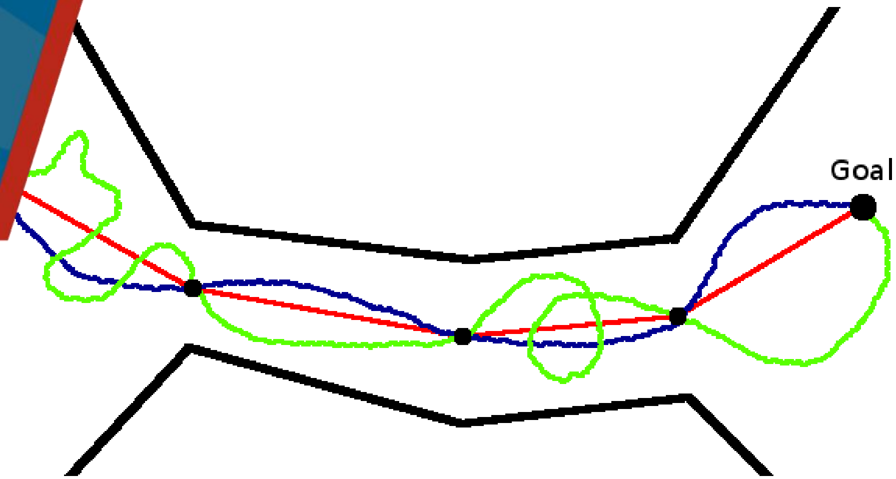
RRLAB
THE ROBOTICS RESEARCH LAB

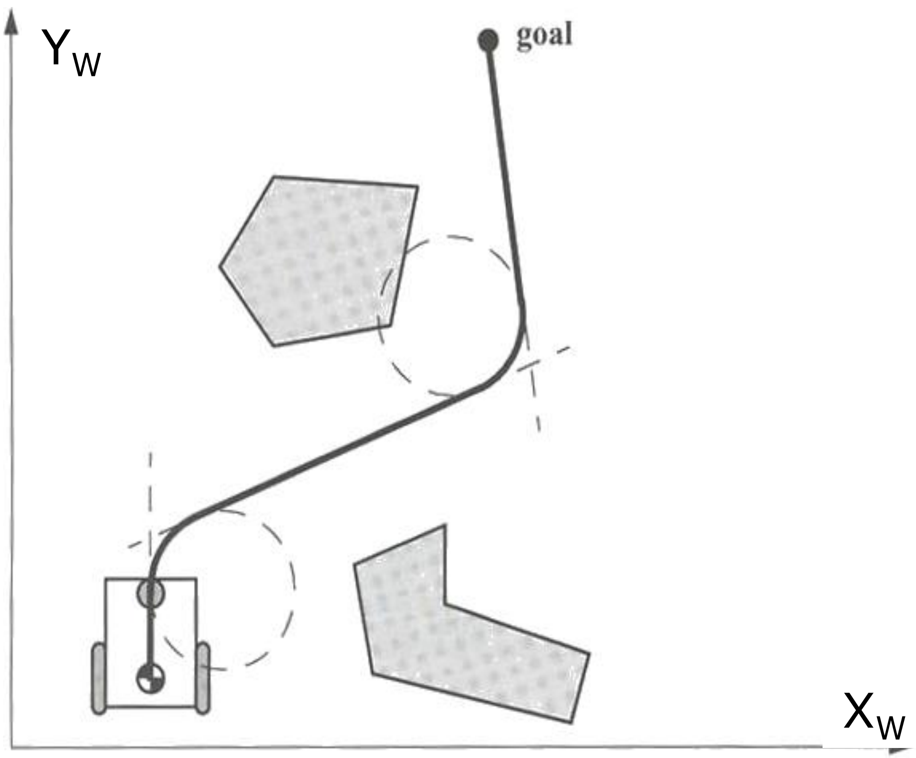
# Trajectory Modeling




TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

# Modeling adequate Trajectories

- Trajectory around obstacles towards a goal
- Efficiency of trajectory due to
  - length
  - time
  - constrains of vehicle
  - precision
- Points in a 2D or in a 3D Euclidian space define the basic trajectories
- Points should be closed but must not lie on the final trajectory



# Additional Constrains

- Movement along a trajectory
  - position or velocity (acceleration) profile as a function of time
  - segmentation of trajectory in simple motion segments (straight lines, segments of a circle, polygons)
  - easy extension of trajectories
  - consideration of dynamical effects

$\Rightarrow$ Goal: smooth trajectory to the goal position

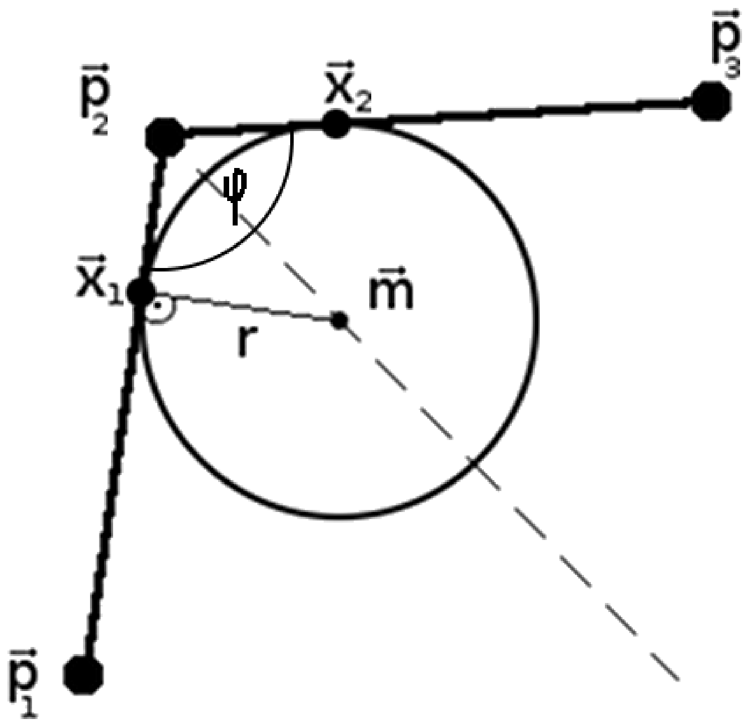
- Determine a function of time $s(t)$ (parametric curve) that is controlled by given points (from constraints)
- E. g. in $2\mathrm{D}$ space: $s(t) \rightarrow (x, y) \in \mathbb{R}^2$

