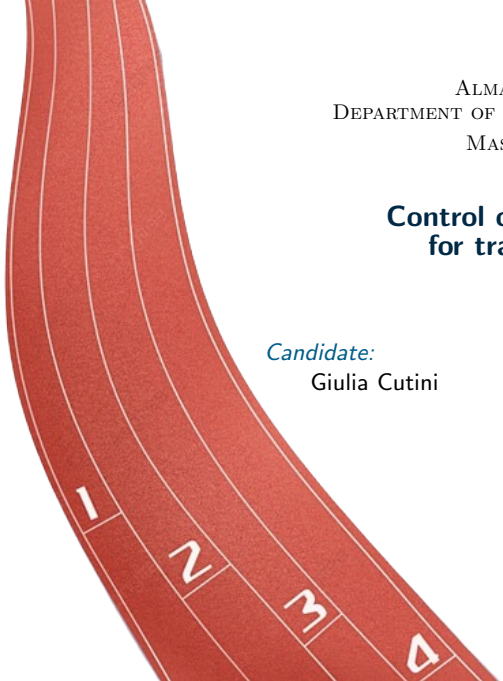


Alma Mater Studiorum Università di Bologna
Department of Electrical, Electronic and Information Engineering
Master's Degree in Automation Engineering

Control of an autonomous aerodynamic airshield for training Olympics 100m sprint athletes

Candidate:
Giulia Cutini

Advisor:
Prof. Giuseppe Notarstefano

Co-Advisors:
Prof. Melanie Zeilinger
Dr. Andrea Carron
Ing. Lorenzo Sforni

Bologna, 18 March 2024

ETH Zürich

Introduction

Motivations

In athletics, the **overspeed training** makes possible to enhance the competition performances.
It can be achieved by isolating the runner from the air resistance using an **airshield**.

Introduction

Motivations

In athletics, the **overspeed training** makes possible to enhance the competition performances.
It can be achieved by isolating the runner from the air resistance using an **airshield**.

The usage of an **autonomous go-kart** improves:

- the safety of the maneuver
- the reliability and the reusability

Introduction

Motivations

In athletics, the **overspeed training** makes possible to enhance the competition performances.
It can be achieved by isolating the runner from the air resistance using an **airshield**.

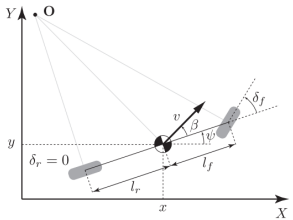
The usage of an **autonomous go-kart** improves:

- the safety of the maneuver
- the reliability and the reusability

Contributions

- Model of the system: the autonomous go-kart with the airshield attached
- Design a controller for the system to regulate it with respect to the runner
- Test the controller

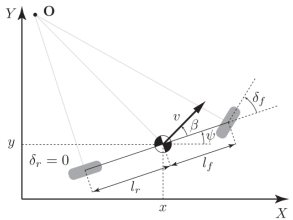
Modeling of the go-kart with the aishield attached



Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

Modeling of the go-kart with the aishield attached

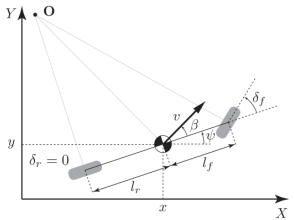


Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The runners are interested in executing only 50-80 meters overspeed training

Modeling of the go-kart with the aishield attached



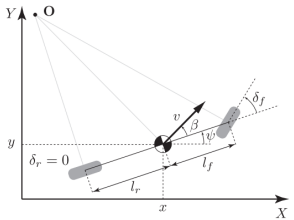
Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The runners are interested in executing only 50-80 meters overspeed training

$\Rightarrow$

Modeling of the go-kart with the aishield attached



Non linear kinematic byclicle model

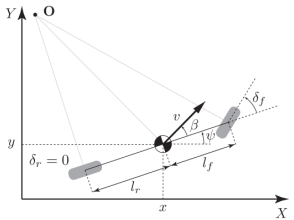
$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The runners are interested in executing only 50-80 meters overspeed training

