Nonlinear Pose Tracking Controller for Bar Tethered to Two Aerial Vehicles with Bounded Linear and Angular Accelerations

Pedro Pereira and Dimos V. Dimarogonas

Department of Automatic Control,
KTH Royal Institute of Technology

December 14th

2018-01-07

Collaborative Transportation of a Bar

Nonlinear Pose Tracking Controller for Bar Tethered to Two Aerial Vehicles with Bounded Linear and Angular Accelerations

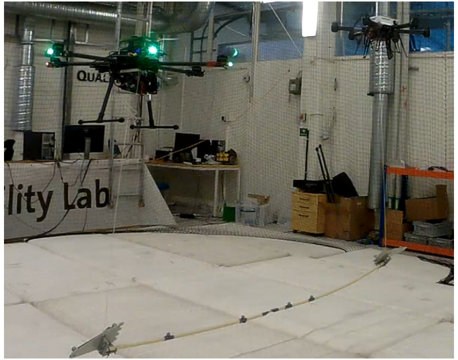
Pedro Pereira and Dimos V. Dimarogonas

Department of Automatic Control,
KTH Royal Institute of Technology

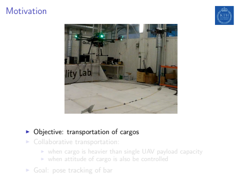
December 14th

- Submission number: 1212
- https://css.paperplaza.net/conferences/conferences/CDC17/program/CDC17_ContentListWeb_3.html
- Schedule Code: ThB17. Room: **218**
- Tentative time of presentation: Thursday December 14, 2017, 10:40–11:00 (GMT+11)
- Session: Lyapunov Methods I
- Chair: Morten Hovd
- Co-chair: Marc Jungers

Motivation

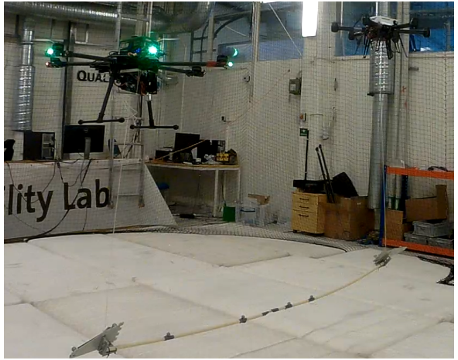


- Objective: transportation of cargos
- Collaborative transportation:
  - when cargo is heavier than single UAV payload capacity
  - when attitude of cargo is also be controlled
- Goal: pose tracking of bar



- Cable does not require a power supply (if it is to be retracted, it does)
- Manipulator is mechanically complex and heavy
- Tethered *vs* manipulator-endowed transportation
- Collaborative transportation: redundancy and resistance to single UAV failure
  - goal: trajectory tracking for a bar tethered to two UAVs
  - cable is mechanically simple and light
  - manipulator provides extra degrees of freedom
- Do it for heterogenous vehicles, cables of different lengths, . . .

Motivation

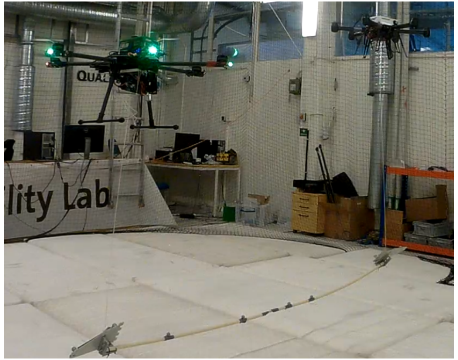


- Objective: transportation of cargos
- Collaborative transportation:
  - when cargo is heavier than single UAV payload capacity
  - when attitude of cargo is also be controlled
- Goal: pose tracking of bar

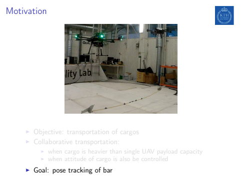
Motivation
Objective: transportation of cargos
Collaborative transportation:
when cargo is heavier than single UAV payload capacity
when attitude of cargo is also be controlled
Goal: pose tracking of bar

- Cable does not require a power supply (if it is to be retracted, it does)
- Manipulator is mechanically complex and heavy
- Tethered *vs* manipulator-endowed transportation
- Collaborative transportation: redundancy and resistance to single UAV failure
  - goal: trajectory tracking for a bar tethered to two UAVs
  - cable is mechanically simple and light
  - manipulator provides extra degrees of freedom
- Do it for heterogenous vehicles, cables of different lengths, . . .

Motivation



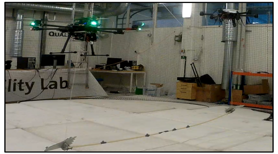
- Objective: transportation of cargos
- Collaborative transportation:
  - when cargo is heavier than single UAV payload capacity
  - when attitude of cargo is also be controlled
- Goal: pose tracking of bar



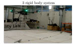
- Cable does not require a power supply (if it is to be retracted, it does)
- Manipulator is mechanically complex and heavy
- Tethered *vs* manipulator-endowed transportation
- Collaborative transportation: redundancy and resistance to single UAV failure
  - goal: trajectory tracking for a bar tethered to two UAVs
  - cable is mechanically simple and light
  - manipulator provides extra degrees of freedom
- Do it for heterogenous vehicles, cables of different lengths, ...

Problem and Strategy

3 rigid body system



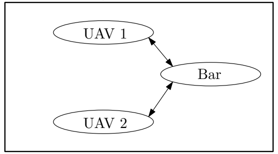
Problem and Strategy



- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

Problem and Strategy

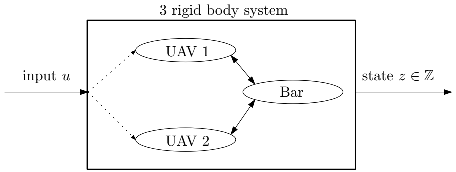
3 rigid body system

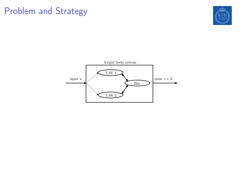


Problem and Strategy

- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

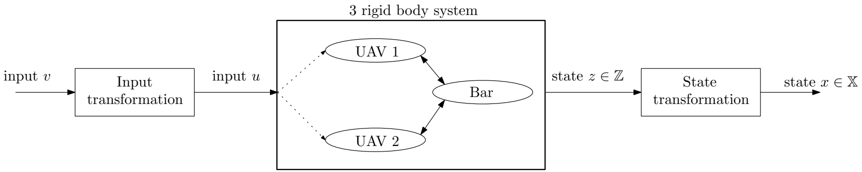
Problem and Strategy

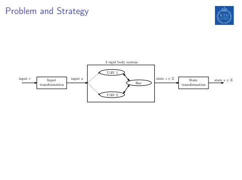




- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

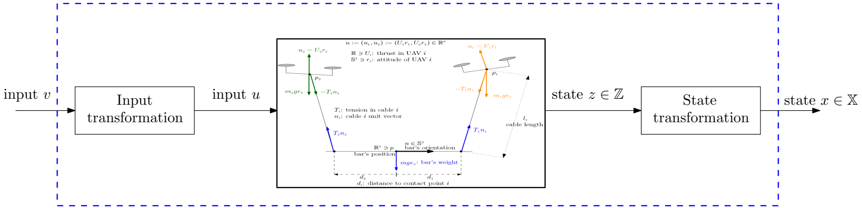
Problem and Strategy

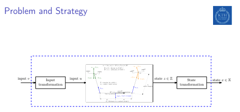




- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

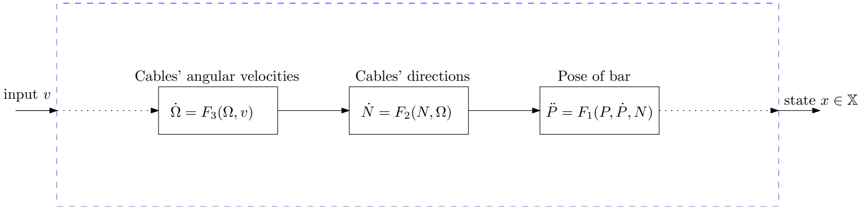
Problem and Strategy

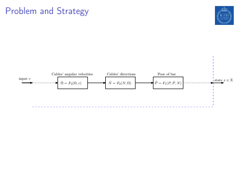




- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

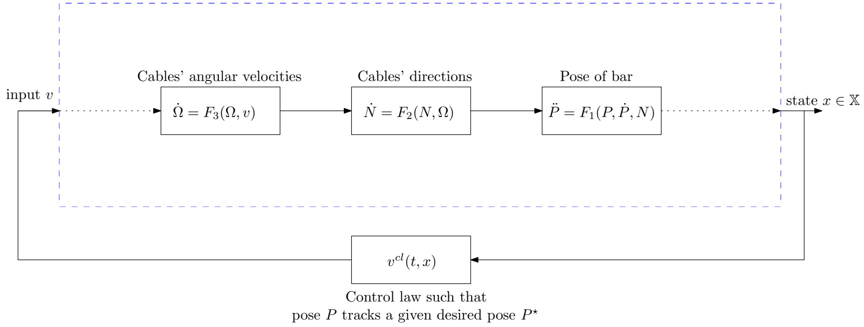
Problem and Strategy

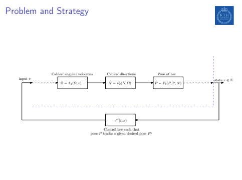




- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

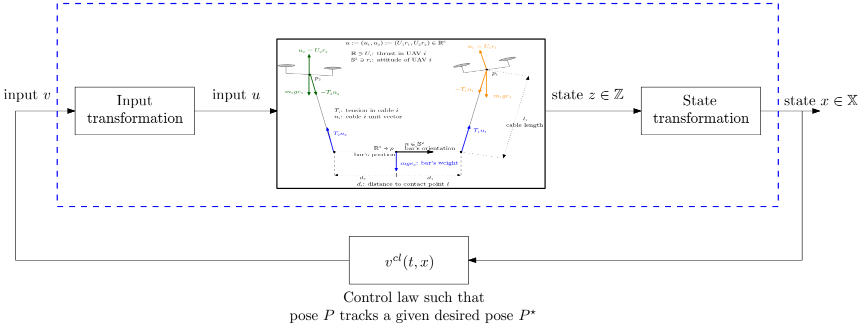
Problem and Strategy

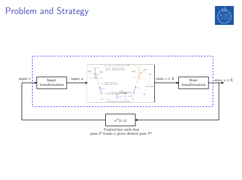




- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

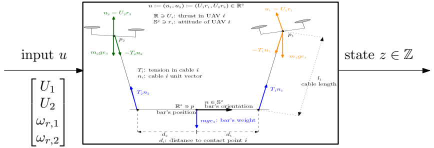
Problem and Strategy

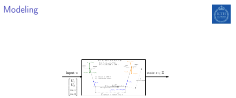




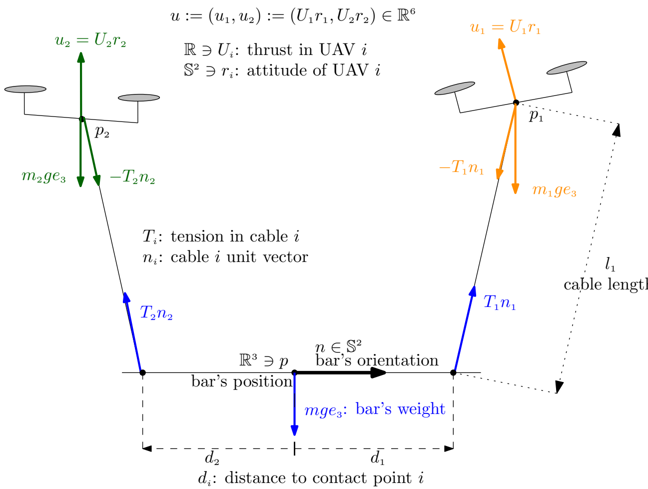
- **pose=(position, attitude) stabilization of bar** – generalization of slung-load with one UAV
- steps for solving the problem
- model system as coupling between 3 bodies
- Find input and state transformation, such that composition yields a new vector field
- New vector field is a cascade of three systems: at the end of the cascade is the pose of the bar (pose = position + orientation)
- given desired pose trajectory, explore cascaded structure for control desgin (by backstepping)

Modeling

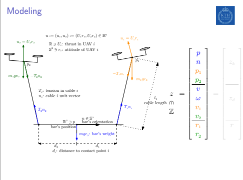

input u
$\begin{bmatrix} U_1 \\ U_2 \\ \omega_{r,1} \\ \omega_{r,2} \end{bmatrix}$
state $z \in \mathbb{Z}$
$u := (u_1, u_2) := (U_1 r_1, U_2 r_2) \in \mathbb{R}^6$
$\mathbb{R} \ni U_i$: thrust in UAV i
$\mathbb{S}^2 \ni r_i$: attitude of UAV i
$u_2 = U_2 r_2$
$u_1 = U_1 r_1$
p_2
p_1
$m_2 g e_3$
$-T_2 n_2$
$-T_1 n_1$
$m_1 g e_3$
T_i: tension in cable i
n_i: cable i unit vector
l_1
cable length
$T_2 n_2$
$T_1 n_1$
$n \in \mathbb{S}^2$
bar's orientation
$\mathbb{R}^3 \ni p$
bar's position
$m g e_3$: bar's weight
d_2
d_1
d_i: distance to contact point i