# Linear and Circular Trajectories

- Standard drive kinematics of vehicles allow
  - straight line movements
  - driving along a circular path with radius bigger than aminimal radius (depends on vehicle design)
- Based on vehicle parameters, velocity of wheels, .. the radius can be calculated and vs. (see geometrical solutions)
- Simple approach allowing fast locomotion
- Influence of dynamical effects (e.g. low friction coefficient) can be estimated

# Linear and Circular Trajectories

- Given: $\vec{p}_1$, $\vec{p}_2$, $\vec{p}_3$, $r$
- Goal: Find $\vec{x}_1, \vec{x}_2$ for minimal radius $r$ to pass $\vec{p}_2$ on a circular path
  - $\vec{a} = \frac{\vec{p}_1 - \vec{p}_2}{\|\vec{p}_1 - \vec{p}_2\|}$, $\vec{b} = \frac{\vec{p}_3 - \vec{p}_2}{\|\vec{p}_3 - \vec{p}_2\|}$, $\cos\varphi = \vec{a}^T \vec{b}$
  - $\vec{x}_1 = \vec{p}_2 + t\vec{a}$, $\vec{x}_2 = \vec{p}_2 + t\vec{b}$
  - $\tan\frac{\varphi}{2} = \frac{r}{t} \Rightarrow t = \frac{r}{\tan\frac{\varphi}{2}}$
  - $\vec{m} = \vec{p}_2 + \sqrt{t^2 + r^2} \cdot \frac{\vec{a} + \vec{b}}{\|\vec{a} + \vec{b}\|}$
- Use suitable drive kinematics to steer on arc around $\vec{m}$ from $\vec{x}_1$ to $\vec{x}_2$
- $r$ depends on used drive model



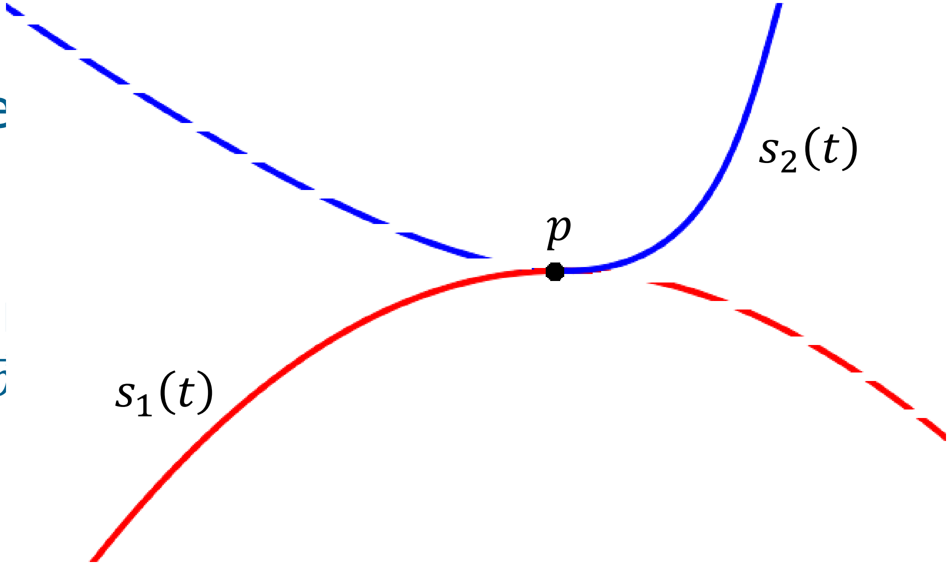
## Smooth Interpolation Curves

- Recap: For $n$ support points one can easily solve a $n-1$ dimensional linear system of equations resulting in a fast computed polynomial curve of degree $n-1$
- However:
  - This kind of interpolation tends to overshoot in outer regions for higher degrees (greater than 6 or 7)
  - Insertion of control points due to additional constraints impossible without changing whole trajectory
- Splines consisting of small curve-segments are more stable and can be locally manipulated

# Interpolating Spline Curves

- An interpolating spline consists of single curves that intersect at control points
- Each segment is a simple parametric curve of low degree (e. g. cubic)
- Smoothness: two neighboring segments must be at least $C^2$ i their connecting control point $\vec{p}$

$$\vec{s}_1(t) = \vec{s}_2(t) = \vec{p}$$
$$\vec{s}_1'(t) = \vec{s}_2'(t)$$
$$\vec{s}_1''(t) = \vec{s}_2''(t)$$