$\Rightarrow$

The model can be simplified by not considering the steering

Modeling of the go-kart with the aishield attached



Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

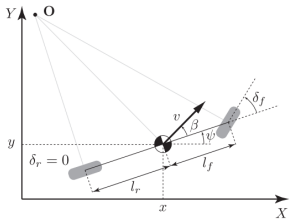
The runners are interested in executing only 50-80 meters overspeed training

$\Rightarrow$

The model can be simplified by not considering the steering

$$\begin{cases} \dot{p}(t) = v(t) \\ \dot{v}(t) = \dfrac{F(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

Modeling of the go-kart with the aishield attached



Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The runners are interested in executing only 50-80 meters overspeed training

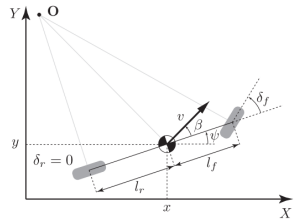
$\Rightarrow$

The model can be simplified by not considering the steering

$$\begin{cases} \dot{p}(t) = v(t) \\ \dot{v}(t) = \dfrac{F(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The non linear term can be removed being minor with respect to the other terms

Modeling of the go-kart with the aishield attached



Non linear kinematic byclicle model

$$\begin{cases} \dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \\ \dot{y}(t) = v(t)\sin(\psi(t) + \beta(t)) \\ \dot{\psi}(t) = \dfrac{v(t)}{l_r}\sin(\beta(t)) \\ \dot{v}(t) = \dfrac{F_x(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) - C_d v^2(t) - C_{roll}) \end{cases}$$

The runners are interested in executing only 50-80 meters overspeed training

$\Rightarrow$

The model can be simplified by not considering the steering

$$\begin{cases} \dot{p}(t) = v(t) \\ \dot{v}(t) = \dfrac{F(t)}{m} = \dfrac{1}{m}(C_{m1}a(t) - C_f v(t) \end{cases}$$

The non linear term can be removed being minor with respect to the other terms

Linear time-invariant system

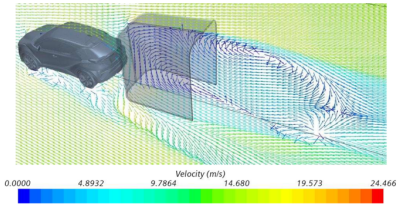
$$x_{k,t+1} = \begin{bmatrix} p_{k,t+1} \\ v_{k,t+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 - dt\frac{C_f}{m} \end{bmatrix} x_{k,t} + \begin{bmatrix} 0 \\ dt\frac{C_{m1}}{m} \end{bmatrix} u_t$$
$$= A\, x_{k,t} + B\, u_t$$

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science "Aerodynamic Shield – new training support technologies"

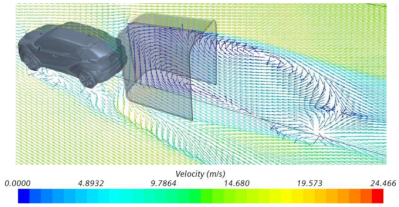


Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science "Aerodynamic Shield – new training support technologies"



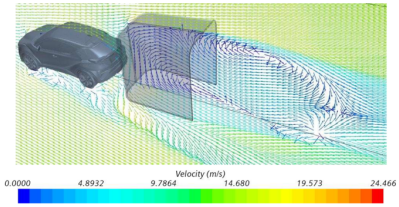
Controller objective

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science "Aerodynamic Shield – new training support technologies"



Controller objective

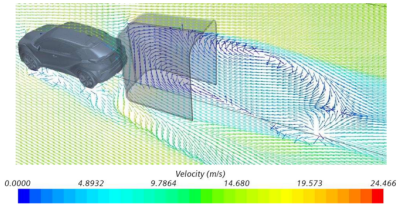
- Maintain the airshield at the reference distance with respect to the runner position

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science "Aerodynamic Shield – new training support technologies"