Modeling

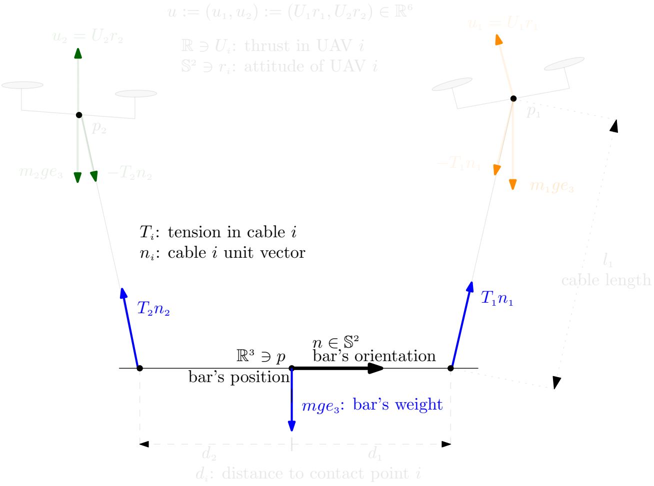


$$z = \left[\begin{array}{c} p \\ n \\ p_1 \\ p_2 \\ \hline v \\ \omega \\ v_1 \\ v_2 \\ \hline r_1 \\ r_2 \end{array}\right] = \left[\begin{array}{c} z_k \\ \hline z_d \\ \hline r \end{array}\right]$$

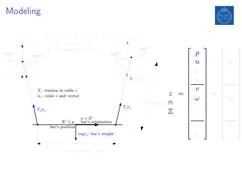


- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

Modeling

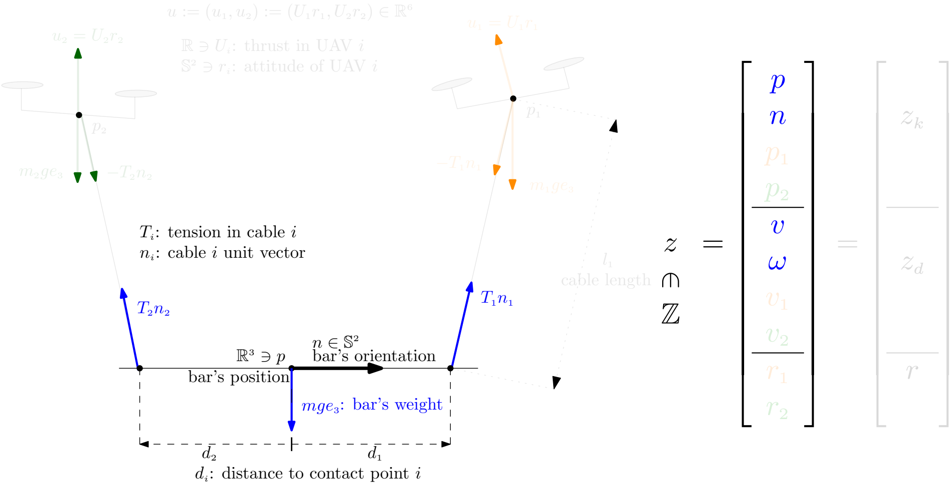


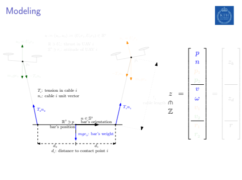
$$\begin{array}{c} z \\ \in \\ \mathbb{Z} \end{array} = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ \hline v \\ \omega \\ v_1 \\ v_2 \\ \hline r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} z_k \\ \hline z_d \\ \hline r \end{bmatrix}$$



- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

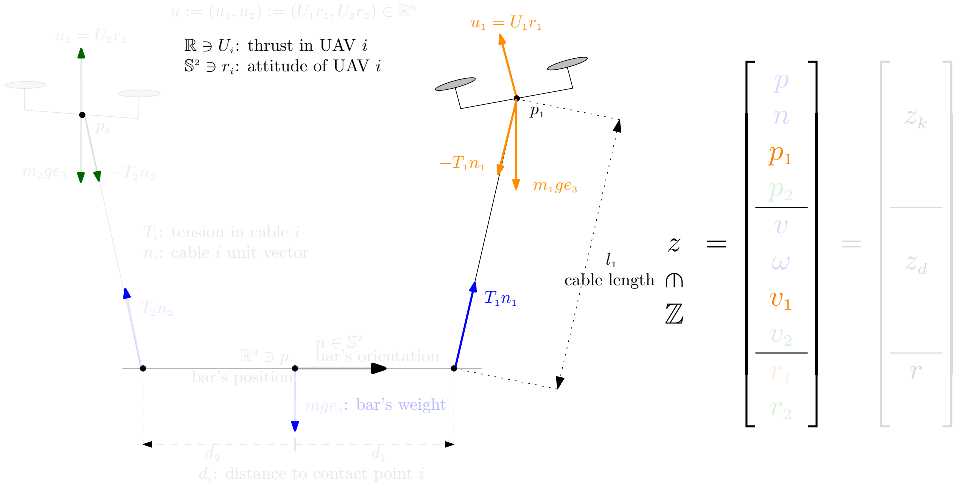
Modeling

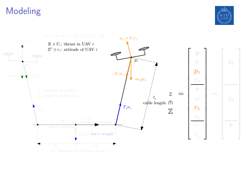




- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

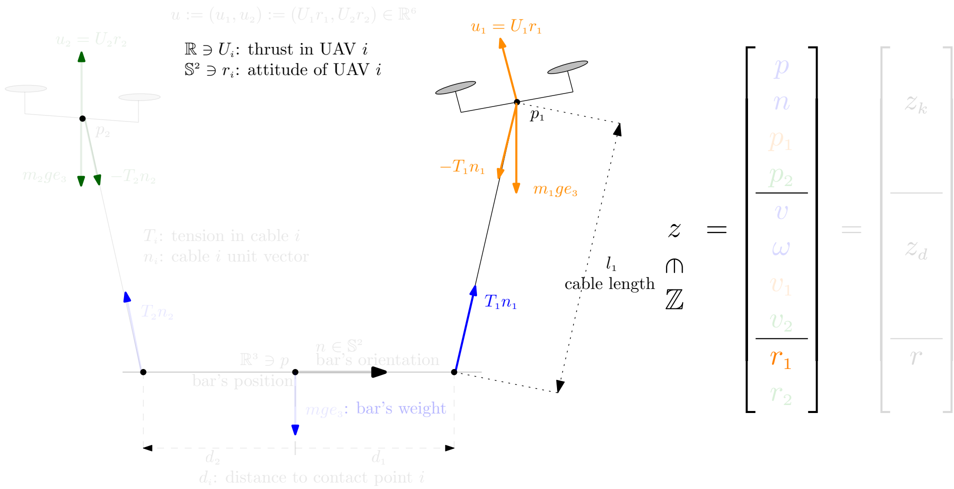
Modeling

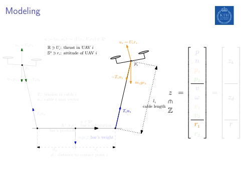




- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

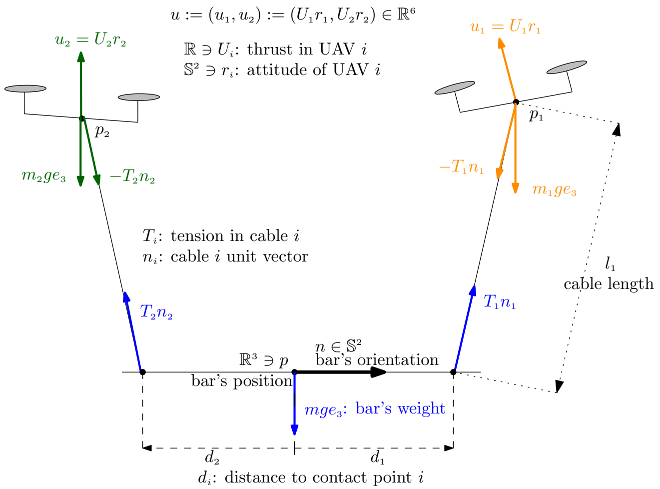
Modeling



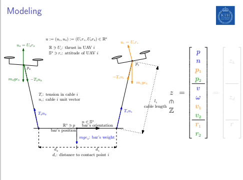


- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

Modeling

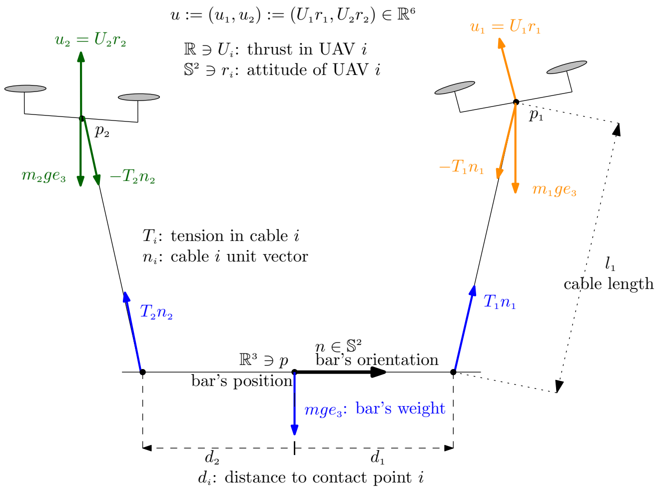


$$\mathbb{Z} \ni z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ \hline v \\ \omega \\ v_1 \\ v_2 \\ \hline r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} z_k \\ \hline z_d \\ \hline r \end{bmatrix}$$

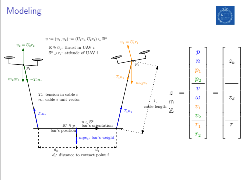


- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

Modeling

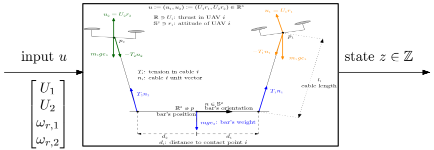


$$
\mathbb{Z} \ni z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ \hline v \\ \omega \\ v_1 \\ v_2 \\ \hline r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} z_k \\ \hline z_d \\ \hline r \end{bmatrix}
$$

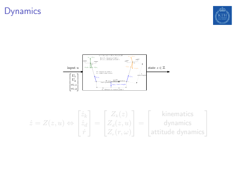


- State decomposition
- Linear and angular positions of bar + linear and angular velocities of bar
- Linear position of UAV + linear velocity of UAV (for both UAVs)
- Angular position of UAV, and we assume we control its angular velocity
- Integral states for the PID control laws

Dynamics

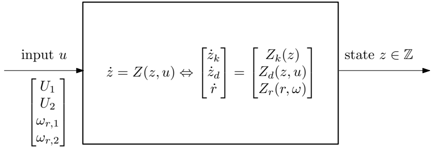


$$\dot{z} = Z(z,u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \\ \dot{r} \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z,u) \\ Z_r(r,\omega) \end{bmatrix} = \begin{bmatrix} \text{kinematics} \\ \text{dynamics} \\ \text{attitude dynamics} \end{bmatrix}$$

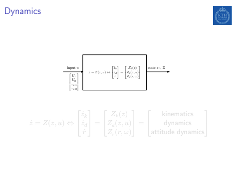


- Replace model with “open-loop vector field”
- Explain three components individually next: kinematics + dynamics + UAV attitude dynamics

Dynamics

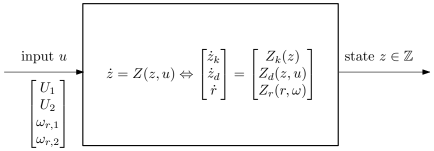


$$\dot{z} = Z(z,u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \\ \dot{r} \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z,u) \\ Z_r(r,\omega) \end{bmatrix} = \begin{bmatrix} \text{kinematics} \\ \text{dynamics} \\ \text{attitude dynamics} \end{bmatrix}$$

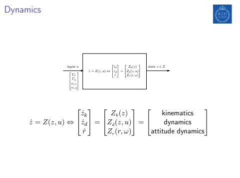


- Replace model with "open-loop vector field"
- Explain three components individually next: kinematics + dynamics + UAV attitude dynamics

Dynamics

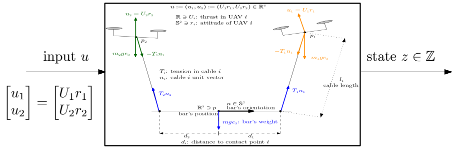


$$\dot{z} = Z(z,u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \\ \dot{r} \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z,u) \\ Z_r(r,\omega) \end{bmatrix} = \begin{bmatrix} \text{kinematics} \\ \text{dynamics} \\ \text{attitude dynamics} \end{bmatrix}$$

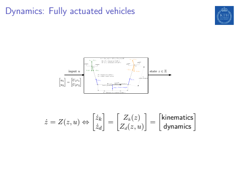


- Replace model with “open-loop vector field”
- Explain three components individually next: kinematics + dynamics + UAV attitude dynamics

Dynamics: Fully actuated vehicles

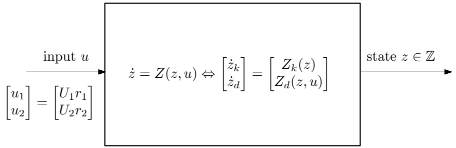


$$\dot{z} = Z(z, u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z, u) \end{bmatrix} = \begin{bmatrix} \text{kinematics} \\ \text{dynamics} \end{bmatrix}$$

