TUTORIAL: MULTI-ROBOT COORDINATION

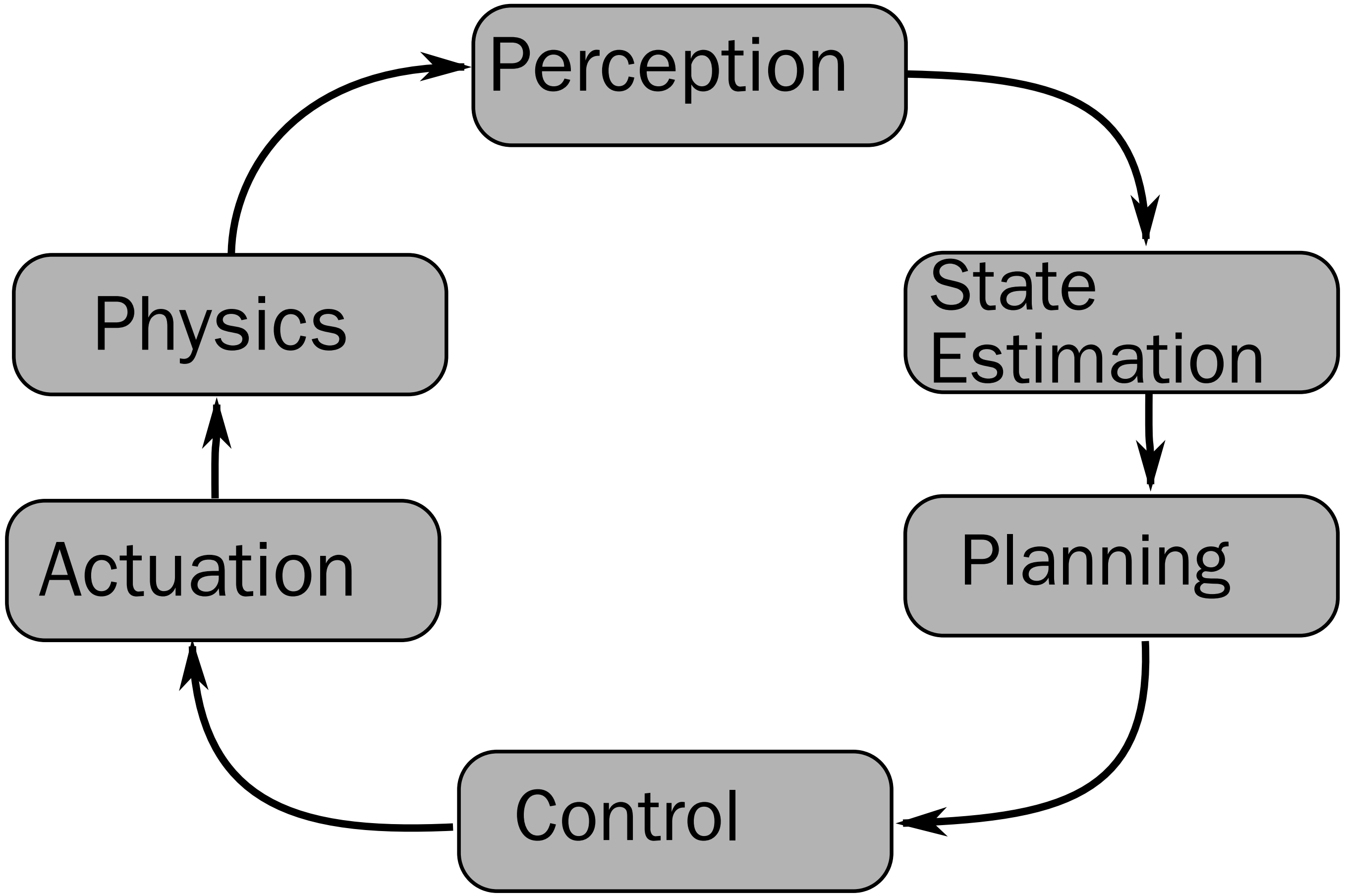
SCIoI & ISAB Summer School 2023 "on Embodied Intelligence – Perception and Learning in Nature and Robotics"

August 21, 2023, Berlin

Wolfgang Hönig (TU Berlin)