Controller objective

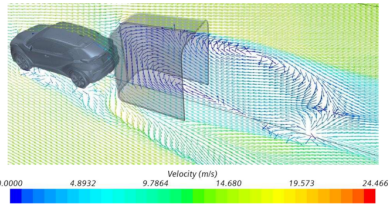
- Maintain the airshield at the reference distance with respect to the runner position
- Regulate the go-kart velocity to match the runner's one

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science "Aerodynamic Shield – new training support technologies"



Controller objective

- Maintain the airshield at the reference distance with respect to the runner position
- Regulate the go-kart velocity to match the runner's one

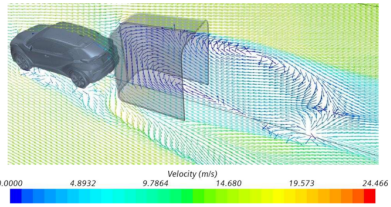
Wheel Encoders
Absolute kart velocity v_k

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science
"Aerodynamic Shield – new training support technologies"



Controller objective

- Maintain the airshield at the reference distance with respect to the runner position
- Regulate the go-kart velocity to match the runner's one

Wheel Encoders
Absolute kart velocity v_k

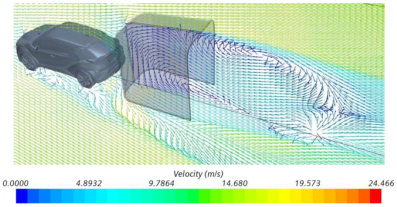
LiDAR (Velodyne)
Relative distance $\Delta p = p_k - p_r$

Application setup

Aerodynamics studies [1] have shown that, by running in the slipstream of a shield:

- Runner results to be isolated from the drag resistance
- A pushing force from behind enhancing the speed
- Reference distance: $d_{\text{des}} = 2.5$ meters

[1] Italian National Olympic Committee (CONI) Institute of Sports Science
"Aerodynamic Shield – new training support technologies"



Controller objective

- Maintain the airshield at the reference distance with respect to the runner position
- Regulate the go-kart velocity to match the runner's one

Wheel Encoders
Absolute kart velocity v_k

LiDAR (Velodyne)
Relative distance $\Delta p = p_k - p_r$

Estimated:

- Relative velocity $\Delta v = v_k - v_r$
- Absolute runner acceleration a_r

Gain Scheduling Linear Quadratic Regulator

$$h_t = \begin{bmatrix} \Delta p_t \\ \Delta v_t \end{bmatrix} = \begin{bmatrix} p_{k,t} - p_{r,t} \\ v_{k,t} - v_{r,t} \end{bmatrix}$$

$$h_{\text{des}} = \begin{bmatrix} d_{\text{des}} \\ 0 \end{bmatrix}$$

- Compute the Linear Quadratic Regulator: K
- Control the system using the state feedback $u^* = -K(h_t - h_{\text{des}})$

Gain Scheduling Linear Quadratic Regulator

$$h_t = \begin{bmatrix} \Delta p_t \\ \Delta v_t \end{bmatrix} = \begin{bmatrix} p_{k,t} - p_{r,t} \\ v_{k,t} - v_{r,t} \end{bmatrix} \qquad h_{\text{des}} = \begin{bmatrix} d_{\text{des}} \\ 0 \end{bmatrix}$$

- Compute the Linear Quadratic Regulator: K
- Control the system using the state feedback $u^* = -K(h_t - h_{\text{des}})$

Gain Scheduling Approach

Catch-up maneuver K_{catch}

Cruise control maneuver K_{cruise}

Gain Scheduling Linear Quadratic Regulator

$$h_t = \begin{bmatrix} \Delta p_t \\ \Delta v_t \end{bmatrix} = \begin{bmatrix} p_{k,t} - p_{r,t} \\ v_{k,t} - v_{r,t} \end{bmatrix} \qquad h_{\text{des}} = \begin{bmatrix} d_{\text{des}} \\ 0 \end{bmatrix}$$

- Compute the Linear Quadratic Regulator: K
- Control the system using the state feedback $u^* = -K(h_t - h_{\text{des}})$

Gain Scheduling Approach

Catch-up maneuver K_{catch}

In the first couple of seconds the runner has an acceleration exceeding the maximum go-kart capabilities.