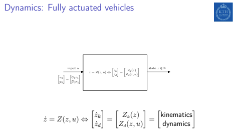


- Replace model with "open-loop vector field"
- Explain three components individually next: kinematics + dynamics + UAV attitude dynamics

Dynamics: Fully actuated vehicles



$$\dot{z} = Z(z,u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z,u) \end{bmatrix} = \begin{bmatrix} \text{kinematics} \\ \text{dynamics} \end{bmatrix}$$



- Replace model with "open-loop vector field"
- Explain three components individually next: kinematics + dynamics + UAV attitude dynamics

Kinematics

input u

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} U_1 r_1 \\ U_2 r_2 \end{bmatrix}$$

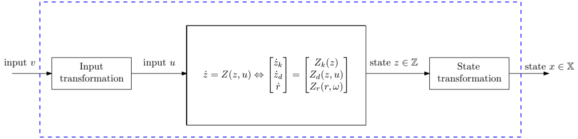
$$\dot{z} = Z(z, u) \Leftrightarrow \begin{bmatrix} \dot{z}_k \\ \dot{z}_d \end{bmatrix} = \begin{bmatrix} Z_k(z) \\ Z_d(z, u) \end{bmatrix}$$

state $z \in \mathbb{Z}$

$$\dot{z}_k = Z_k(z) \Leftrightarrow \begin{bmatrix} \dot{p} \\ \dot{n} \\ \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} = \begin{bmatrix} v \\ \mathcal{S}(\omega) n \\ v_1 \\ v_2 \end{bmatrix}$$

- kinematic equations

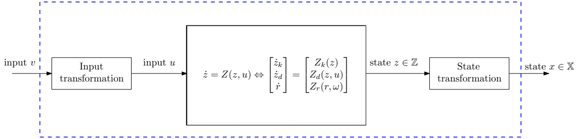
Dynamics



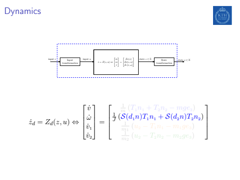
$$\dot{z}_d = Z_d(z,u) \Leftrightarrow \begin{bmatrix} \dot{v} \\ \dot{\omega} \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\left(T_1 n_1 + T_2 n_2 - m g e_3\right) \\ \frac{1}{J}\left(\mathcal{S}(d_1 n) T_1 n_1 + \mathcal{S}(d_2 n) T_2 n_2\right) \\ \frac{1}{m_1}\left(u_2 - T_1 n_1 - m_1 g e_3\right) \\ \frac{1}{m_2}\left(u_2 - T_2 n_2 - m_2 g e_3\right) \end{bmatrix}$$

- g stands for the acceleration due to gravity
- T_1, T_2 stand for the tensions on the cables
- tensions on the cables are functions of the state and the input
- Cable direction: $n_1 \equiv n_1(z) = \frac{p_1 - (p + d_1 n)}{l_1}$
- Tension on cable: $T_1 = T_1(z,u) = \ldots$

Dynamics

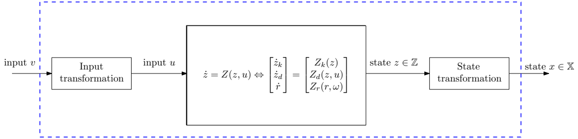


$$\dot{z}_d = Z_d(z,u) \Leftrightarrow \begin{bmatrix} \dot{v} \\ \dot{\omega} \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\left(T_1 n_1 + T_2 n_2 - m g e_3\right) \\ \frac{1}{J}\left(\mathcal{S}(d_1 n) T_1 n_1 + \mathcal{S}(d_2 n) T_2 n_2\right) \\ \frac{1}{m_1}\left(u_2 - T_1 n_1 - m_1 g e_3\right) \\ \frac{1}{m_2}\left(u_2 - T_2 n_2 - m_2 g e_3\right) \end{bmatrix}$$

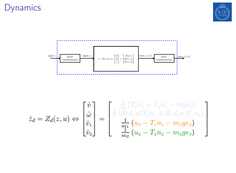


- g stands for the acceleration due to gravity
- T_1, T_2 stand for the tensions on the cables
- tensions on the cables are functions of the state and the input
- Cable direction: $n_1 \equiv n_1(z) = \frac{p_1 - (p + d_1 n)}{l_1}$
- Tension on cable: $T_1 = T_1(z,u) = \ldots$

Dynamics

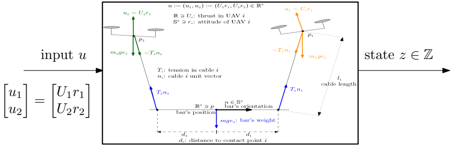


$$\dot{z}_d = Z_d(z,u) \Leftrightarrow \begin{bmatrix}\dot{v} \\ \dot{\omega} \\ \dot{v}_1 \\ \dot{v}_2\end{bmatrix} = \begin{bmatrix}\frac{1}{m}\left(T_1 n_1 + T_2 n_2 - m g e_3\right) \\ \frac{1}{J}\left(\mathcal{S}(d_1 n) T_1 n_1 + \mathcal{S}(d_2 n) T_2 n_2\right) \\ \frac{1}{m_1}\left(u_2 - T_1 n_1 - m_1 g e_3\right) \\ \frac{1}{m_2}\left(u_2 - T_2 n_2 - m_2 g e_3\right)\end{bmatrix}$$



- g stands for the acceleration due to gravity
- T_1, T_2 stand for the tensions on the cables
- tensions on the cables are functions of the state and the input
- Cable direction: $n_1 \equiv n_1(z) = \frac{p_1 - (p + d_1 n)}{l_1}$
- Tension on cable: $T_1 = T_1(z,u) = \ldots$

Problem statement



Problem

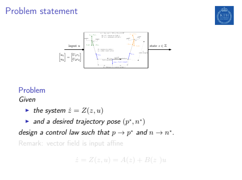
Given

- *the system* $\dot{z} = Z(z, u)$
- *and a desired trajectory pose* $(p^\star, n^\star)$

design a control law such that $p \to p^\star$ *and* $n \to n^\star$.

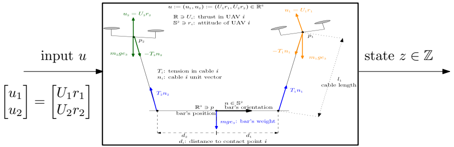
Remark: vector field is input affine

$$\dot{z} = Z(z, u) = A(z) + B(z)u$$



- being input affine is helpfull

Problem statement



Problem

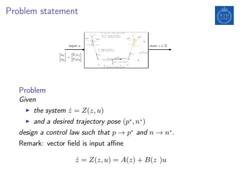
Given

- *the system* $\dot{z} = Z(z, u)$
- *and a desired trajectory pose* $(p^\star, n^\star)$

design a control law such that $p \to p^\star$ *and* $n \to n^\star$.

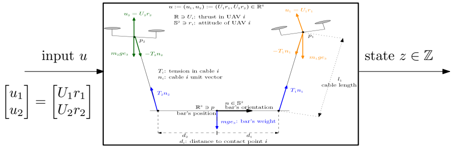
Remark: vector field is input affine

$$\dot{z} = Z(z, u) = A(z) + B(z\ )u$$



- being input affine is helpfull

Problem statement



Problem

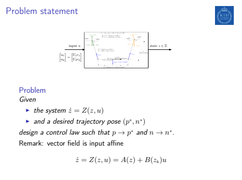
Given

- *the system* $\dot{z} = Z(z, u)$
- *and a desired trajectory pose* $(p^\star, n^\star)$

design a control law such that $p \to p^\star$ *and* $n \to n^\star$.

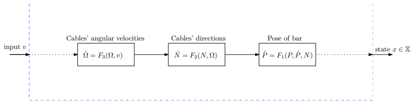
Remark: vector field is input affine

$$\dot{z} = Z(z, u) = A(z) + B(z_k)u$$



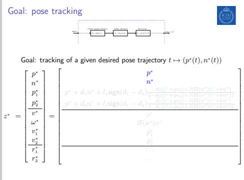
- being input affine is helpfull

Goal: pose tracking



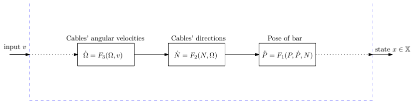
Goal: tracking of a given desired pose trajectory $t \mapsto (p^\star(t), n^\star(t))$

$$
z^\star = \left[\begin{array}{c} p^\star \\ n^\star \\ p_1^\star \\ p_2^\star \\ \hline v^\star \\ \omega^\star \\ v_1^\star \\ v_2^\star \\ \hline r_1^\star \\ r_2^\star \end{array}\right] = \left[\begin{array}{c} p^\star \\ n^\star \\ p^\star + d_1 n^\star + l_1 \mathrm{sign}(d_1 - d_2) \frac{-d_2(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)}{\|-d_2(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)\|} \\ p^\star + d_2 n^\star + l_2 \mathrm{sign}(d_1 - d_2) \frac{d_1(\ddot p^\star + g e_3) - J(\Pi(n^\star)\ddot n^\star + k n^\star)}{\|d_1(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)\|} \\ \hline \dot p^\star \\ S(n^\star)\dot n^\star \\ \dot p_1^\star \\ \dot p_2^\star \\ \hline \cdots \\ \cdots \end{array}\right]
$$



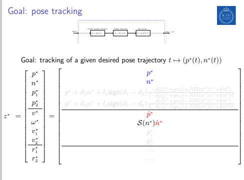
- We are given a desired pose: $t \mapsto P^\star(t) = (p^\star, n^\star)$
- We can find the **equilibrium state trajectory** for which the pose is tracked
- Not all pose trajectories are trackable
- There is also a **equilibrium input trajectory**
- Differential flatness w.r.t. $(p, n) = (p^\star, n^\star)$

Goal: pose tracking



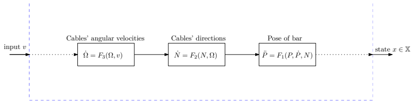
Goal: tracking of a given desired pose trajectory $t \mapsto (p^\star(t), n^\star(t))$

$$z^\star = \begin{bmatrix} p^\star \\ n^\star \\ p_1^\star \\ p_2^\star \\ \hline v^\star \\ \omega^\star \\ v_1^\star \\ v_2^\star \\ \hline r_1^\star \\ r_2^\star \end{bmatrix} = \begin{bmatrix} p^\star \\ n^\star \\ p^\star + d_1 n^\star + l_1 \mathsf{sign}(d_1 - d_2) \frac{-d_2(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)}{\|-d_2(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)\|} \\ p^\star + d_2 n^\star + l_2 \mathsf{sign}(d_1 - d_2) \frac{d_1(\ddot p^\star + g e_3) - J(\Pi(n^\star)\ddot n^\star + k n^\star)}{\|d_1(\ddot p^\star + g e_3) + J(\Pi(n^\star)\ddot n^\star + k n^\star)\|} \\ \hline \dot p^\star \\ \mathcal{S}(n^\star)\dot n^\star \\ \dot p_1^\star \\ \dot p_2^\star \\ \hline \cdots \\ \cdots \end{bmatrix}$$



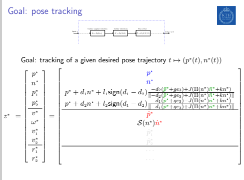
- We are given a desired pose: $t \mapsto P^\star(t) = (p^\star, n^\star)$
- We can find the **equilibrium state trajectory** for which the pose is tracked
- Not all pose trajectories are trackable
- There is also a **equilibrium input trajectory**
- Differential flatness w.r.t. $(p, n) = (p^\star, n^\star)$

Goal: pose tracking



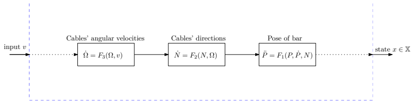
Goal: tracking of a given desired pose trajectory $t \mapsto (p^\star(t), n^\star(t))$

$$
z^\star = \left[\begin{array}{c} p^\star \\ n^\star \\ p_1^\star \\ p_2^\star \\ \hline v^\star \\ \omega^\star \\ v_1^\star \\ v_2^\star \\ \hline r_1^\star \\ r_2^\star \end{array}\right] = \left[\begin{array}{c} p^\star \\ n^\star \\ p^\star + d_1 n^\star + l_1 \mathsf{sign}(d_1 - d_2)\frac{-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\|-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ p^\star + d_2 n^\star + l_2 \mathsf{sign}(d_1 - d_2)\frac{d_1(\ddot{p}^\star + g e_3) - J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\| d_1(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ \hline \dot{p}^\star \\ \mathcal{S}(n^\star)\dot{n}^\star \\ \dot{p}_1^\star \\ \dot{p}_2^\star \\ \hline \cdots \\ \cdots \end{array}\right]
$$



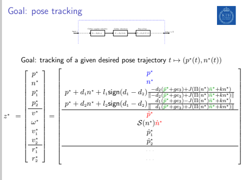
- We are given a desired pose: $t \mapsto P^\star(t) = (p^\star, n^\star)$
- We can find the **equilibrium state trajectory** for which the pose is tracked
- Not all pose trajectories are trackable
- There is also a **equilibrium input trajectory**
- Differential flatness w.r.t. $(p, n) = (p^\star, n^\star)$

Goal: pose tracking

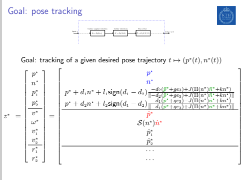


Goal: tracking of a given desired pose trajectory $t \mapsto (p^\star(t), n^\star(t))$

