoints
P1-1	Waypoints	→	Trajectory *****
	Trajectory	→	Flight

P1-3

You can get an early start by sticking together the sandboxes from P1-1 and P1-2.

Connective Tissue (and room for creativity)

How do you assign times to waypoints?

- One simple idea: based on distance assuming a max velocity, if segments are long.

How can you select “good” waypoints?

- One simple idea: remove points that are in a straight line.
- More complex idea: Ramer–Douglas–Peucker (find a “similar” curve with fewer points)

How do you avoid crashing into things?

- Use margin to pick a safe path, and stay close to that path.

How do I leverage my excellent P1-1 and P1-2 code?

- If your A* is fast, you can use a smaller *resolution*.
- If your controller is accurate, you can use a smaller *margin*.