- Ask the runner to start the sprint at $\Delta p_{\text{init}} = d_{\text{des}} + \bar{d}$
- Control using $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$
- Match the velocities with a safe distance bigger than d_{des}

Cruise control maneuver K_{cruise}

Gain Scheduling Linear Quadratic Regulator

$$h_t = \begin{bmatrix} \Delta p_t \\ \Delta v_t \end{bmatrix} = \begin{bmatrix} p_{k,t} - p_{r,t} \\ v_{k,t} - v_{r,t} \end{bmatrix} \qquad h_{\text{des}} = \begin{bmatrix} d_{\text{des}} \\ 0 \end{bmatrix}$$

- Compute the Linear Quadratic Regulator: K
- Control the system using the state feedback $u^* = -K(h_t - h_{\text{des}})$

Gain Scheduling Approach

Catch-up maneuver K_{catch}

In the first couple of seconds the runner has an acceleration exceeding the maximum go-kart capabilities.

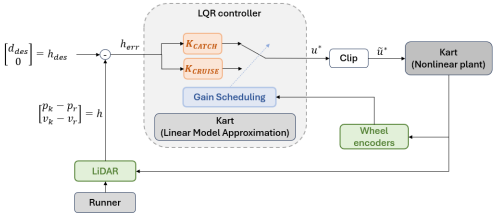
- Ask the runner to start the sprint at $\Delta p_{\text{init}} = d_{\text{des}} + \bar{d}$
- Control using $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$
- Match the velocities with a safe distance bigger than d_{des}

Cruise control maneuver K_{cruise}

Once the runner and go-kart velocities are matched at 80%

- Reduce the distance up to the reference d_{des}
- Control using $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$
- Maintain the desired reference distance
- Maintain to zero the relative velocity

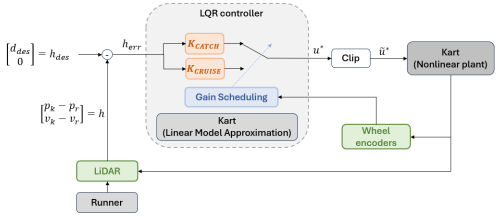
Gain Scheduling Linear Quadratic Regulator



Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Gain Scheduling Linear Quadratic Regulator



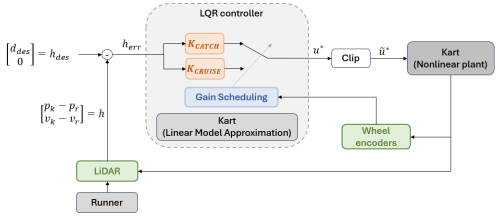
Pros LQR controller

- Reduced computational effort required

Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Gain Scheduling Linear Quadratic Regulator



Pros LQR controller

- Reduced computational effort required

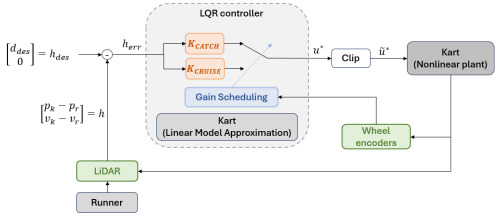
Cons of LQR controller

- Impossibility of directly including constraints

Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Gain Scheduling Linear Quadratic Regulator



Pros LQR controller

- Reduced computational effort required

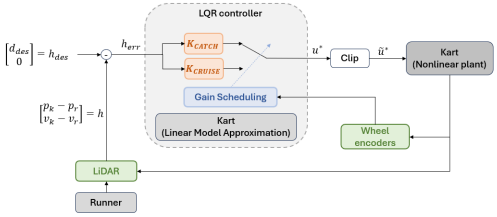
Cons of LQR controller

- Impossibility of directly including constraints
- Clip block required to respect actuator limits $a_{\min} \leq u \leq a_{\max}$

Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Gain Scheduling Linear Quadratic Regulator