MOTIVATION

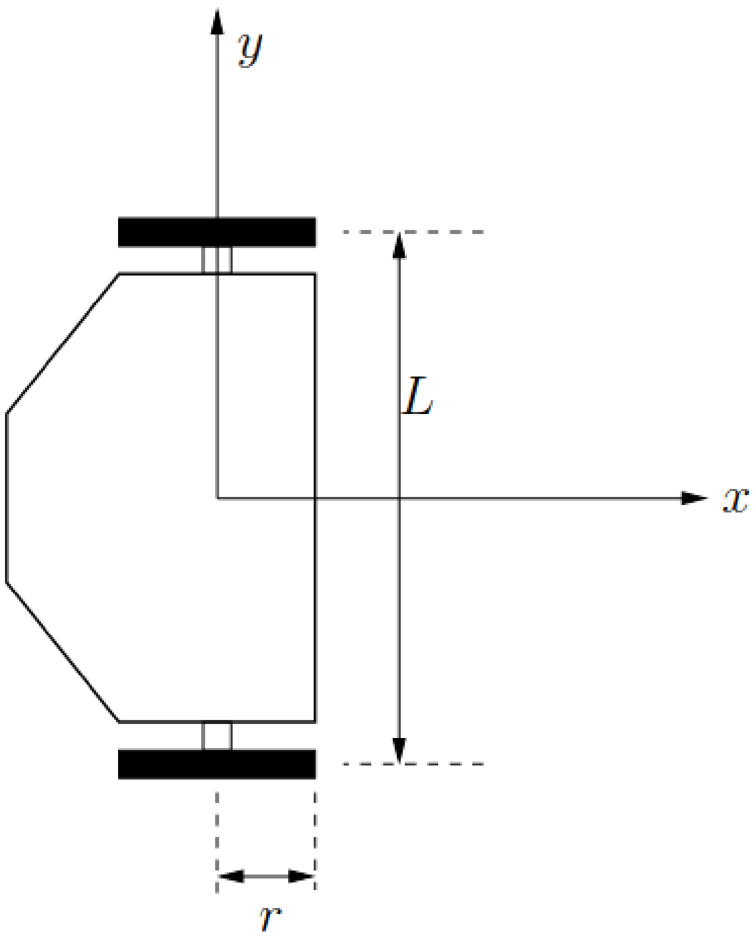
Multi-Robot affects most of these!



OVERVIEW

- Single Robot
  - Physics
  - Controller
  - Differential Flatness & Motion Planning
- Multi-Robot
  - Voronoi Cells
  - Collision Avoidance: Buffered Voronoi Cells

PHYSICS: DIFFERENTIAL DRIVE ROBOT



- State: position and orientation x, y, θ
- Action: Angular speed of wheels u_l, u_r
- Dynamics

$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta$$
$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l)$$

PHYSICS: DIFFERENTIAL DRIVE ROBOT

- Dynamics

$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta$$
$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l)$$

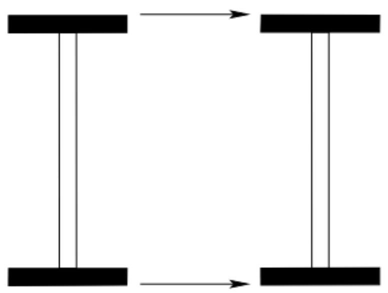
- Substitute actions to $v = \frac{r}{2}(u_l + u_r)$ and $\omega = \frac{r}{L}(u_r - u_l)$

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$

- "original" actions can be easily computed as

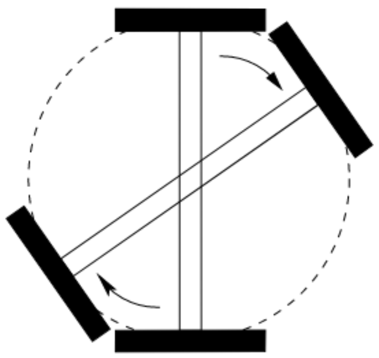
$$u_r = \frac{2v + L\omega}{2r}$$
$$u_l = \frac{2v - L\omega}{2r}$$

PHYSICS: DIFFERENTIAL DRIVE ROBOT



- $u_l = u_r$

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = 0$$



- $u_l = -u_r$

$$\dot{x} = 0$$
$$\dot{y} = 0$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l)$$