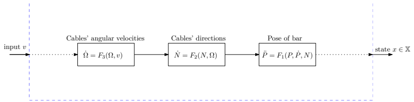
$$
z^\star = \left[\begin{array}{c} p^\star \\ n^\star \\ p_1^\star \\ p_2^\star \\ \hline v^\star \\ \omega^\star \\ v_1^\star \\ v_2^\star \\ \hline r_1^\star \\ r_2^\star \end{array}\right] = \left[\begin{array}{c} p^\star \\ n^\star \\ p^\star + d_1 n^\star + l_1 \mathsf{sign}(d_1 - d_2) \frac{-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\|-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ p^\star + d_2 n^\star + l_2 \mathsf{sign}(d_1 - d_2) \frac{d_1(\ddot{p}^\star + g e_3) - J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\| d_1(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ \hline \dot{p}^\star \\ \mathcal{S}(n^\star)\dot{n}^\star \\ \dot{p}_1^\star \\ \dot{p}_2^\star \\ \hline \cdots \\ \cdots \end{array}\right]
$$



- We are given a desired pose: $t \mapsto P^\star(t) = (p^\star, n^\star)$
- We can find the **equilibrium state trajectory** for which the pose is tracked
- Not all pose trajectories are trackable
- There is also a **equilibrium input trajectory**
- Differential flatness w.r.t. $(p, n) = (p^\star, n^\star)$



Goal: pose tracking

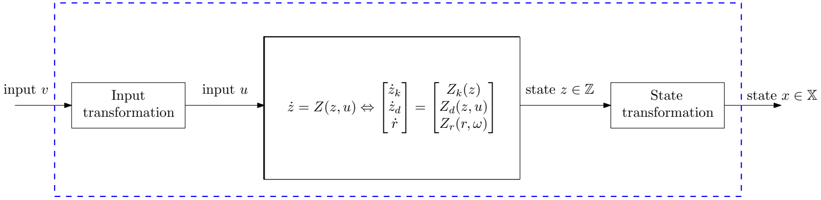


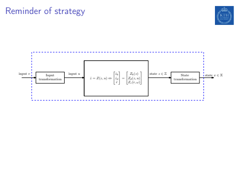
Goal: tracking of a given desired pose trajectory $t \mapsto (p^\star(t), n^\star(t))$

$$
z^\star = \left[\begin{array}{c} p^\star \\ n^\star \\ p_1^\star \\ p_2^\star \\ \hline v^\star \\ \omega^\star \\ v_1^\star \\ v_2^\star \\ \hline r_1^\star \\ r_2^\star \end{array}\right] = \left[\begin{array}{c} p^\star \\ n^\star \\ p^\star + d_1 n^\star + l_1 \mathsf{sign}(d_1 - d_2)\frac{-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\|-d_2(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ p^\star + d_2 n^\star + l_2 \mathsf{sign}(d_1 - d_2)\frac{d_1(\ddot{p}^\star + g e_3) - J(\Pi(n^\star)\ddot{n}^\star + k n^\star)}{\| d_1(\ddot{p}^\star + g e_3) + J(\Pi(n^\star)\ddot{n}^\star + k n^\star)\|} \\ \hline \dot{p}^\star \\ \mathcal{S}(n^\star)\dot{n}^\star \\ \dot{p}_1^\star \\ \dot{p}_2^\star \\ \hline \cdots \\ \cdots \end{array}\right]
$$

- We are given a desired pose: $t \mapsto P^\star(t) = (p^\star, n^\star)$
- We can find the **equilibrium state trajectory** for which the pose is tracked
- Not all pose trajectories are trackable
- There is also a **equilibrium input trajectory**
- Differential flatness w.r.t. $(p, n) = (p^\star, n^\star)$

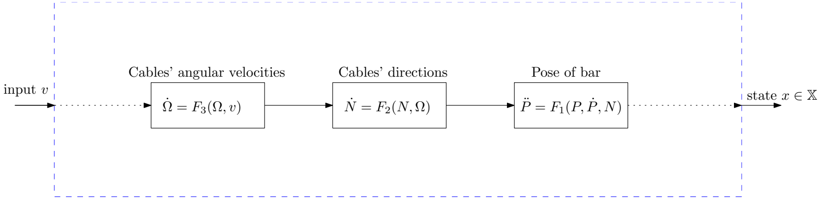
Reminder of strategy

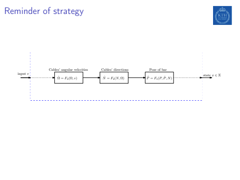




- vector field described before
- equilibrium described before
- equilibrium integral variables depend on model uncertainty
- (local) exponential stability is inferred from linearization

Reminder of strategy





- vector field described before
- equilibrium described before
- equilibrium integral variables depend on model uncertainty
- (local) exponential stability is inferred from linearization

Input transformation

What we wish to consider as inputs

$$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in\mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in\mathbb{R}^{8\times 6}} u$$

Input transformation

$$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

- Change *physical inputs* to more *appropriate inputs*
- Computing tensions and angular accelerations is explained in the article
- Vector field is input affine: $\dot{z} = Z(z, u) = A(z) + B(z_k)u$
- Input transformation is a proper inverse of $v = A_R + B_R u$ provided that v lies in the image space of $A_R + B_R\cdot$ (which it will): angular acceleration of cable i needs to be orthogonal to cable i

Input transformation

What we wish to consider as inputs

$$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in\mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in\mathbb{R}^{8\times 6}} u$$

Input transformation

$$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

Input transformation
What we wish to consider as inputs
$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = A_R(z) + B_R(z_k)\, u$
Input transformation
$u = (B_R(z_k)^T B_R(z_k))^{-1} B_R(z_k)^T (v - A_R(z_k))$

- Change *physical inputs* to more *appropriate inputs*
- Computing tensions and angular accelerations is explained in the article
- Vector field is input affine: $\dot{z} = Z(z, u) = A(z) + B(z_k)u$
- Input transformation is a proper inverse of $v = A_R + B_R u$ provided that v lies in the image space of $A_R + B_R\cdot$ (which it will): angular acceleration of cable i needs to be orthogonal to cable i

Input transformation

What we wish to consider as inputs

$$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in \mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in \mathbb{R}^{8\times 6}} u$$

Input transformation

$$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

Input transformation
What we wish to consider as inputs
$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in \mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in \mathbb{R}^{8\times 6}} u$
Input transformation
$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$

- Change *physical inputs* to more *appropriate inputs*
- Computing tensions and angular accelerations is explained in the article
- Vector field is input affine: $\dot{z} = Z(z, u) = A(z) + B(z_k)u$
- Input transformation is a proper inverse of $v = A_R + B_R u$ provided that v lies in the image space of $A_R + B_R\cdot$ (which it will): angular acceleration of cable i needs to be orthogonal to cable i

Input transformation

What we wish to consider as inputs

$$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in \mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in \mathbb{R}^{8\times 6}} u$$

Input transformation

$$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

Input transformation

What we wish to consider as inputs

$$v = \begin{bmatrix} \text{Tension in cable 1} \\ \text{Tension in cable 2} \\ \text{Angular acceleration of cable 1} \\ \text{Angular acceleration of cable 2} \end{bmatrix} = \underbrace{A_R(z)}_{\in \mathbb{R}^8} + \underbrace{B_R(z_k)}_{\in \mathbb{R}^{8\times 6}} u$$

Input transformation

$$u = \left(B_R(z_k)^T B_R(z_k)\right)^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

- Change *physical inputs* to more *appropriate inputs*
- Computing tensions and angular accelerations is explained in the article
- Vector field is input affine: $\dot{z} = Z(z, u) = A(z) + B(z_k)u$
- Input transformation is a proper inverse of $v = A_R + B_R u$ provided that v lies in the image space of $A_R + B_R\cdot$ (which it will): angular acceleration of cable i needs to be orthogonal to cable i

Input transformation

$$u = (B_R(z_k)^T B_R(z_k))^{-1} B_R(z_k)^T \left(v - A_R(z_k)\right)$$

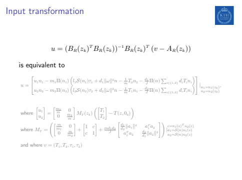
is equivalent to

$$u = \begin{bmatrix} u_1 n_1 - m_1 \Pi(n_1)\left(l_1 \mathcal{S}(n_1)\tau_1 + d_1\|\omega\|^2 n - \frac{1}{m}T_2 n_2 - \frac{d_1}{J}\Pi(n)\sum_{i\in\{1,2\}} d_i T_i n_i\right) \\ u_2 n_2 - m_2 \Pi(n_2)\left(l_2 \mathcal{S}(n_2)\tau_2 + d_2\|\omega\|^2 n - \frac{1}{m}T_1 n_1 - \frac{d_2}{J}\Pi(n)\sum_{i\in\{1,2\}} d_i T_i n_i\right) \end{bmatrix} \Big|_{\substack{n_1 = n_1(z_k) \\ n_2 = n_2(z_k)}},$$

where $\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{m_1}{m} & 0 \\ 0 & \frac{m_2}{m} \end{bmatrix} M_T(z_k) \left(\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - T(z, 0_6) \right)$

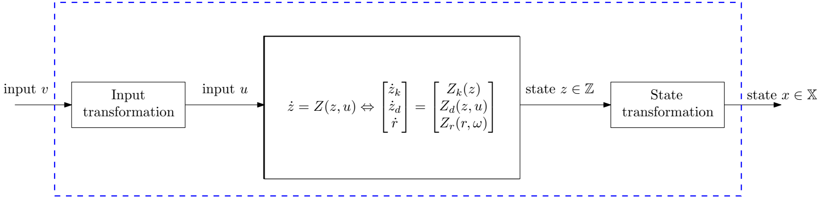
where $M_T = \left(\begin{bmatrix} \frac{m}{m_1} & 0 \\ 0 & \frac{m}{m_2} \end{bmatrix} + \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix} + \frac{m d_1 d_2}{J} \begin{bmatrix} \frac{d_1}{d_2}\|a_1\|^2 & a_1^T a_2 \\ a_1^T a_2 & \frac{d_2}{d_1}\|a_2\|^2 \end{bmatrix} \right) \Big|_{\substack{c = n_1(z)^T n_2(z) \\ a_1 = \mathcal{S}(n) n_1(z) \\ a_2 = \mathcal{S}(n) n_2(z)}}$

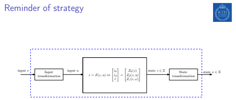
and where $v = (T_1, T_2, \tau_1, \tau_2)$



- Input transformation is well defined

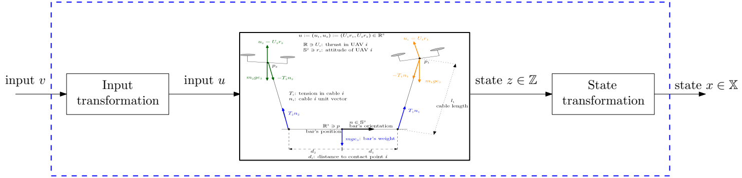
Reminder of strategy





- move on to state transformation

State transformation

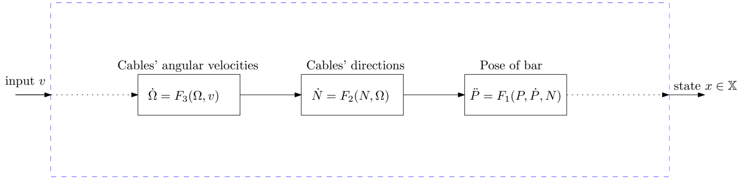


$$z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} g_1(z) \\ \hline \frac{p_1-(p+d_1 n)}{\|p_1-(p+d_1 n)\|} = n_1 \\ \frac{p_2-(p+d_2 n)}{\|p_2-(p+d_2 n)\|} = n_2 \\ \hline \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \\ \hline \\ \hline \end{bmatrix}$$

The inverse mapping g_3^{-1} is also necessary.

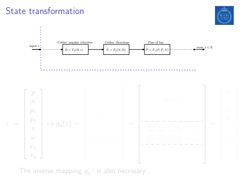
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

State transformation



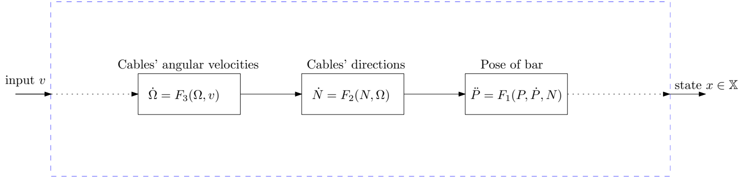
$$z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \quad = \begin{bmatrix} g_1(z) \\ \frac{p_1 - (p + d_1 n)}{\|p_1 - (p + d_1 n)\|} = n_1 \\ \frac{p_2 - (p + d_2 n)}{\|p_2 - (p + d_2 n)\|} = n_2 \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \end{bmatrix}$$

The inverse mapping g_3^{-1} is also necessary.