Pros LQR controller

- Reduced computational effort required

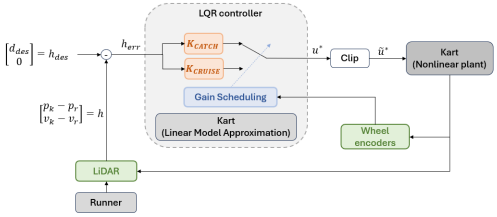
Cons of LQR controller

- Impossibility of directly including constraints
- Clip block required to respect actuator limits $a_{\min} \leq u \leq a_{\max}$
- Impossibility of including state constraint (i.e. $\Delta p \leq d_{\text{safe}}$)

Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Gain Scheduling Linear Quadratic Regulator



Pros LQR controller

- Reduced computational effort required

Cons of LQR controller

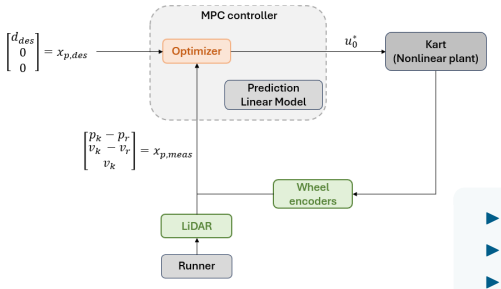
- Impossibility of directly including constraints
- Clip block required to respect actuator limits $a_{\min} \leq u \leq a_{\max}$
- Impossibility of including state constraint (i.e. $\Delta p \leq d_{\text{safe}}$)
- Maintain to zero the relative velocity

Algorithm 1 LQR implementation on hardware system

catched = false;
Set initial runner position with a proper distance $\bar{d}$ from kart;
Define desired system behaviour $h_{\text{des}} = [d_{\text{des}}, 0]^T$;
for $t = 0, 1, 2, \ldots$ **do**;
    Take LiDAR information Δp_t and estimate Δv_t;
    $h_t = [\Delta p_t, \Delta v_t]^T$
    Take absolute kart velocity $v_{k,t}$ from wheel encoders;
    Evaluate absolute runner velocity $v_{r,t} = v_{k,t} - \Delta v_t$;
    if $v_{k,t} \geq 0.8\, v_{r,t}$ **and** catched = false **then**
        catched = true;
    end if
    if catched = false **then**
        $u^* = -K_{\text{catch}}(h_t - h_{\text{des}})$;
    else
        $u^* = -K_{\text{cruise}}(h_t - h_{\text{des}})$;
    end if
    Clip input according to limit saturation limits $\tilde{u}^* = \text{sat}_{[a_{\min}, a_{\max}]}(u^*)$;
    Inject $\tilde{u}^*$ in the plant;
end for

Model Predictive Control

$$\begin{bmatrix}\Delta p_{t+1}\\ \Delta v_{t+1}\\ v_{k,t+1}\end{bmatrix} = \begin{bmatrix}1 & dt & 0\\ 0 & 1 & -dt\frac{C_f}{m}\\ 0 & 0 & 1-dt\frac{C_f}{m}\end{bmatrix} x_{p,t} + \begin{bmatrix}0\\ dt\frac{C_{m1}}{m}\\ dt\frac{C_{m1}}{m}\end{bmatrix} u_t + \begin{bmatrix}0\\ -dta_{r,t}\\ 0\end{bmatrix}$$
$$= A_p\, x_{p,t} + B_p\, u_t + a_t$$



$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{i=t}^{t+N-1} \|(x_{p,i|t} - x_{p,\text{des}})\|_Q + \|u_{i|t}\|_R$$

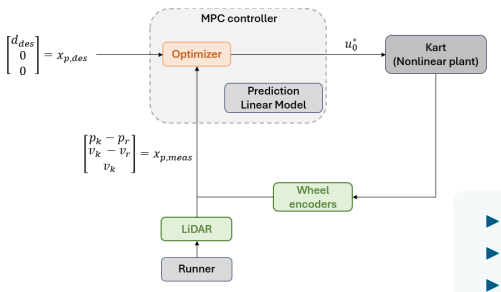
subj. to
$$x_{p,i+1|t} = A_p\, x_{p,i|t} + B_p\, u_{i|t} + a_t$$
$$a_{\min} \le u_{i|t} \le a_{\max}$$
$$d_{\text{safe}} \le \Delta p_{i|t}$$
$$x_{p,t|t} = x_{p,\text{meas}}$$