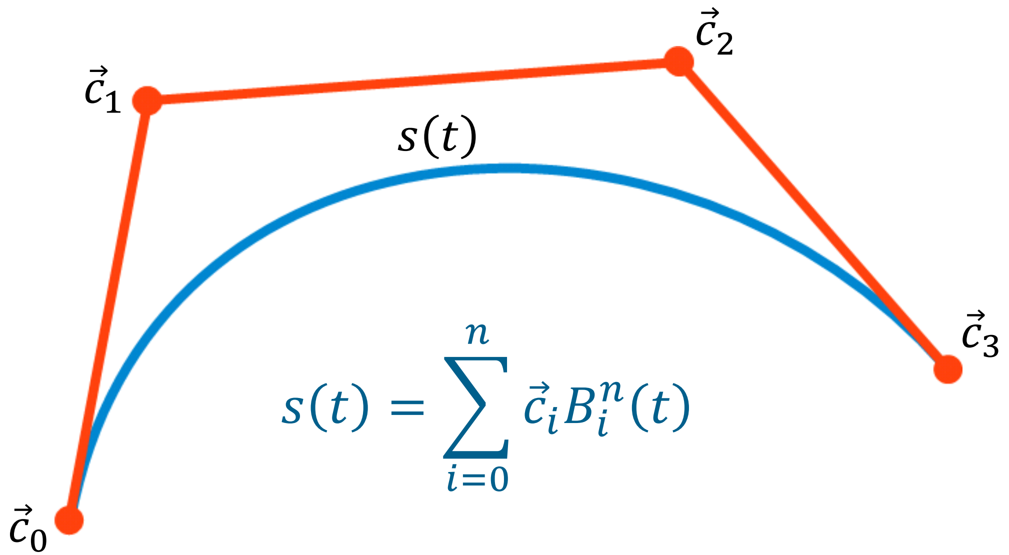

Spline curve consisting of two cubic curves

# Bézier Curves

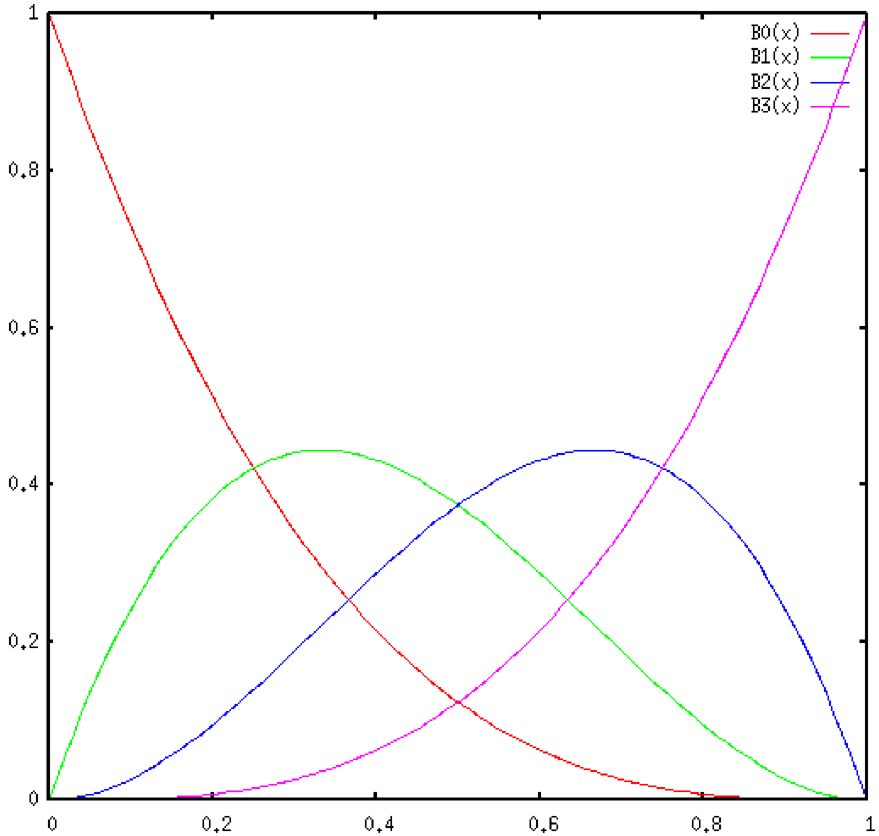
- Linear combination of Bernstein polynomials

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

- $n+1$ control points → polynomial curve of degree $n$
- Control points $\vec{c}_0, \vec{c}_n$ define begin and end of parametric curve
- $\vec{c}_1 - \vec{c}_0$ and $\vec{c}_{n-1} - \vec{c}_n$ define first derivative in $\vec{c}_0$ and $\vec{c}_n$

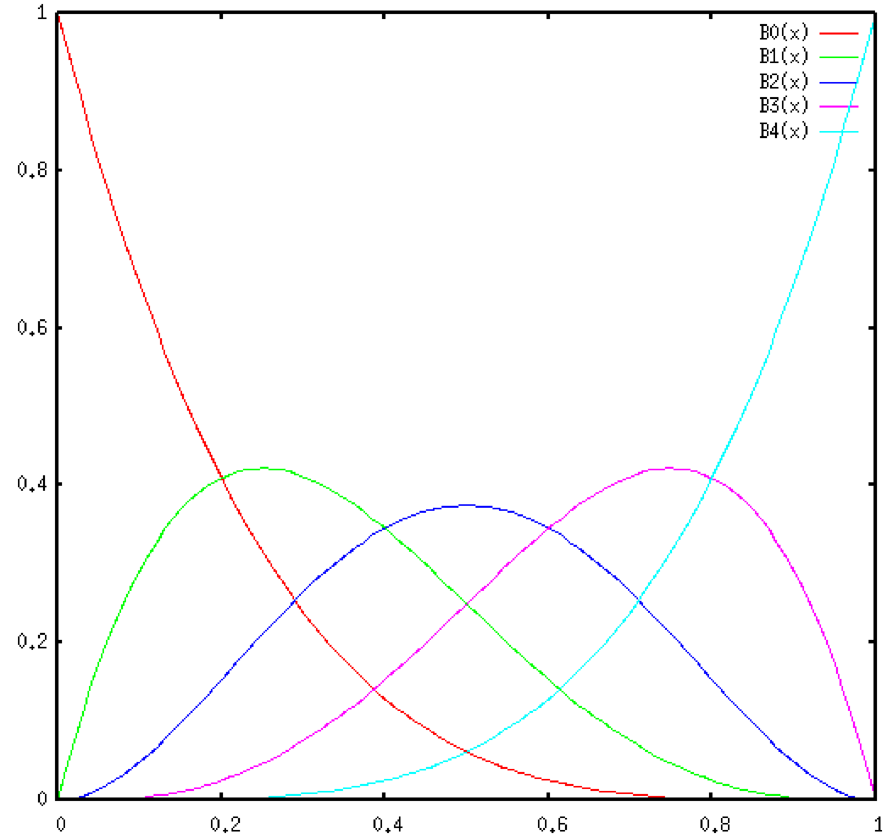


# Bernstein Polynomials




Cubic Bernstein polynomials

$B_i^3(t),\ i \in \{0, \dots, 3\}$



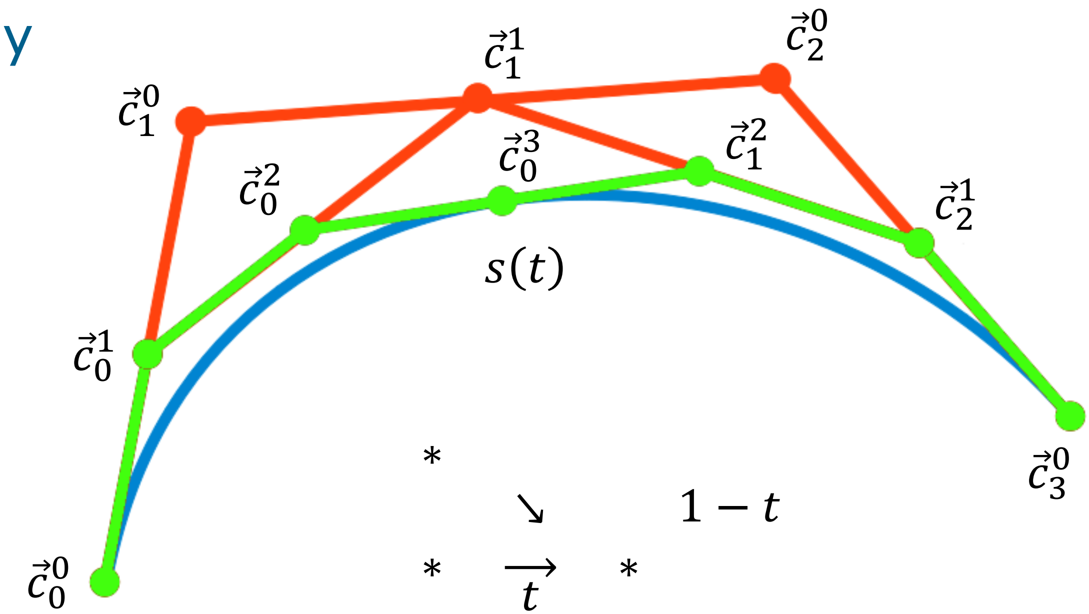

Bernstein polynomials of degree 4

$B_i^4(t),\ i \in \{0, \dots, 4\}$

# De Casteljau’s Algorithm

Evaluating a Bézier curve for a specific parameter $t$ is easily done using de Casteljau’s algorithm