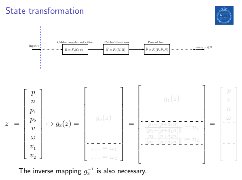
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

State transformation



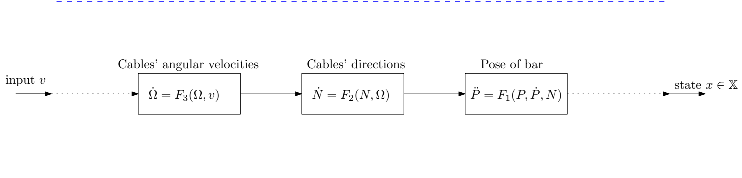
$$z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \begin{bmatrix} g_2(z) \\ \hline \ldots = \omega_1 \\ \ldots = \omega_2 \end{bmatrix} = \begin{bmatrix} g_1(z) \\ \hline \frac{p_1 - (p + d_1 n)}{\|p_1 - (p + d_1 n)\|} = n_1 \\ \frac{p_2 - (p + d_2 n)}{\|p_2 - (p + d_2 n)\|} = n_2 \\ \hline \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \\ \hline \\ \hline \\ \end{bmatrix}$$

The inverse mapping g_3^{-1} is also necessary.



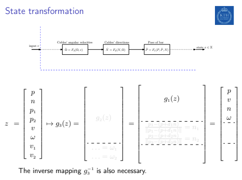
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

State transformation



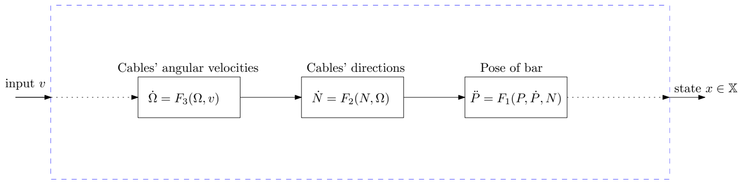
$$z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \begin{bmatrix} g_2(z) \\ \hdashline \ldots = \omega_1 \\ \ldots = \omega_2 \end{bmatrix} = \begin{bmatrix} g_1(z) \\ \hdashline \frac{p_1-(p+d_1 n)}{\|p_1-(p+d_1 n)\|} = n_1 \\ \frac{p_2-(p+d_2 n)}{\|p_2-(p+d_2 n)\|} = n_2 \\ \hdashline \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \\ \hdashline \\ \hdashline \end{bmatrix}$$

The inverse mapping g_3^{-1} is also necessary.



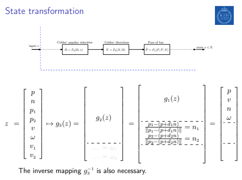
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

State transformation



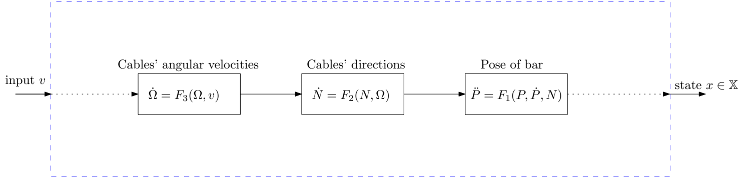
$$z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \begin{bmatrix} g_2(z) \\ \hline \ldots = \omega_1 \\ \ldots = \omega_2 \end{bmatrix} = \begin{bmatrix} g_1(z) \\ \hline \frac{p_1-(p+d_1 n)}{\|p_1-(p+d_1 n)\|} = n_1 \\ \frac{p_2-(p+d_2 n)}{\|p_2-(p+d_2 n)\|} = n_2 \\ \hline \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \\ \hline \\ \hline \\ \end{bmatrix}$$

The inverse mapping g_3^{-1} is also necessary.



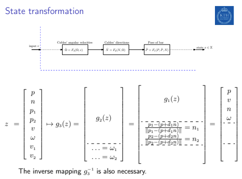
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

State transformation



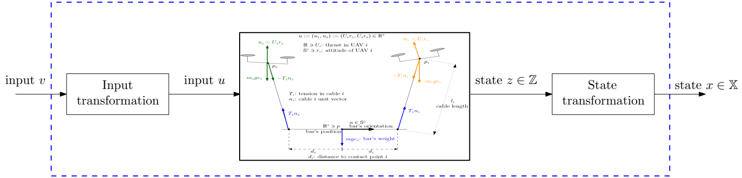
$$
z = \begin{bmatrix} p \\ n \\ p_1 \\ p_2 \\ v \\ \omega \\ v_1 \\ v_2 \end{bmatrix} \mapsto g_3(z) = \begin{bmatrix} g_2(z) \\ \hdashline \ldots = \omega_1 \\ \ldots = \omega_2 \end{bmatrix} = \begin{bmatrix} g_1(z) \\ \hdashline \frac{p_1 - (p + d_1 n)}{\|p_1 - (p + d_1 n)\|} = n_1 \\ \frac{p_2 - (p + d_2 n)}{\|p_2 - (p + d_2 n)\|} = n_2 \\ \hdashline \end{bmatrix} = \begin{bmatrix} p \\ v \\ n \\ \omega \\ \hdashline \\ \hdashline \\ \end{bmatrix}
$$

The inverse mapping g_3^{-1} is also necessary.



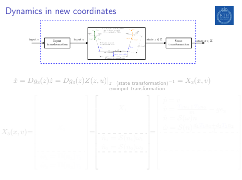
- g_1 isolates the pose of the bar, and the twist of the bar
- g_2 maps to the cables' unit vectors
- g_3 maps to the cables' angular velocities
- The inverse mapping g_3^{-1} is also necessary, and it is found in the article

Dynamics in new coordinates



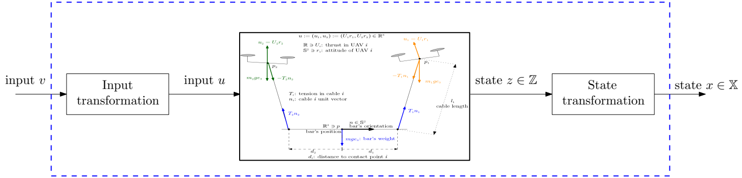
$$\dot{x} = Dg_3(z)\dot{z} = Dg_3(z)Z(z,u)\big|_{\substack{z=(\text{state transformation})^{-1} \\ u=\text{input transformation}}} = X_3(x,v)$$

$$X_3(x,v) = \left[\begin{array}{c} \\ \\ \hline \\ \end{array}\right] = \left[\begin{array}{c} X_1 \\ \hline \dot{n}_1 = S(n_1)\omega_1 \\ \dot{n}_2 = S(n_2)\omega_2 \\ \hline \\ \end{array}\right] = \left[\begin{array}{c} \dot{p} = v \\ \dot{v} = \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \dot{n} = S(\omega)n \\ \dot{\omega} = S(n)\frac{d_1 T_1 n_1 + d_2 T_2 n_2}{J} \\ \hline \\ \hline \\ \end{array}\right]$$



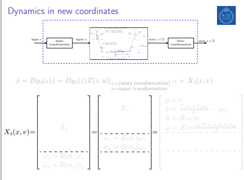
- transformed input $v = (T_1, T_2, \tau_1, \tau_2)$
- The choice of input and state transformations is now clear: it induces a cascaded structure with three layers, which can be explored in the control design process.

Dynamics in new coordinates



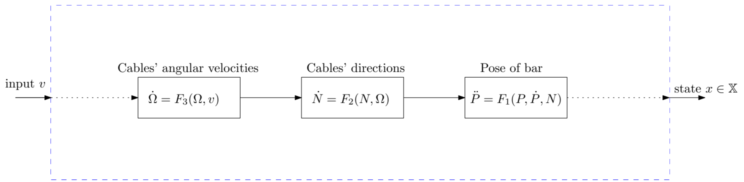
$$\dot{x} = Dg_3(z)\dot{z} = Dg_3(z)Z(z,u)\Big|_{\substack{z=(\text{state transformation})^{-1} \\ u=\text{input transformation}}} = X_3(x,v)$$

$$X_3(x,v) = \begin{bmatrix} X_2 \\ \hline \dot{\omega}_1 = \Pi(n_1)\tau_1 \\ \dot{\omega}_2 = \Pi(n_2)\tau_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ \hline \dot{n}_1 = \mathcal{S}(n_1)\omega_1 \\ \dot{n}_2 = \mathcal{S}(n_2)\omega_2 \\ \hline \end{bmatrix} = \begin{bmatrix} \dot{p} = v \\ \dot{v} = \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \dot{n} = \mathcal{S}(\omega) n \\ \dot{\omega} = \mathcal{S}(n)\frac{d_1 T_1 n_1 + d_2 T_2 n_2}{J} \\ \hline \\ \hline \end{bmatrix}$$



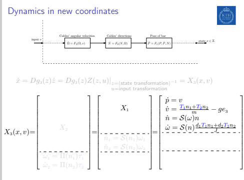
- transformed input $v = (T_1, T_2, \tau_1, \tau_2)$
- The choice of input and state transformations is now clear: it induces a cascaded structure with three layers, which can be explored in the control design process.

Dynamics in new coordinates



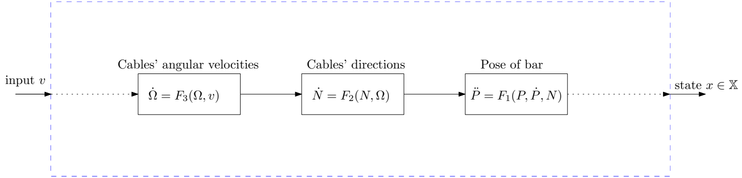
$$\dot{x} = Dg_3(z)\dot{z} = Dg_3(z)Z(z,u)\big|_{\substack{z=(\text{state transformation})^{-1} \\ u=\text{input transformation}}} = X_3(x,v)$$

$$X_3(x,v) = \left[\begin{array}{c} X_2 \\ \hline \dot{\omega}_1 = \Pi(n_1)\tau_1 \\ \dot{\omega}_2 = \Pi(n_2)\tau_2 \end{array}\right] = \left[\begin{array}{c} X_1 \\ \hline \dot{n}_1 = \mathcal{S}(n_1)\omega_1 \\ \dot{n}_2 = \mathcal{S}(n_2)\omega_2 \\ \hline \\ \end{array}\right] = \left[\begin{array}{l} \dot{p} = v \\ \dot{v} = \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \dot{n} = \mathcal{S}(\omega) n \\ \dot{\omega} = \mathcal{S}(n) \frac{d_1 T_1 n_1 + d_2 T_2 n_2}{J} \\ \hline \\ \hline \\ \end{array}\right]$$



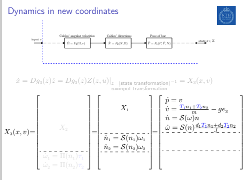
- transformed input $v = (T_1, T_2, \tau_1, \tau_2)$
- The choice of input and state transformations is now clear: it induces a cascaded structure with three layers, which can be explored in the control design process.

Dynamics in new coordinates



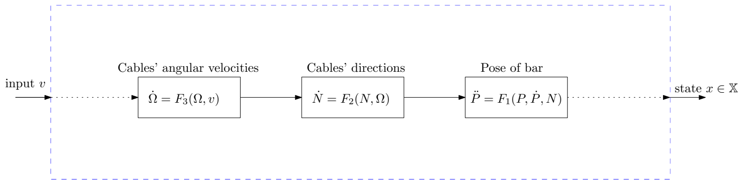
$$\dot x = Dg_3(z)\dot z = Dg_3(z)Z(z,u)\big|_{\substack{z=(\text{state transformation})^{-1}\\ u=\text{input transformation}}} = X_3(x,v)$$

$$X_3(x,v)=\begin{bmatrix} X_2 \\ \hdashline \dot\omega_1 = \Pi(n_1)\tau_1 \\ \dot\omega_2 = \Pi(n_2)\tau_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ \hdashline \dot n_1 = \mathcal{S}(n_1)\omega_1 \\ \dot n_2 = \mathcal{S}(n_2)\omega_2 \\ \hdashline \\ \end{bmatrix} = \begin{bmatrix} \dot p = v \\ \dot v = \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \dot n = \mathcal{S}(\omega) n \\ \dot\omega = \mathcal{S}(n)\frac{d_1 T_1 n_1 + d_2 T_2 n_2}{J} \\ \hdashline \\ \hdashline \\ \end{bmatrix}$$



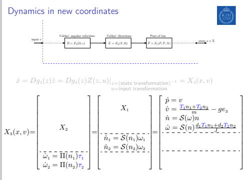
- transformed input $v = (T_1, T_2, \tau_1, \tau_2)$
- The choice of input and state transformations is now clear: it induces a cascaded structure with three layers, which can be explored in the control design process.

Dynamics in new coordinates



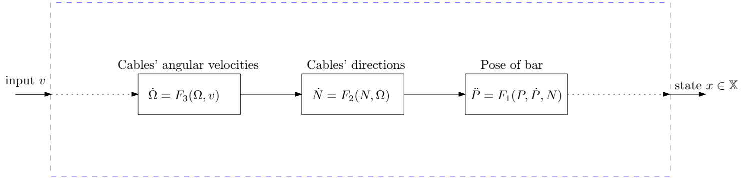
$$\dot x = Dg_3(z)\dot z = Dg_3(z)Z(z,u)\Big|_{\substack{z=(\text{state transformation})^{-1}\\ u=\text{input transformation}}} = X_3(x,v)$$

$$X_3(x,v) = \left[\begin{array}{c} X_2 \\ \hdashline \dot\omega_1 = \Pi(n_1)\tau_1 \\ \dot\omega_2 = \Pi(n_2)\tau_2 \end{array}\right] = \left[\begin{array}{c} X_1 \\ \hdashline \dot n_1 = \mathcal{S}(n_1)\omega_1 \\ \dot n_2 = \mathcal{S}(n_2)\omega_2 \\ \hdashline \\ \end{array}\right] = \left[\begin{array}{l} \dot p = v \\ \dot v = \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \dot n = \mathcal{S}(\omega) n \\ \dot\omega = \mathcal{S}(n)\frac{d_1 T_1 n_1 + d_2 T_2 n_2}{J} \\ \hdashline \\ \hdashline \\ \end{array}\right]$$



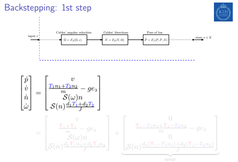
- transformed input $v = (T_1, T_2, \tau_1, \tau_2)$
- The choice of input and state transformations is now clear: it induces a cascaded structure with three layers, which can be explored in the control design process.