DEMO

ROBOT CONTROLLER

- Assume we have desired state x_d, y_d, θ_d and desired v_d, ω_d
- controller has a feedforward term and feedback term (Reference)
- $K_x, K_y, K_\theta \in \mathbb{R}^+$ are tuning gains

$$
\begin{aligned}
x_e &= (x_d - x)\cos\theta + (y_d - y)\sin\theta \\
y_e &= -(x_d - x)\sin\theta + (y_d - y)\cos\theta \\
\theta_e &= \theta_d - \theta \\
v_{ctrl} &= v_d \cos\theta_e + K_x x_e \\
\omega_{ctrl} &= \omega_d + v_d(K_y y_e + K_\theta \sin\theta_e)
\end{aligned}
$$

DEMO

DIFFERENTIAL FLATNESS

- Find a *mapping* from workspace to state space
- If we have a desired 2D smooth curve $p(t)$ (e.g., polynomial), can we compute desired states?

$$\frac{\dot{y}}{\dot{x}} = \frac{\frac{r}{2}(u_l + u_r)\sin\theta}{\frac{r}{2}(u_l + u_r)\cos\theta}$$

$$\frac{\dot{y}}{\dot{x}} = \frac{\sin\theta}{\cos\theta} = \tan\theta$$

$$\Rightarrow \theta = \arctan\frac{\dot{y}}{\dot{x}}$$

DIFFERENTIAL FLATNESS

$$\omega = \dot{\theta} = \frac{d}{dt}\arctan\left(\frac{\dot{y}}{\dot{x}}\right)$$
$$= \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}$$

$$s = \frac{\dot{x}}{\cos\theta}$$
$$= \frac{\dot{x}}{\cos\left(\arctan\left(\frac{\dot{y}}{\dot{x}}\right)\right)}$$
$$= \dot{x}\sqrt{\frac{\dot{y}^2}{\dot{x}^2} + 1} = \dot{x}\sqrt{\frac{\dot{y}^2}{\dot{x}^2} + \frac{\dot{x}^2}{\dot{x}^2}}$$
$$= \pm\sqrt{\dot{y}^2 + \dot{x}^2}$$

DEMO

BÉZIER CURVES

Bézier Curve

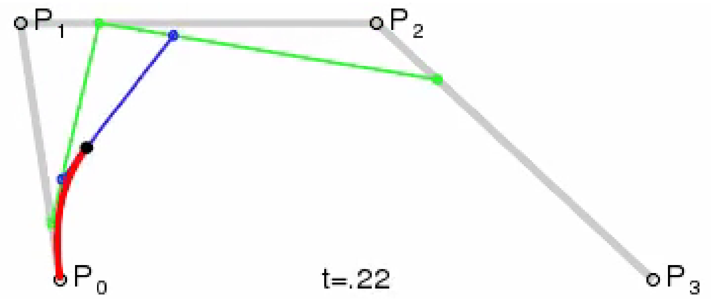
A Bézier curve $\mathbf{p} : [0,1] \to \mathbb{R}^d$ of degree n is defined by $n+1$ control points $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ as follows:

$$\mathbf{p}(t) = \sum_{i=0}^{n} b_{i,n}(t)\mathbf{p}_i$$

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}.$$

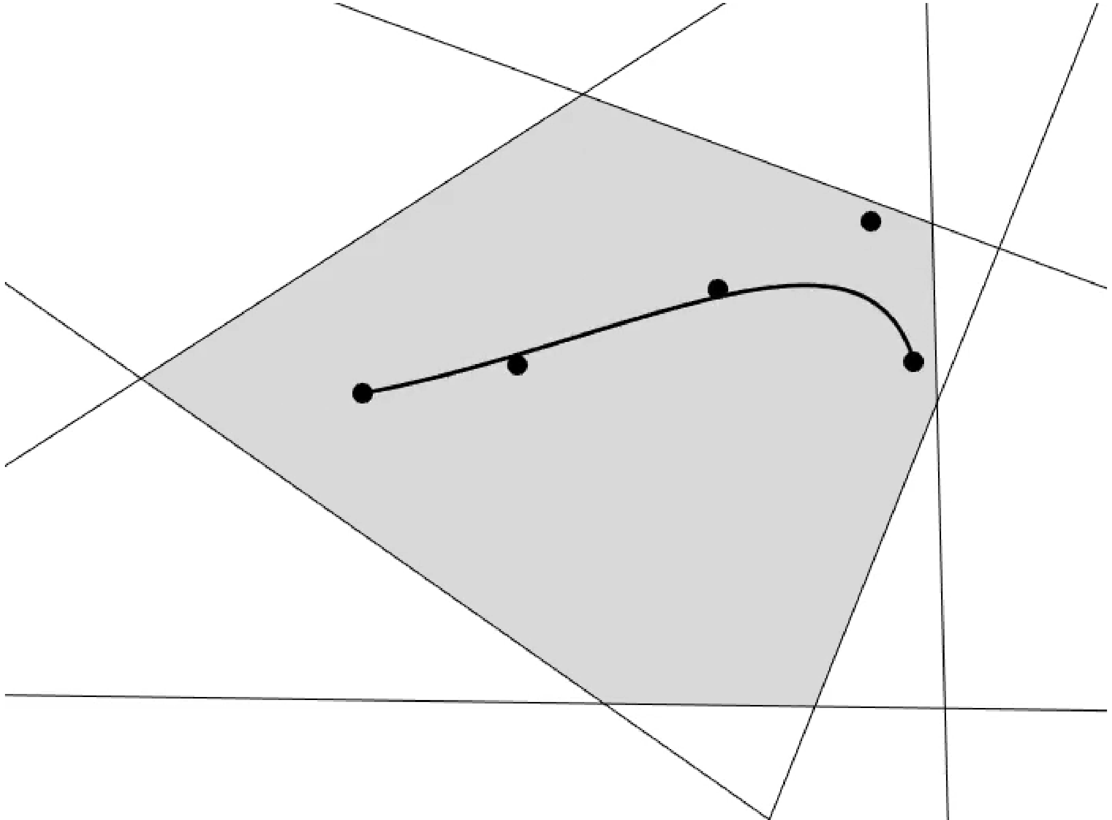
Cubic Bézier Curve

$$\mathbf{p}(t) = (1-t)^3\mathbf{p}_0 + 3t(1-t)^2\mathbf{p}_1 + 3t^2(1-t)\mathbf{p}_2 + t^3\mathbf{p}_3$$



BÉZIER CURVES PROPERTIES

- **Endpoint interpolation:** The curve connects $\mathbf{p}_0$ and $\mathbf{p}_n$, i.e., $\mathbf{p}(0) = \mathbf{p}_0$ and $\mathbf{p}(1) = \mathbf{p}_n$
- C^n smoothness
- **Convex hull property**: The curve lies inside the convex hull of their control points, i.e., $\mathbf{p}(t) \in ConvexHull\{\mathbf{p}_0, \ldots, \mathbf{p}_n\}\ \forall t \in [0, 1]$



CUBIC BÉZIER CURVE OPTIMIZATION

Cubic Bézier Curve

$$\mathbf{p}(t) = (1-t)^3\mathbf{p}_0 + 3t(1-t)^2\mathbf{p}_1 + 3t^2(1-t)\mathbf{p}_2 + t^3\mathbf{p}_3$$

$$\mathbf{p}(0) = \mathbf{p}_0$$

$$\mathbf{p}(1) = \mathbf{p}_3$$

$$\dot{\mathbf{p}}(t) = 3(1-t)^2(\mathbf{p}_1 - \mathbf{p}_0) + 6(1-t)t(\mathbf{p}_2 - \mathbf{p}_1) + 3t^2(\mathbf{p}_3 - \mathbf{p}_2)$$

$$\dot{\mathbf{p}}(0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\dot{\mathbf{p}}(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