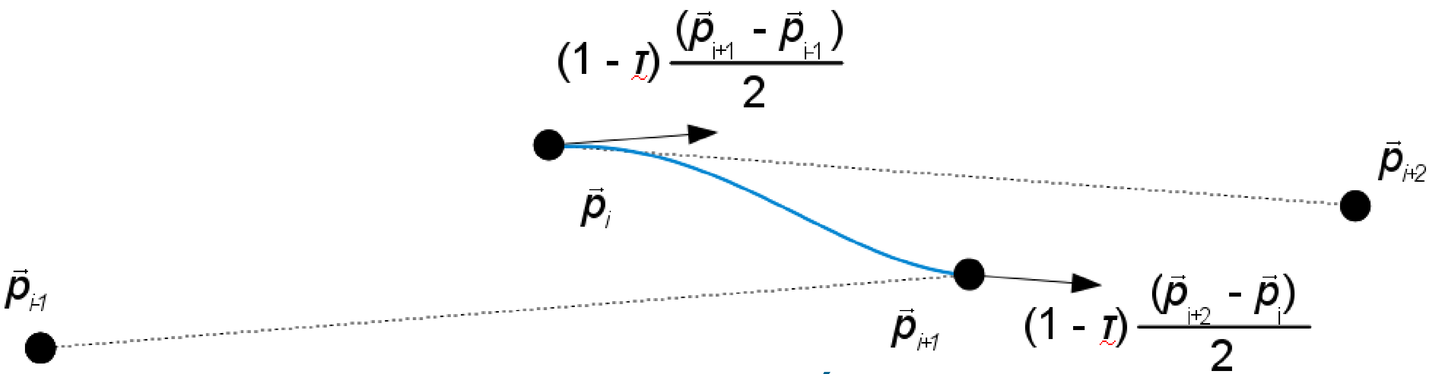
$$
\begin{array}{ccccc}
\vec{c}_0^{\,0} & & & & \\
\vec{c}_1^{\,0} & \vec{c}_0^{\,1} & & & \\
\vec{c}_2^{\,0} & \vec{c}_1^{\,1} & \vec{c}_0^{\,2} & & \\
\vdots & & & \ddots & \\
\vec{c}_n^{\,0} & \vec{c}_{n-1}^{\,1} & \vec{c}_{n-2}^{\,2} & \cdots & \vec{c}_0^{\,n}
\end{array}
$$




Initial points set to control points $\vec{c}_i^{\,0} = P_i$

Evaluation using recurrent relation $\vec{c}_i^{\,k+1} = (1-t)\vec{c}_i^{\,k} + t\vec{c}_{i+1}^{\,k}$

# Interpolation using Cubic Bézier Curves

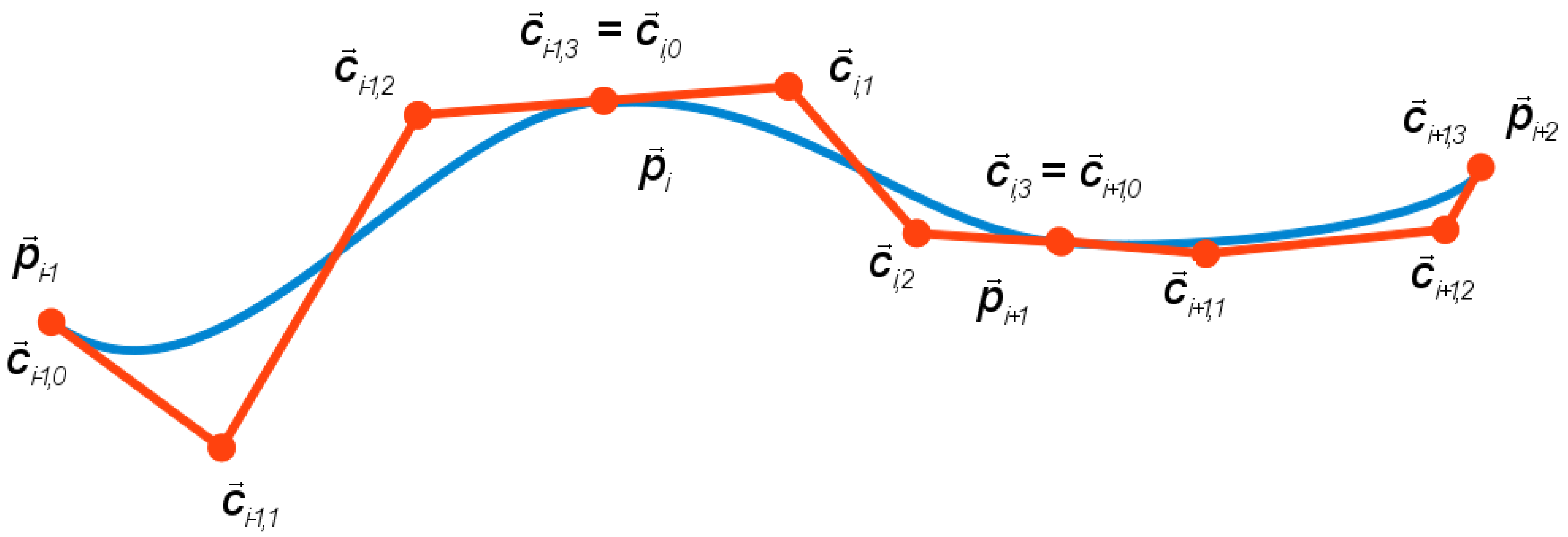



- For each spline segment, a cubic Bézier curve is calculated that fulfills the spline constraints with the tension-parameter $\tau$
- Therefore, for each pair $(\vec{p}_i, \vec{p}_{i+1}) | i \in \{1, \dots, n-1\}$ two more Bézier control points must be computed from their neighbors