Backstepping: 1st step



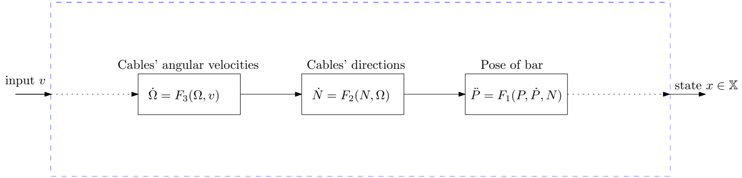
$$
\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n) \frac{d_1 T_1 + d_2 T_2}{J} \end{bmatrix}
$$

$$
= \begin{bmatrix} v \\ \frac{\mathbf{T}_1 + \mathbf{T}_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n) \frac{d_1 \mathbf{T}_1 n_1 + d_2 \mathbf{T}_2 n_2}{J} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{\mathbf{T}_1 - T_1 n_1 + \mathbf{T}_2 - T_2 n_2}{m} - g e_3 \\ 0 \\ \mathcal{S}(n) \frac{d_1 (\mathbf{T}_1 - T_1 n_1) + d_2 (\mathbf{T}_2 - T_2 n_2)}{J} \end{bmatrix}}_{\text{error}}
$$



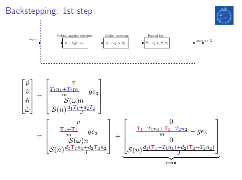
•

Backstepping: 1st step



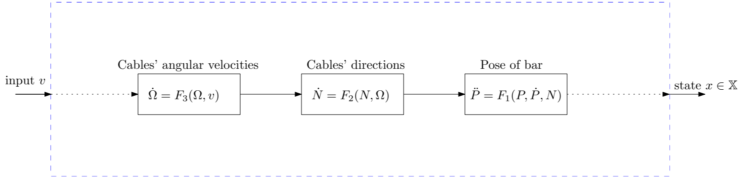
$$\begin{bmatrix}\dot{p}\\ \dot{v}\\ \dot{n}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} v\\ \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n)\frac{d_1 T_1 + d_2 T_2}{J}\end{bmatrix}$$

$$= \begin{bmatrix} v\\ \frac{\mathbf{T}_1 + \mathbf{T}_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n)\frac{d_1 \mathbf{T}_1 n_1 + d_2 \mathbf{T}_2 n_2}{J}\end{bmatrix} + \underbrace{\begin{bmatrix} 0\\ \frac{\mathbf{T}_1 - T_1 n_1 + \mathbf{T}_2 - T_2 n_2}{m} - g e_3 \\ 0 \\ \mathcal{S}(n)\frac{d_1(\mathbf{T}_1 - T_1 n_1) + d_2(\mathbf{T}_2 - T_2 n_2)}{J}\end{bmatrix}}_{\text{error}}$$



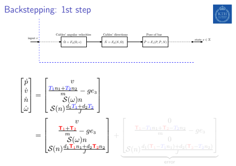
-

Backstepping: 1st step



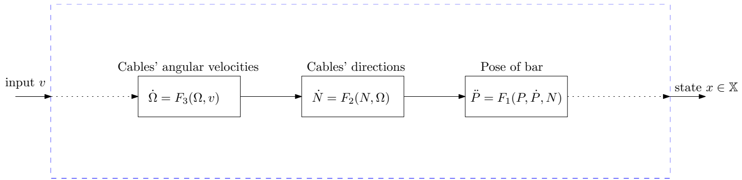
$$
\begin{bmatrix} \dot p \\ \dot v \\ \dot n \\ \dot\omega \end{bmatrix}
=
\begin{bmatrix} v \\ \frac{T_1 n_1 + T_2 n_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n)\frac{d_1 T_1 + d_2 T_2}{J} \end{bmatrix}
$$

$$
=
\begin{bmatrix} v \\ \frac{\mathbf{T}_1 + \mathbf{T}_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n)\frac{d_1 \mathbf{T}_1 n_1 + d_2 \mathbf{T}_2 n_2}{J} \end{bmatrix}
+
\underbrace{\begin{bmatrix} 0 \\ \frac{\mathbf{T}_1 - T_1 n_1 + \mathbf{T}_2 - T_2 n_2}{m} - g e_3 \\ 0 \\ \mathcal{S}(n)\frac{d_1(\mathbf{T}_1 - T_1 n_1) + d_2(\mathbf{T}_2 - T_2 n_2)}{J} \end{bmatrix}}_{\text{error}}
$$



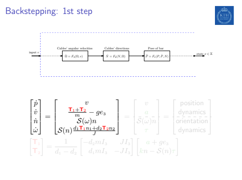
•

Backstepping: 1st step



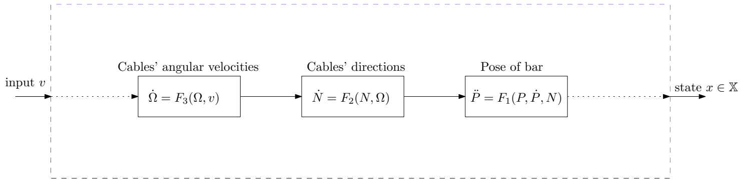
$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ \frac{\mathbf{T}_1 + \mathbf{T}_2}{m} - g e_3 \\ \mathcal{S}(\omega) n \\ \mathcal{S}(n) \frac{d_1 \mathbf{T}_1 n_1 + d_2 \mathbf{T}_2 n_2}{J} \end{bmatrix} = \begin{bmatrix} v \\ a \\ \mathcal{S}(\omega) n \\ \tau \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{dynamics} \\ \text{orientation} \\ \text{dynamics} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \frac{1}{d_1 - d_2} \begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3 \end{bmatrix} \begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n) \tau \end{bmatrix}$$



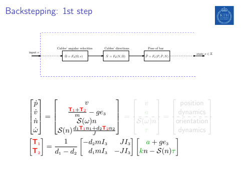
- Two decoupled systems
- Another input transformation
- k is a degree of freedom in the sense that it does not affect linear or angular acceleration

Backstepping: 1st step



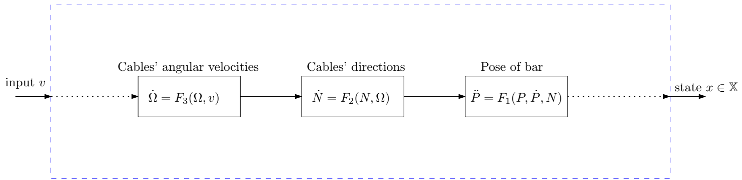
$$\begin{bmatrix}\dot p\\ \dot v\\ \dot n\\ \dot\omega\end{bmatrix} = \begin{bmatrix} v\\ \frac{\mathsf{T}_1+\mathsf{T}_2}{m} - g e_3 \\ \mathcal{S}(\omega)n \\ \mathcal{S}(n)\frac{d_1 \mathsf{T}_1 n_1 + d_2 \mathsf{T}_2 n_2}{J}\end{bmatrix} = \begin{bmatrix} v\\ a\\ \mathcal{S}(\omega)n\\ \tau\end{bmatrix} = \begin{bmatrix}\text{position}\\ \text{dynamics}\\ \text{orientation}\\ \text{dynamics}\end{bmatrix}$$

$$\begin{bmatrix}\mathsf{T}_1\\ \mathsf{T}_2\end{bmatrix} = \frac{1}{d_1 - d_2}\begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3\end{bmatrix}\begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n)\tau\end{bmatrix}$$



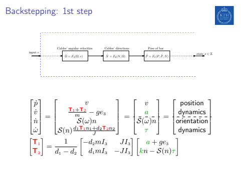
- Two decoupled systems
- Another input transformation
- k is a degree of freedom in the sense that it does not affect linear or angular acceleration

Backstepping: 1st step



$$\begin{bmatrix}\dot p\\ \dot v\\ \dot n\\ \dot\omega\end{bmatrix} = \begin{bmatrix} v\\ \frac{\mathsf{T}_1+\mathsf{T}_2}{m}-ge_3\\ \mathcal{S}(\omega)n\\ \mathcal{S}(n)\frac{d_1 \mathsf{T}_1 n_1 + d_2 \mathsf{T}_2 n_2}{J}\end{bmatrix} = \left[\begin{array}{c} v\\ a\\ \hline \mathcal{S}(\omega)n\\ \tau\end{array}\right] = \left[\begin{array}{c}\text{position}\\ \text{dynamics}\\ \hline \text{orientation}\\ \text{dynamics}\end{array}\right]$$

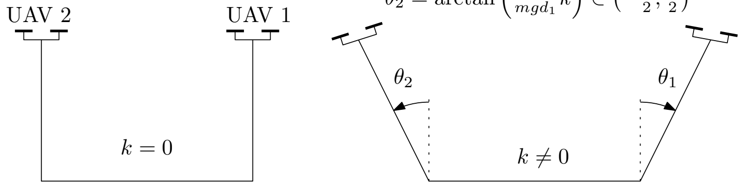
$$\begin{bmatrix}\mathsf{T}_1\\ \mathsf{T}_2\end{bmatrix} = \frac{1}{d_1 - d_2}\begin{bmatrix}-d_2 m I_3 & J I_3\\ d_1 m I_3 & -J I_3\end{bmatrix}\begin{bmatrix} a + ge_3\\ kn - \mathcal{S}(n)\tau\end{bmatrix}$$



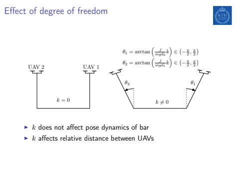
- Two decoupled systems
- Another input transformation
- k is a degree of freedom in the sense that it does not affect linear or angular acceleration

Effect of degree of freedom

$$\theta_1 = \arctan\left(\frac{J}{mgd_2}k\right) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$$
$$\theta_2 = \arctan\left(\frac{J}{mgd_1}k\right) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$$



- k does not affect pose dynamics of bar
- k affects relative distance between UAVs



- Input k **does not affect pose dynamics of bar**
- Input k only affects relative distance between UAVs

Backstepping: 1st step

- Tensions that minimize error

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} n_1^T \mathbf{T}_1 \\ n_2^T \mathbf{T}_2 \end{bmatrix}$$

- Pose dynamics with chosen tensions

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ a \\ \bar{S}(\bar{\omega})\bar{n} \\ \tau \end{bmatrix} + \sum \begin{bmatrix} 0 \\ \|\mathbf{T}_i\| \frac{1}{m} \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \\ 0 \\ \|\mathbf{T}_i\| \frac{d_i}{J} S(n) \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \end{bmatrix}$$

- Lyapunov function for backstepping

$$V_{\text{step 1}} = k_{\text{linear}} V_{\text{linear}} + k_{\text{angular}} V_{\text{angular}}$$

$$V_{\text{step 2}} = V_{\text{step 1}} + \gamma_1 \left(1 - n_1^T \frac{\mathbf{T}_1}{\|\mathbf{T}_1\|}\right) + \gamma_2 \left(1 - n_2^T \frac{\mathbf{T}_2}{\|\mathbf{T}_2\|}\right)$$

Backstepping: 1st step
- Tensions that minimize error

- Input k **does not affect pose dynamics of bar**
- Input k only affects relative distance between UAVs

Backstepping: 1st step

- Tensions that minimize error

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} n_1^T \mathbf{T}_1 \\ n_2^T \mathbf{T}_2 \end{bmatrix}$$

- Pose dynamics with chosen tensions

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ a \\ \mathcal{S}(\omega)n \\ \tau \end{bmatrix} + \sum \begin{bmatrix} 0 \\ \|\mathbf{T}_i\| \frac{1}{m} \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \\ 0 \\ \|\mathbf{T}_i\| \frac{d_i}{J} \mathcal{S}(n) \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \end{bmatrix}$$

- Lyapunov function for backstepping

$$V_{\text{step 1}} = k_{\text{linear}} V_{\text{linear}} + k_{\text{angular}} V_{\text{angular}}$$

$$V_{\text{step 2}} = V_{\text{step 1}} + \gamma_1 \left(1 - n_1^T \frac{\mathbf{T}_1}{\|\mathbf{T}_1\|}\right) + \gamma_2 \left(1 - n_2^T \frac{\mathbf{T}_2}{\|\mathbf{T}_2\|}\right)$$

- Input k **does not affect pose dynamics of bar**
- Input k only affects relative distance between UAVs

Backstepping: 1st step

- Tensions that minimize error

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} n_1^T \mathbf{T}_1 \\ n_2^T \mathbf{T}_2 \end{bmatrix}$$

- Pose dynamics with chosen tensions

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ a \\ \bar{\mathcal{S}}(\bar{\omega})\bar{n} \\ \tau \end{bmatrix} + \sum \begin{bmatrix} 0 \\ \|\mathbf{T}_i\| \frac{1}{m} \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \\ 0 \\ \|\mathbf{T}_i\| \frac{d_i}{J} \mathcal{S}(n) \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \end{bmatrix}$$