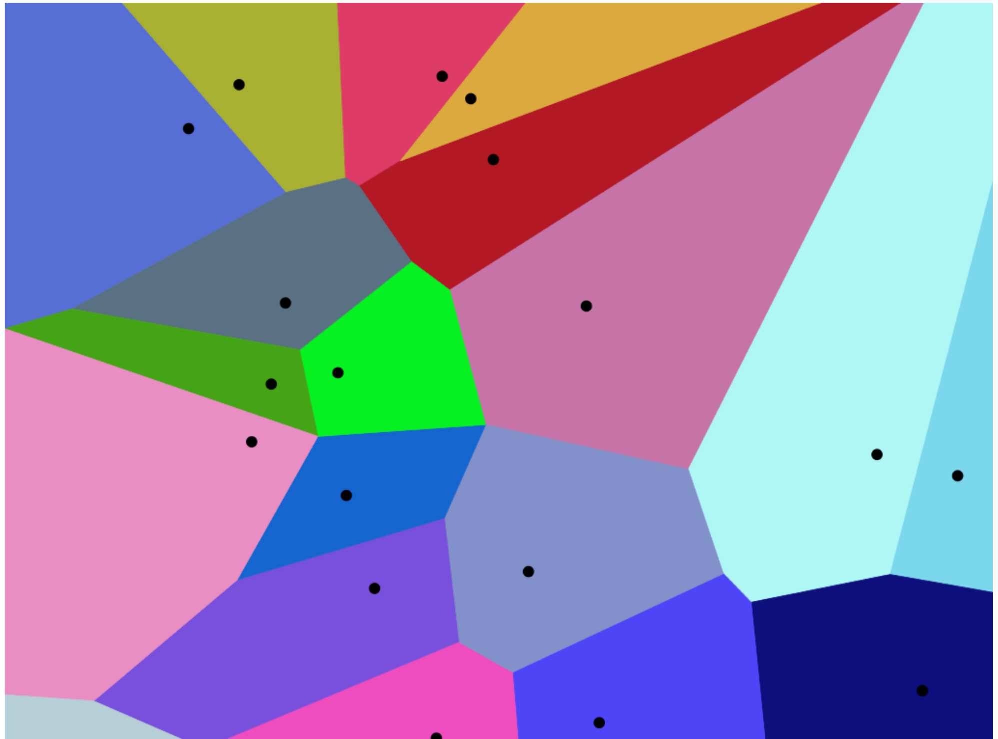
DEMO

VORONOI CELLS

Voronoi region

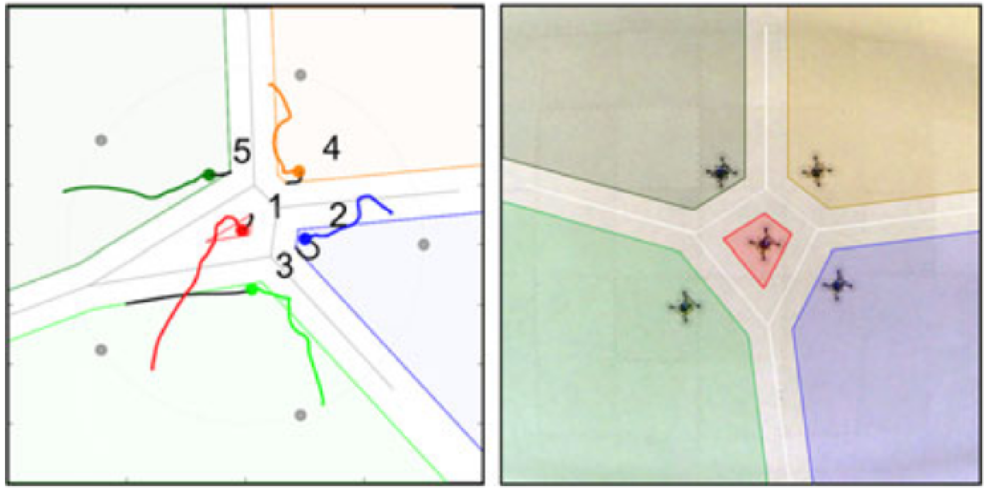
Let $q_1, \ldots, q_K$ be a set of configurations on the state space $\mathcal{Q}$. The Voronoi region is defined as

$$R_k = \{q \in \mathcal{Q} \mid d(q, R_k) \leq d(q, R_j),\ \text{for all } j \neq k\}$$



COLLISION AVOIDANCE VIA BUFFERED VORONOI CELLS (BVC)

- Sense neighbor robots' position
- Compute Voronoi regions (safe polyhedra per robot)
- Shrink ("buffer") regions to account for robot size (still polyhedra)
- Trajectory optimization / control with constraint to stay within safe region for some time



D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, “Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells,” IEEE

DEMO

TRY IT YOURSELF

https://github.com/pbideau/SCIoI_summer_school