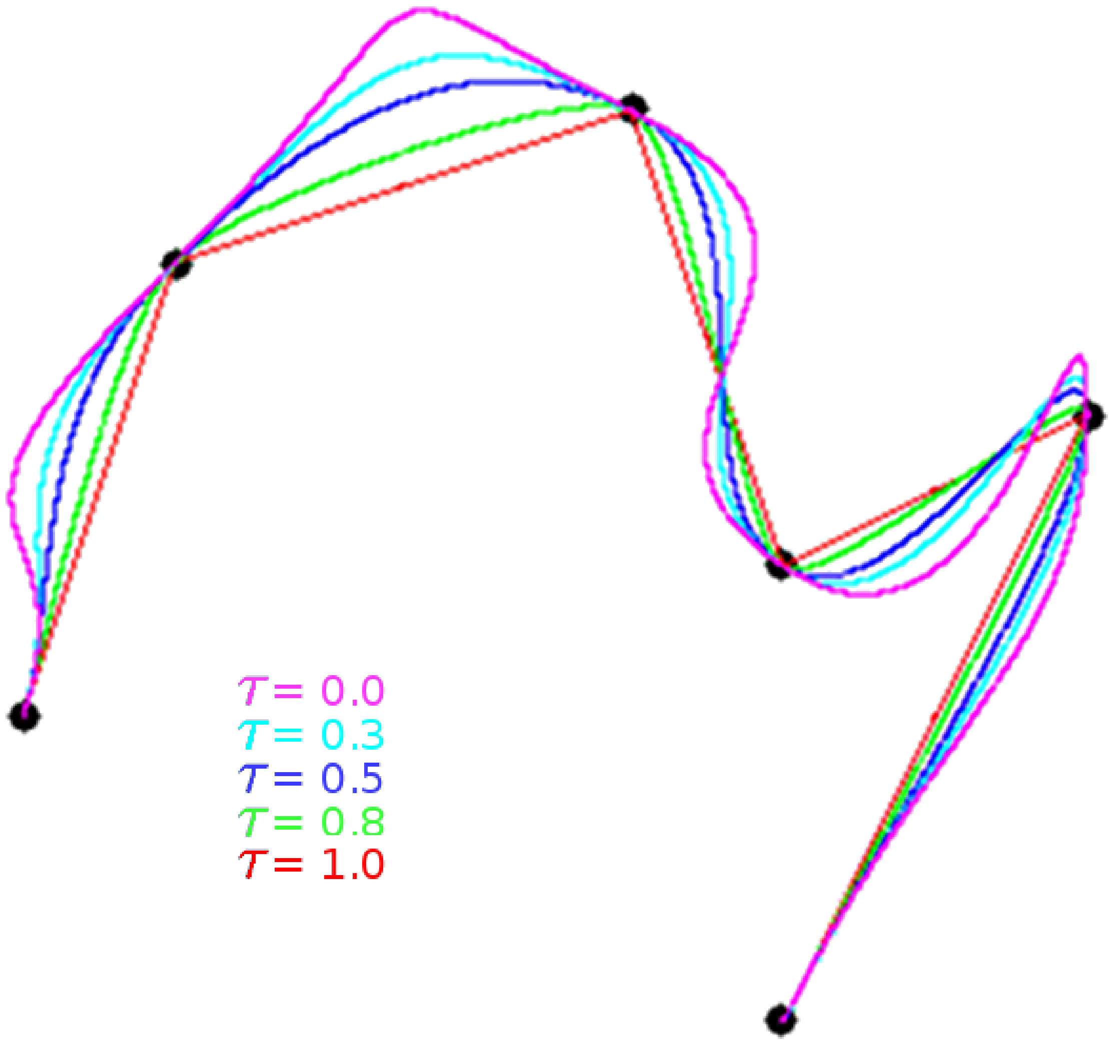
$$\begin{pmatrix} \vec{c}_{i,0} \\ \vec{c}_{i,1} \\ \vec{c}_{i,2} \\ \vec{c}_{i,3} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & 2 & 0 & 0 \\ \tau - 1 & 2 & 1 - \tau & 0 \\ 0 & 1 - \tau & 2 & \tau - 1 \\ 0 & 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} \vec{p}_{i-1} \\ \vec{p}_i \\ \vec{p}_{i+1} \\ \vec{p}_{i+2} \end{pmatrix}$$

# Interpolation using Cubic Bézier Curves

By adding two free constraints (first derivatives in first and last interpolation point, e. g. direction to neighbor point) the whole spline can be constructed



# Effect of the Tension Parameter $\tau$

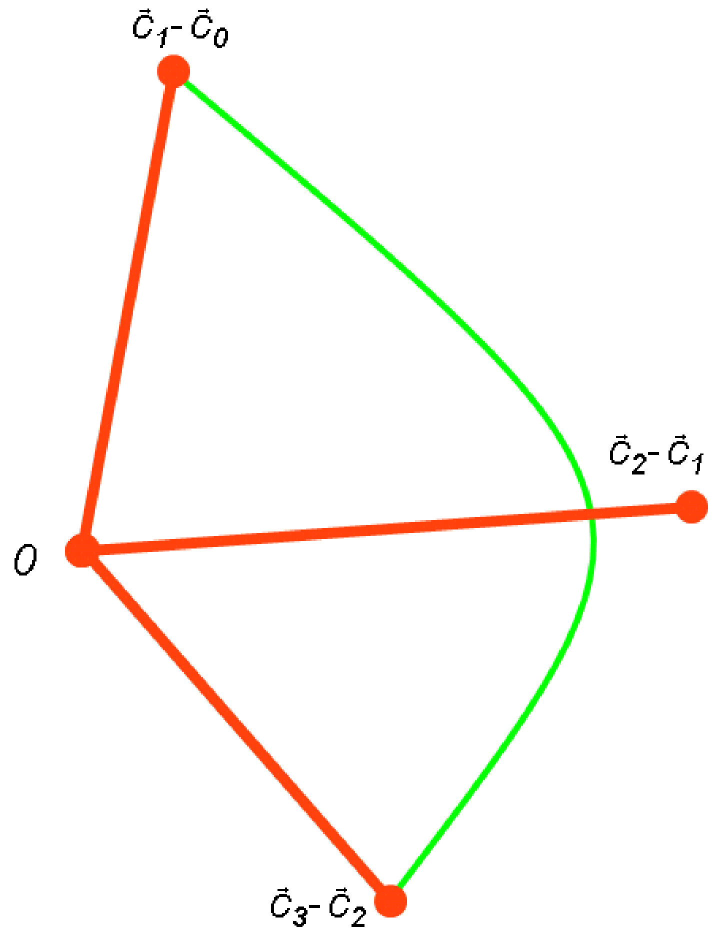


# Evaluation of Spline Curves

- A Bézier curve $s_i(t)$ is defined for the parameter interval $t \in [0,1]$
- As the parameter $t$ of the spline curve $s(t)$ has to grow monotonously, evaluation must consider parameter mapping $u_i(t)$ for each single segment $s_i\big(u_i(t)\big)$
- One possibility is to define $s(i) = \vec{p}_i$ for all interpolation points $\vec{p}_i$
  - Then, $s_i\big(u_i(t)\big)$ for the segment between $\vec{p}_i$ and $\vec{p}_{i+1}$ can be evaluated for $u_i(t) = t - i$
- Another solution is to let all $\vec{p}_i$, $i \in \{0, \dots, n\}$ be located at equidistant locations between $0$ and $1$
  - $\vec{p}_0 = s(0)$, $\vec{p}_n = s(1)$, $\vec{p}_i = s\left(\frac{i}{n}\right)$
  - $u_i(t) = \left(t - \frac{i}{n}\right) n$