- Lyapunov function for backstepping

$$V_{\text{step 1}} = k_{\text{linear}} V_{\text{linear}} + k_{\text{angular}} V_{\text{angular}}$$

$$V_{\text{step 2}} = V_{\text{step 1}} + \gamma_1 \left(1 - n_1^T \frac{\mathbf{T}_1}{\|\mathbf{T}_1\|}\right) + \gamma_2 \left(1 - n_2^T \frac{\mathbf{T}_2}{\|\mathbf{T}_2\|}\right)$$

- Input k **does not affect pose dynamics of bar**
- Input k only affects relative distance between UAVs

Backstepping: 1st step

- Tensions that minimize error

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} n_1^T \mathbf{T}_1 \\ n_2^T \mathbf{T}_2 \end{bmatrix}$$

- Pose dynamics with chosen tensions

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ a \\ \bar{\mathcal{S}}(\bar{\omega})\bar{n} \\ \tau \end{bmatrix} + \sum \begin{bmatrix} 0 \\ \|\mathbf{T}_i\| \frac{1}{m} \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \\ 0 \\ \|\mathbf{T}_i\| \frac{d_i}{J} \mathcal{S}(n) \Pi(n_i) \frac{\mathbf{T}_i}{\|\mathbf{T}_i\|} \end{bmatrix}$$

- Lyapunov function for backstepping

$$V_{\text{step 1}} = k_{\text{linear}} V_{\text{linear}} + k_{\text{angular}} V_{\text{angular}}$$

$$V_{\text{step 2}} = V_{\text{step 1}} + \gamma_1 \left(1 - n_1^T \frac{\mathbf{T}_1}{\|\mathbf{T}_1\|}\right) + \gamma_2 \left(1 - n_2^T \frac{\mathbf{T}_2}{\|\mathbf{T}_2\|}\right)$$

- Input k **does not affect pose dynamics of bar**
- Input k only affects relative distance between UAVs

Backstepping: 1st step

- Problem: $\|\mathbf{T}_1\| \neq 0$ and $\|\mathbf{T}_2\| \neq 0$

$$\begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \frac{1}{d_1 - d_2} \begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3 \end{bmatrix} \begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n)\tau \end{bmatrix}$$

- No problem if

$$\|a\| + \frac{J}{m \min(|d_1|, |d_2|)} (\|\tau\| + |k|) < g$$

- Control laws for linear and angular position control must be bounded

- a is the linear acceleration, and τ is the angular acceleration
- Control law for position control must be bounded
- Control law for orientation control must also bounded
- Feasible desired pose trajectory if

$$\|\ddot{p}^\star\| + \frac{J}{m \min(|d_1|, |d_2|)} (\|\ddot{n}^\star\| + |k^\star|) < g$$

- relate with equilibrium trajectory described before

Backstepping: 1st step

- Problem: $\|\mathbf{T}_1\| \neq 0$ and $\|\mathbf{T}_2\| \neq 0$

$$\begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \frac{1}{d_1 - d_2} \begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3 \end{bmatrix} \begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n)\tau \end{bmatrix}$$

- No problem if

$$\|a\| + \frac{J}{m \min(|d_1|, |d_2|)}(\|\tau\| + |k|) < g$$

- Control laws for linear and angular position control must be bounded

- a is the linear acceleration, and τ is the angular acceleration
- Control law for position control must be bounded
- Control law for orientation control must also bounded
- Feasible desired pose trajectory if

$$\|\ddot{p}^\star\| + \frac{J}{m \min(|d_1|, |d_2|)}(\|\ddot{n}^\star\| + |k^\star|) < g$$

- relate with equilibrium trajectory described before

Backstepping: 1st step

- Problem: $\|\mathbf{T}_1\| \neq 0$ and $\|\mathbf{T}_2\| \neq 0$

$$\begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \frac{1}{d_1 - d_2} \begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3 \end{bmatrix} \begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n) \tau \end{bmatrix}$$

- No problem if

$$\|a\| + \frac{J}{m \min(|d_1|, |d_2|)} (\|\tau\| + |k|) < g$$

- Control laws for linear and angular position control must be bounded

Backstepping: 1st step

- Problem: $\|\mathbf{T}_1\| \neq 0$ and $\|\mathbf{T}_2\| \neq 0$

$$\begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} = \frac{1}{d_1 - d_2} \begin{bmatrix} -d_2 m I_3 & J I_3 \\ d_1 m I_3 & -J I_3 \end{bmatrix} \begin{bmatrix} a + g e_3 \\ k n - \mathcal{S}(n) \tau \end{bmatrix}$$

- No problem if

$$\|a\| + \frac{J}{m \min(|d_1|, |d_2|)} (\|\tau\| + |k|) < g$$

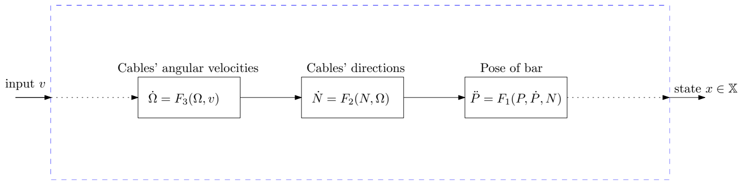
- Control laws for linear and angular position control must be bounded

- a is the linear acceleration, and τ is the angular acceleration
- Control law for position control must be bounded
- Control law for orientation control must also bounded
- Feasible desired pose trajectory if

$$\|\ddot{p}^\star\| + \frac{J}{m \min(|d_1|, |d_2|)} (\|\ddot{n}^\star\| + |k^\star|) < g$$

- relate with equilibrium trajectory described before

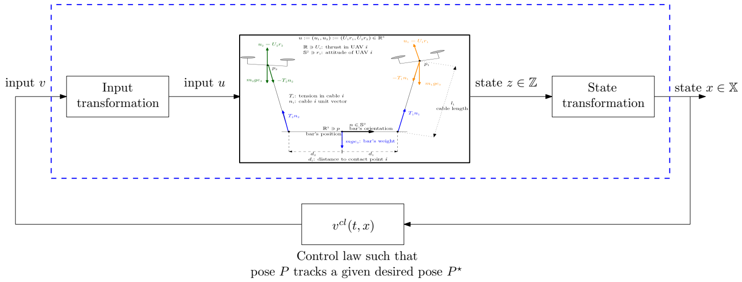
Backstepping: 2nd and third steps



- 2nd step: design cables angular velocities that 1st steer error to zero
- 3rd step: design cables angular accelerations that 2nd steer error to zero

- 2nd and 3rd backstepping steps

Result



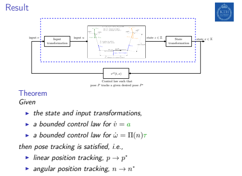
Theorem

Given

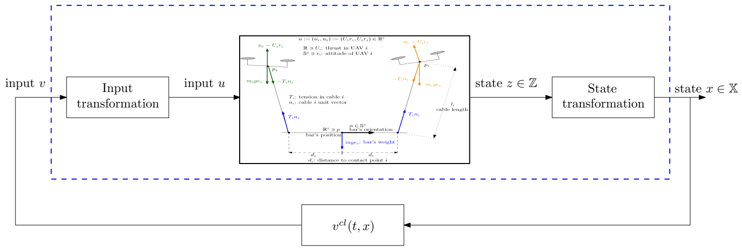
- *the state and input transformations,*
- *a bounded control law for* $\dot{v} = a$
- *a bounded control law for* $\dot{\omega} = \Pi(n)\tau$

then pose tracking is satisfied, i.e.,

- *linear position tracking,* $p \to p^\star$
- *angular position tracking,* $n \to n^\star$

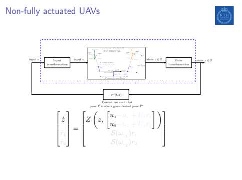


Non-fully actuated UAVs



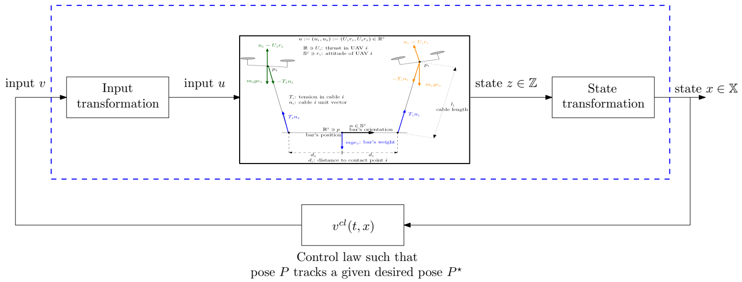
Control law such that
pose P tracks a given desired pose $P^\star$

$$\begin{bmatrix} \dot{z} \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \begin{bmatrix} Z\left(z, \begin{bmatrix} u_1 - u_1 + U_1 r_1 \\ u_2 - u_2 + U_2 r_2 \end{bmatrix}\right) \\ S(\omega_{r_1}) r_1 \\ S(\omega_{r_2}) r_2 \end{bmatrix}$$

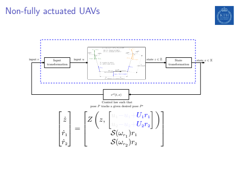


- Recall that Z is input affine

Non-fully actuated UAVs

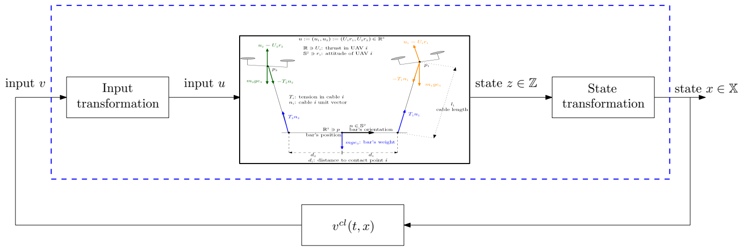


$$\begin{bmatrix} \dot{z} \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \begin{bmatrix} Z\left(z, \begin{bmatrix} u_1 - u_1 + U_1 r_1 \\ u_2 - u_2 + U_2 r_2 \end{bmatrix}\right) \\ \mathcal{S}(\omega_{r_1}) r_1 \\ \mathcal{S}(\omega_{r_2}) r_2 \end{bmatrix}$$



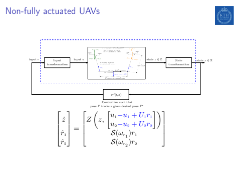
- Recall that Z is input affine

Non-fully actuated UAVs



Control law such that
pose P tracks a given desired pose $P^\star$

$$\begin{bmatrix} \dot{z} \\ \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \begin{bmatrix} Z\left(z, \begin{bmatrix} u_1 - u_1 + U_1 r_1 \\ u_2 - u_2 + U_2 r_2 \end{bmatrix}\right) \\ \mathcal{S}(\omega_{r_1}) r_1 \\ \mathcal{S}(\omega_{r_2}) r_2 \end{bmatrix}$$



- Recall that Z is input affine

Non-fully actuated UAVs

$$
\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \\ \dot{n}_1 \\ \dot{n}_2 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ 0 \\ 0 \end{bmatrix}}_{\text{cascaded structure}} + \begin{bmatrix} 0 \\ \sum B_{T_1,i}(U_i r_i - u_i) \\ 0 \\ \sum B_{T_2,i}(U_i r_i - u_i) \\ 0 \\ 0 \\ \sum B_{\tau_1,i}(U_i r_i - u_i) \\ \sum B_{\tau_2,i}(U_i r_i - u_i) \\ S(\omega_{r_1}) r_1 \\ S(\omega_{r_2}) r_2 \end{bmatrix}
$$

Option 1:

- Minimize the error $\|U_i r_i - u_i\| \Rightarrow U_i = r_i^T u_i$
- Not a good option because it breaks cascaded structure

- Recall that Z is input affine
- For option 1, backstepping cannot be use because cascaded structure does not exist
- We cannot pursue backstepping with this approach because there is no cascaded system!

Non-fully actuated UAVs

$$
\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \\ \dot{n}_1 \\ \dot{n}_2 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix}
=
\underbrace{\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \hdashline \cdots \\ \cdots \\ \hdashline \cdots \\ \cdots \\ \hdashline 0 \\ 0 \end{bmatrix}}_{\text{cascaded structure}}
+
\begin{bmatrix} 0 \\ \sum B_{T_1,i}({\color{blue}U_i r_i - u_i}) \\ 0 \\ \sum B_{T_2,i}({\color{blue}U_i r_i - u_i}) \\ \hdashline 0 \\ 0 \\ \hdashline \sum B_{\tau_1,i}({\color{blue}U_i r_i - u_i}) \\ \sum B_{\tau_2,i}({\color{blue}U_i r_i - u_i}) \\ \hdashline \mathcal{S}(\omega_{r_1}) r_1 \\ \mathcal{S}(\omega_{r_2}) r_2 \end{bmatrix}
$$

Option 1:

- Minimize the error $\|U_i r_i - u_i\| \Rightarrow U_i = r_i^T u_i$
- Not a good option because it breaks cascaded structure

Non-fully actuated UAVs

- Recall that Z is input affine
- For option 1, backstepping cannot be use because cascaded structure does not exist
- We cannot pursue backstepping with this approach because there is no cascaded system!