- Possibility of explicitly including constraints in the optimization
- Possibility of having safety guarantees
- Possibility of including a runner model in the prediction model

Model Predictive Control

$$\begin{bmatrix}\Delta p_{t+1}\\ \Delta v_{t+1}\\ v_{k,t+1}\end{bmatrix} = \begin{bmatrix}1 & dt & 0\\ 0 & 1 & -dt\frac{C_f}{m}\\ 0 & 0 & 1-dt\frac{C_f}{m}\end{bmatrix} x_{p,t} + \begin{bmatrix}0\\ dt\frac{C_{m1}}{m}\\ dt\frac{C_{m1}}{m}\end{bmatrix} u_t + \begin{bmatrix}0\\ -dta_{r,t}\\ 0\end{bmatrix}$$
$$= A_p\, x_{p,t} + B_p\, u_t + a_t$$

- The prediction model is augmented with the kart velocity state



$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{i=t}^{t+N-1} \|(x_{p,i|t} - x_{p,\text{des}})\|_Q + \|u_{i|t}\|_R$$

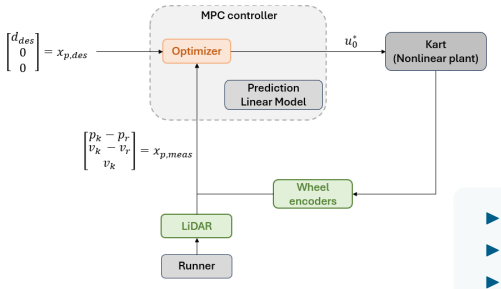
$$\text{subj. to} \quad x_{p,i+1|t} = A_p\, x_{p,i|t} + B_p\, u_{i|t} + a_t$$
$$a_{\min} \le u_{i|t} \le a_{\max}$$
$$d_{\text{safe}} \le \Delta p_{i|t}$$
$$x_{p,t|t} = x_{p,\text{meas}}$$

- Possibility of explicitly including constraints in the optimization
- Possibility of having safety guarantees
- Possibility of including a runner model in the prediction model

Model Predictive Control

$$\begin{bmatrix} \Delta p_{t+1} \\ \Delta v_{t+1} \\ v_{k,t+1} \end{bmatrix} = \begin{bmatrix} 1 & dt & 0 \\ 0 & 1 & -dt\frac{C_f}{m} \\ 0 & 0 & 1 - dt\frac{C_f}{m} \end{bmatrix} x_{p,t} + \begin{bmatrix} 0 \\ dt\frac{C_{m1}}{m} \\ dt\frac{C_{m1}}{m} \end{bmatrix} u_t + \begin{bmatrix} 0 \\ -dta_{r,t} \\ 0 \end{bmatrix}$$
$$= A_p\, x_{p,t} + B_p\, u_t + a_t$$

- The prediction model is augmented with the kart velocity state
- Runner acceleration considered in the predictions



$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{i=t}^{t+N-1} \|(x_{p,i|t} - x_{p,\text{des}})\|_Q + \|u_{i|t}\|_R$$

subj. to
$$x_{p,i+1|t} = A_p\, x_{p,i|t} + B_p\, u_{i|t} + a_t$$
$$a_{\min} \leq u_{i|t} \leq a_{\max}$$
$$d_{\text{safe}} \leq \Delta p_{i|t}$$
$$x_{p,t|t} = x_{p,\text{meas}}$$

- Possibility of explicitly including constraints in the optimization
- Possibility of having safety guarantees
- Possibility of including a runner model in the prediction model

Offset-free Model Predictive Control

Cio

Simulation tests and Controllers comparison

Hardware-in-the-loop tests

Focus on the catch-up maneuver

Hardware-in-the-loop tests

Focus on maintaining a constant distance with the runner having an almost constant velocity