# Derivative of Bézier Curves

- The heading and curvature (radius) in every point of the curve is needed to be applied to a vehicle
- Deriving a two dimensional parametric curve at a point $\vec{s}(t)$ yields a two dimensional vector representing the heading and velocity of the curve in this point
- Deriving a whole Bézier curve of degree $n$ is easily done via its control polygon and yields a Bézier curve of degree $n-1$




First hodograph of $s(t)$

# Osculating Circle

- The second derivative of a curve yields a two dimensional vector representing the change of the heading (curvature) and change of velocity (acceleration) of the curve in this point.
  - The curvature component $\kappa$ is the length of the projection of the second derivative to the line perpendicular to the first derivative
  - $r = \frac{1}{\kappa}$ is the radius of a circle that osculates to $\vec{s}(t)$
  - A straight line ($\kappa = 0$) results in $r = \infty$
- Again, the drive kinematics can be used to steer the vehicle along the trajectory sampling the osculating circle in regular steps

# Road Model

- Road curvature: inverse of road radius i. e. $\frac{1}{R}$
- Continuity important
- Splines and Bézier Curves do not consider vehicle dynamics
  -> limited heading rate
- Clothoid spiral
  - Curve
  - Transitions smoothly from one curvature value to another
  - Curvature is a linear function of its arc length
  - Uses Fresnel integrals
  - Example: Going from a straight road to a circular road

# Clothoid spiral

- Parametric equation
  - Clothoid in first quadrant
  - Converges to $\left(\frac{a}{2}, \frac{a}{2}\right)$

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = a \begin{bmatrix} C(t) \\ S(t) \end{bmatrix}$$

- Fresnel integrals

$$C(t) = \int_0^t \cos\left(\frac{\pi u^2}{2}\right) du$$

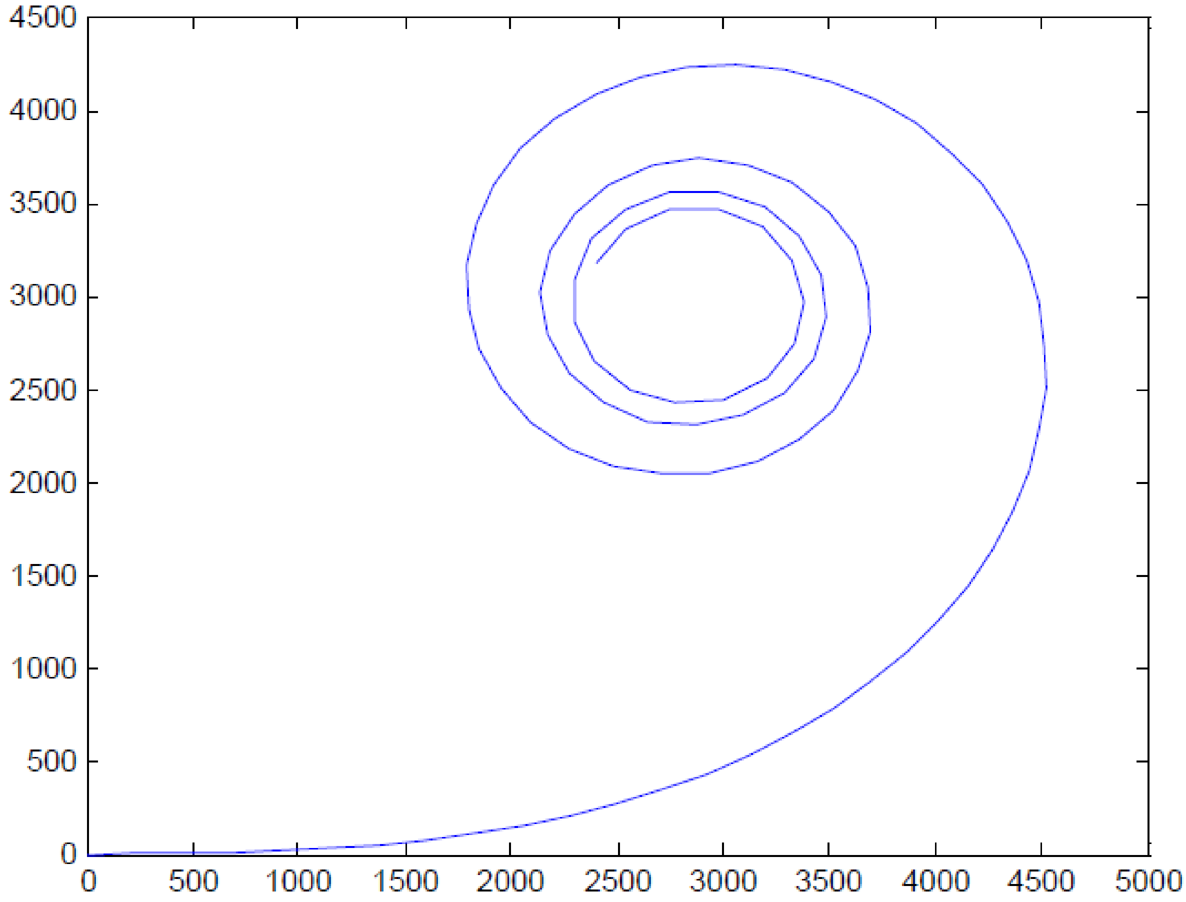
$$S(t) = \int_0^t \sin\left(\frac{\pi u^2}{2}\right) du$$

- Clothoid spiral: Integrals of the Fresnel integrals

$$C_I(t) = \int_0^t C(u)\, du = tC(t) - \frac{1}{\pi} \sin\left(\frac{\pi t^2}{2}\right)$$

$$S_I(t) = \int_0^t S(u)\, du = tS(t) + \frac{1}{\pi} \cos\left(\frac{\pi t^2}{2}\right) - \frac{1}{\pi}$$