Non-fully actuated UAVs

$$
\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \\ \dot{n}_1 \\ \dot{n}_2 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix}
=
\underbrace{\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \hdashline \cdots \\ \cdots \\ \hdashline \cdots \\ \cdots \\ \hdashline 0 \\ 0 \end{bmatrix}}_{\text{cascaded structure}}
+
\begin{bmatrix} 0 \\ \sum B_{T_1,i}(U_i r_i - u_i) \\ 0 \\ \sum B_{T_2,i}(U_i r_i - u_i) \\ \hdashline 0 \\ 0 \\ \hdashline \sum B_{\tau_1,i}(U_i r_i - u_i) \\ \sum B_{\tau_2,i}(U_i r_i - u_i) \\ \hdashline \mathcal{S}(\omega_{r_1}) r_1 \\ \mathcal{S}(\omega_{r_2}) r_2 \end{bmatrix}
$$

Option 1:

- Minimize the error $\|U_i r_i - u_i\| \Rightarrow U_i = r_i^T u_i$
- Not a good option because it breaks cascaded structure

Non-fully actuated UAVs
Option 1:
Minimize the error $\|U_i r_i - u_i\| \Rightarrow U_i = r_i^T u_i$
Not a good option because it breaks cascaded structure

- Recall that Z is input affine
- For option 1, backstepping cannot be use because cascaded structure does not exist
- We cannot pursue backstepping with this approach because there is no cascaded system!

Non-fully actuated UAVs

$$
\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{n} \\ \dot{\omega} \\ \dot{n}_1 \\ \dot{n}_2 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{r}_1 \\ \dot{r}_2 \end{bmatrix}
=
\underbrace{\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \hline \cdots \\ \cdots \\ \hline \cdots \\ \cdots \\ \hline 0 \\ 0 \end{bmatrix}}_{\text{cascaded structure}}
+
\begin{bmatrix} 0 \\ \cancel{\sum B_{T_1,i}(U_i r_i - u_i)} \\ 0 \\ \cancel{\sum B_{T_2,i}(U_i r_i - u_i)} \\ \hline 0 \\ 0 \\ \hline \sum B_{\tau_1,i}(U_i r_i - u_i) \\ \sum B_{\tau_2,i}(U_i r_i - u_i) \\ \hline \mathcal{S}(\omega_{r_1}) r_1 \\ \mathcal{S}(\omega_{r_2}) r_2 \end{bmatrix}
$$

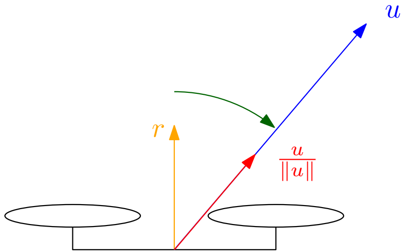
Option 2:

- Preserve cascaded structure: $U_i = \dfrac{n_i^T u_i}{n_i^T r_i}$
- Design UAVs angular velocities $\Rightarrow$ 4th backstepping step

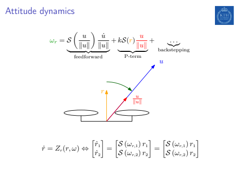
- Recall that Z is input affine
- $\omega_{r_1}, \omega_{r_1}$: UAVs angular velocities
- We can pursue a 4th backstepping step because cascaded structure exists

Attitude dynamics

$$\omega_r = \underbrace{\mathcal{S}\left(\frac{u}{\|u\|}\right)\frac{\dot{u}}{\|u\|}}_{\text{feedforward}} + \underbrace{k\mathcal{S}(r)\frac{u}{\|u\|}}_{\text{P-term}} + \underbrace{\cdots}_{\text{backstepping}}$$



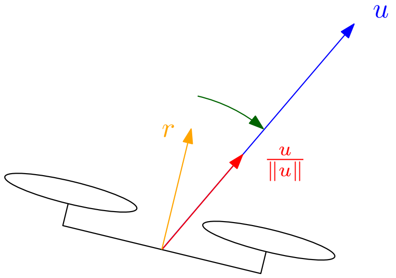
$$\dot{r} = Z_r(r,\omega) \Leftrightarrow \begin{bmatrix} \dot{r}_1 \\ \dot{r}_2 \end{bmatrix} = \begin{bmatrix} \mathcal{S}\left(\omega_{r,1}\right) r_1 \\ \mathcal{S}\left(\omega_{r,2}\right) r_2 \end{bmatrix} = \begin{bmatrix} \mathcal{S}\left(\omega_{r,1}\right) r_1 \\ \mathcal{S}\left(\omega_{r,2}\right) r_2 \end{bmatrix}$$



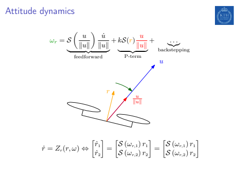
- Attitude inner loop: makes the UAV track the desired attitude
- Desired attitude is given by the $\frac{u}{\|u\|}$

Attitude dynamics

$$\omega_r = \underbrace{\mathcal{S}\left(\frac{u}{\|u\|}\right)\frac{\dot{u}}{\|u\|}}_{\text{feedforward}} + \underbrace{k\mathcal{S}(r)\frac{u}{\|u\|}}_{\text{P-term}} + \underbrace{\cdots}_{\text{backstepping}}$$



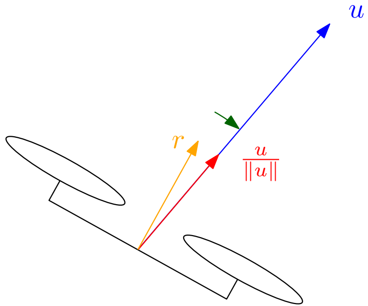
$$\dot{r} = Z_r(r,\omega) \Leftrightarrow \begin{bmatrix}\dot{r}_1\\ \dot{r}_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1\\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1\\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix}$$



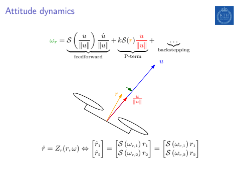
- Attitude inner loop: makes the UAV track the desired attitude
- Desired attitude is given by the $\frac{u}{\|u\|}$

Attitude dynamics

$$\omega_r = \underbrace{\mathcal{S}\left(\frac{u}{\|u\|}\right)\frac{\dot{u}}{\|u\|}}_{\text{feedforward}} + \underbrace{k\mathcal{S}(r)\frac{u}{\|u\|}}_{\text{P-term}} + \underbrace{\cdots}_{\text{backstepping}}$$



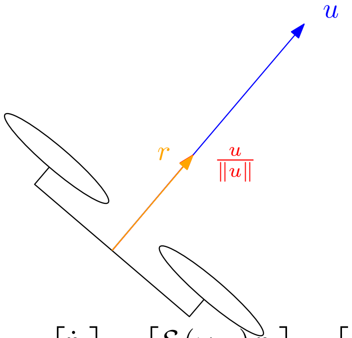
$$\dot{r} = Z_r(r,\omega) \Leftrightarrow \begin{bmatrix}\dot{r}_1\\ \dot{r}_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1\\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1\\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix}$$



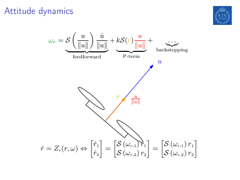
- Attitude inner loop: makes the UAV track the desired attitude
- Desired attitude is given by the $\frac{u}{\|u\|}$

Attitude dynamics

$$\omega_r = \underbrace{\mathcal{S}\left(\frac{u}{\|u\|}\right)\frac{\dot{u}}{\|u\|}}_{\text{feedforward}} + \underbrace{k\mathcal{S}(r)\frac{u}{\|u\|}}_{\text{P-term}} + \underbrace{\cdots}_{\text{backstepping}}$$

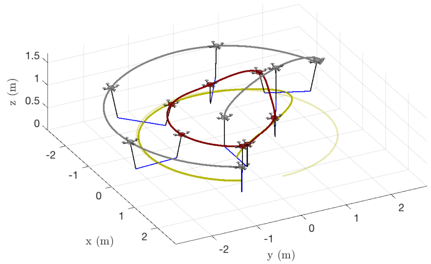


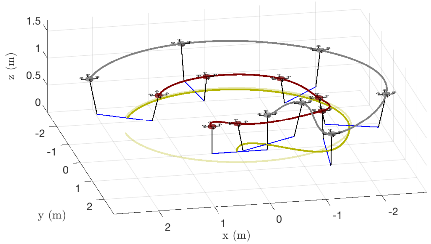
$$\dot{r} = Z_r(r,\omega) \Leftrightarrow \begin{bmatrix}\dot{r}_1 \\ \dot{r}_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1 \\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}\left(\omega_{r,1}\right) r_1 \\ \mathcal{S}\left(\omega_{r,2}\right) r_2\end{bmatrix}$$

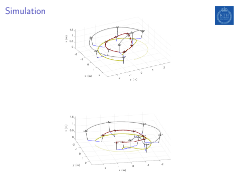


- Attitude inner loop: makes the UAV track the desired attitude
- Desired attitude is given by the $\frac{u}{\|u\|}$

Simulation

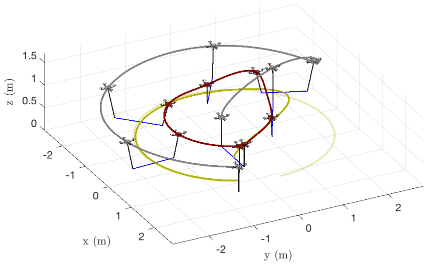


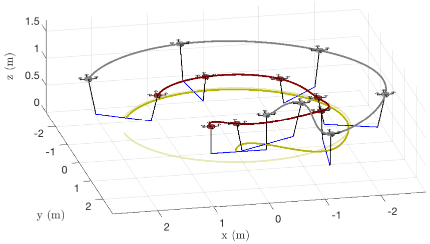


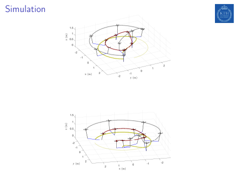


- Desired pose trajectory:
- Physical parameters listed in the article
- degree of freedom $k \neq 0$: cables are not vertical in static equilibrium

Simulation

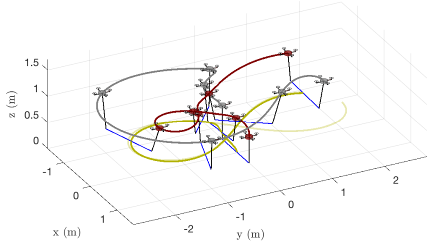


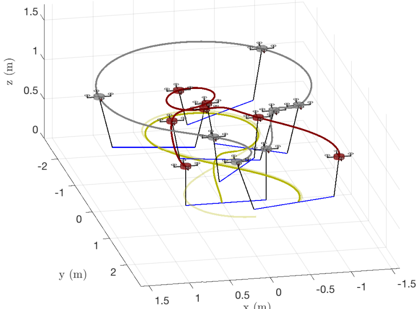


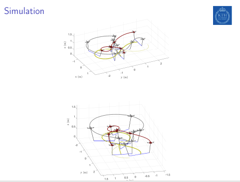


- Desired pose trajectory: circular motion in horizontal place
- Physical parameters listed in the article
- degree of freedom $k \neq 0$: cables are not vertical in static equilibrium
- Full lines – real system. Transparent lines – desired system

Simulation

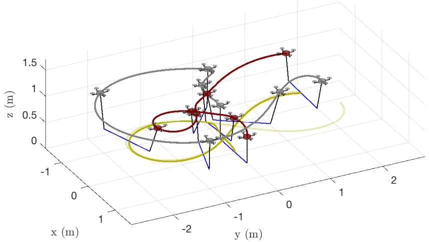


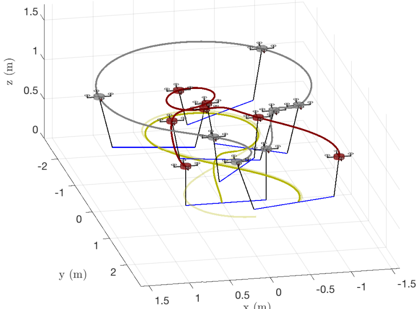


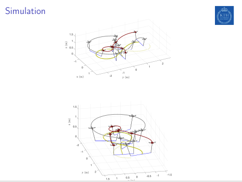


- Desired pose trajectory: eight-like trajectory in horizontal plane
- Physical parameters listed in the article
- degree of freedom $k \neq 0$: cables are not vertical in static equilibrium

Simulation

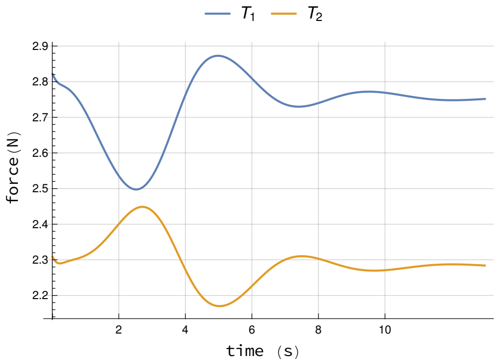




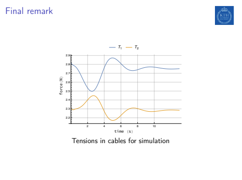


- Desired pose trajectory: eight-like trajectory in horizontal plane
- Physical parameters listed in the article
- degree of freedom $k \neq 0$: cables are not vertical in static equilibrium
- Full lines – real system. Transparent lines – desired system

Final remark



Tensions in cables for simulation



- Tensions on cables are positive: cables need to remain under tension (taut), otherwise model is not valid

Thank you! Questions?

Thank you! Questions?