# Clothoid spiral

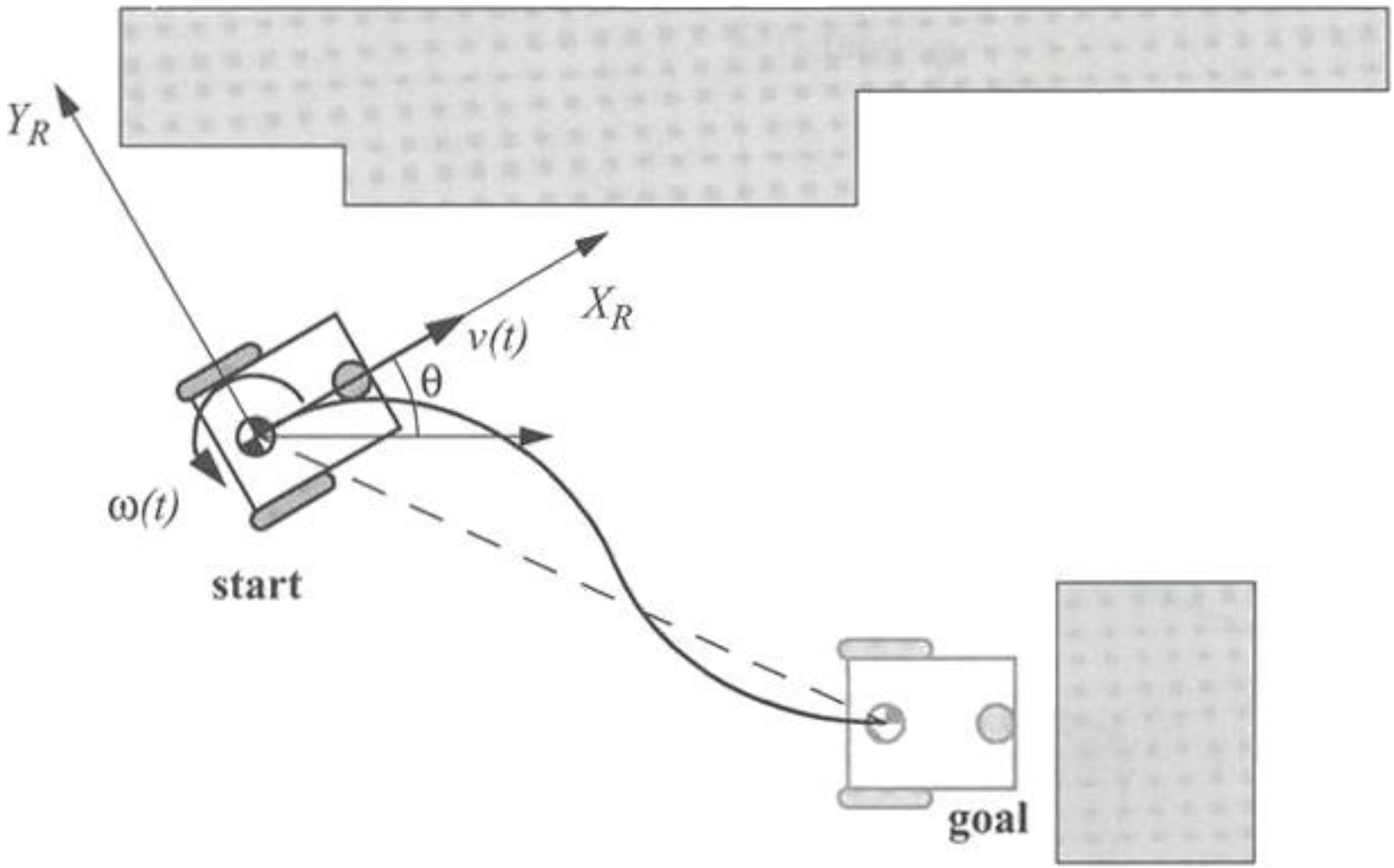



Clothoid spiral using a scaling value $a = 6000$

# Geometric Formulae of Clothoids

| | Geometric Element | Parametric Expression |
|---|---|---|
| 1 | Angle of tangent | $\frac{\pi}{2}t^2$ |
| 2 | Curvature | $\frac{\pi}{a}t$ |
| 3 | Arc length | $ds = a\,dt$ |
| 4 | Center of circle of curvature | $\left(\frac{a}{t}C_I(t), \frac{a}{t}\left\{S_I(t) + \frac{1}{\pi}\right\}\right)$ |

# Feedback Control




Robot motion control

# Feedback Control

- Control via subgoals (feedback control)
- Setting intermediate positions lying on the requested path
- Usage of a pose error $\vec{e} = (x, y, \theta)^T$ in the robot reference frame with target coordinates $x$, $y$, $\theta$
- Task: Find control matrix

$$K = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{pmatrix} | k_{ij} = k(t, \vec{e})$$

such that

$$\begin{pmatrix} v(t) \\ \omega(t) \end{pmatrix} = K \cdot \vec{e} = K \cdot \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

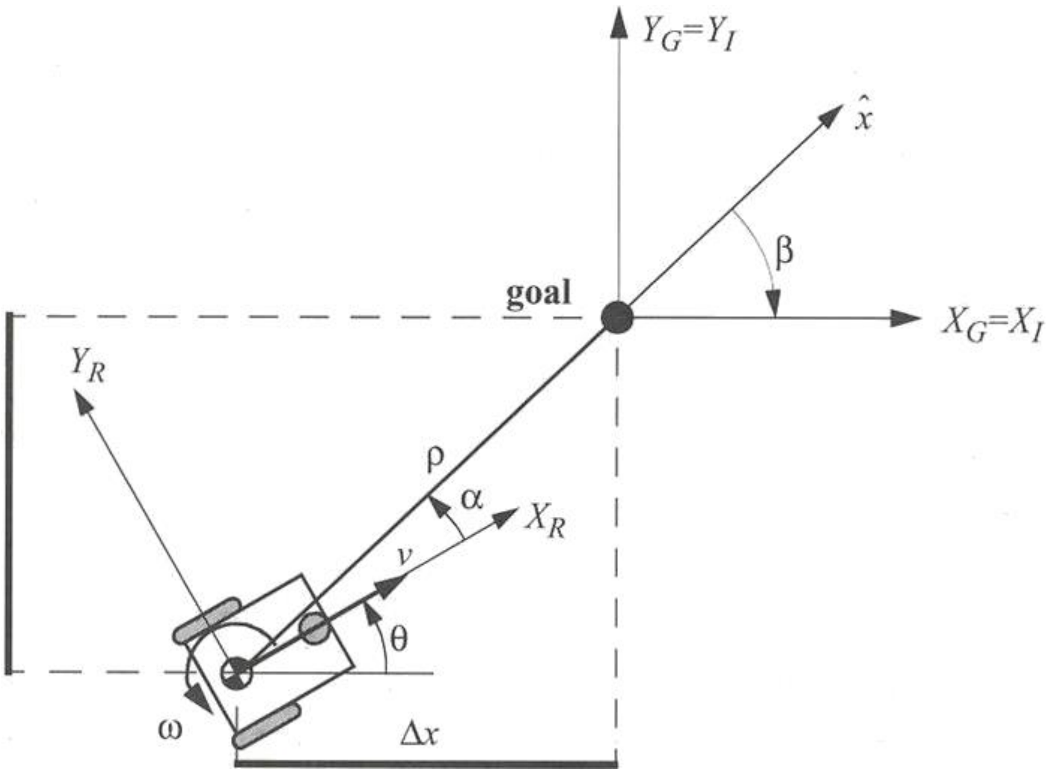
$$\lim_{t \to \infty} \vec{e}(t) = 0$$

# Example: Feedback Control for Differential Drive

Assumptions:

- Goal is at the origin of the inertial frame
- Velocities of the robot refer the inertial frame

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$



Feedback control example

# Example: Feedback Control for Differential Drive

- Transformation into polar coordinates
  $\rho = \sqrt{\Delta x^2 + \Delta y^2}$, $\alpha = -\theta + \operatorname{atan2}(\Delta x, \Delta y)$, $\beta = -\theta - \alpha$

$$\Rightarrow \begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

- If $\alpha \in I_2 = \left(-\pi, -\frac{\pi}{2}\right) \cup \left(\frac{\pi}{2}, \pi\right)$ set $v := -v$

$$\Rightarrow \begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} \cos\alpha & 0 \\ -\frac{\sin\alpha}{\rho} & -1 \\ \frac{\sin\alpha}{\rho} & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

- Task: Find trajectory with $\alpha(0) \in I_1 = \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \wedge \alpha(t) \in I_1$

# Example: Feedback Control for Differential Drive

- Calculation of changes of polar coordinates (linear control law) with $v = k_p \rho$ and $\omega = k_\alpha \alpha + k_\beta \beta$ we get

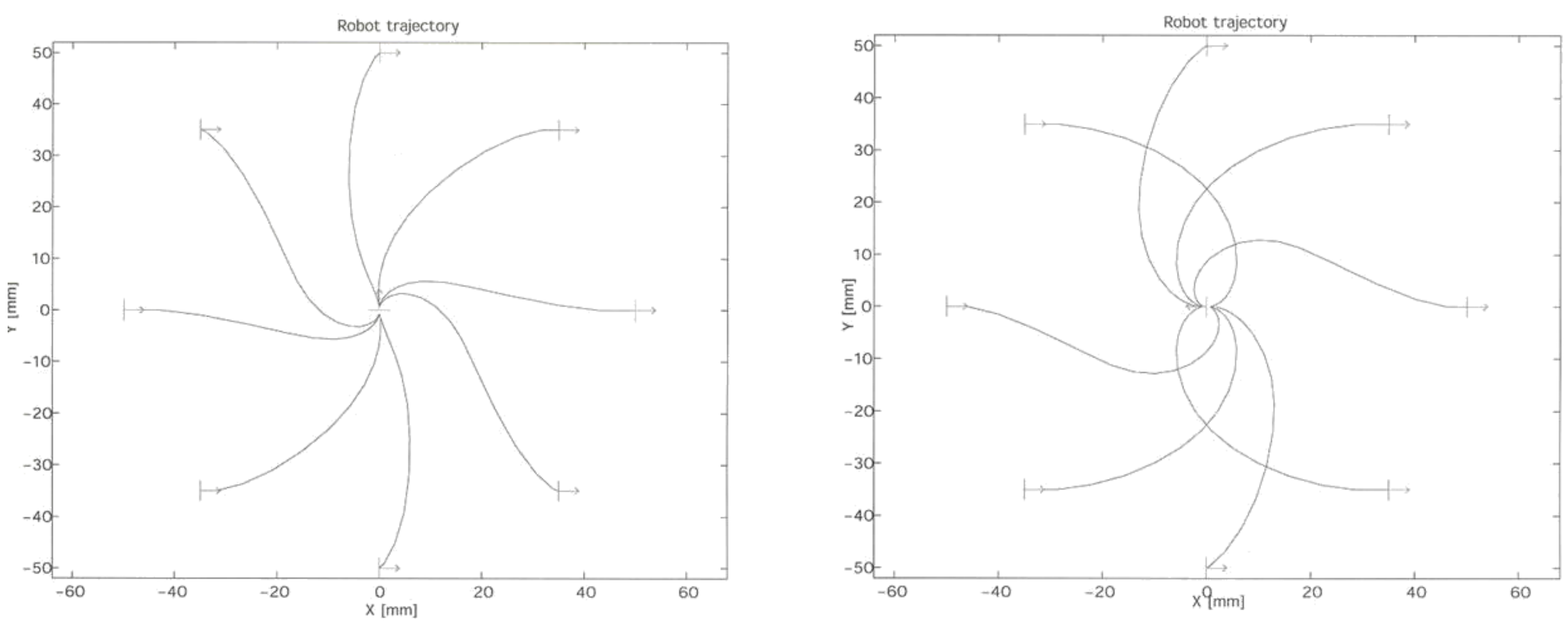
$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -k_\rho \rho \cos \alpha \\ k_\rho \sin \alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin \alpha \end{pmatrix}$$

- Results for this description
  - No singularities
  - $\alpha$ and $\beta$ are always in the range $(-\pi, \pi)$
  - Control signal $v$ depends on $\alpha(0)$:
    if $\alpha(0) > 0$ then $v$ positive, $-v$ otherwise

# Example: Feedback Control for Differential Drive

Trajectories to travel from a circle path towards the goal in the center with the following parameters:

$$K = \left(k_\rho, k_\alpha, k_\beta\right) = (3,8,-1.5)$$




Robot trajectories

# Local Stability Problem

The closed-loop control system is locally exponentially stable if

$$k_\rho > 0, \qquad k_\beta < 0, \qquad k_\alpha - k_\rho > 0$$

linearized around the equilibrium $(\cos\alpha = 1,\ \sin\alpha = 0)$

$$\begin{pmatrix} \dot\rho \\ \dot\alpha \\ \dot\beta \end{pmatrix} = \begin{pmatrix} -k_\rho \rho \cos\alpha \\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin\alpha \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \dot\rho \\ \dot\alpha \\ \dot\beta \end{pmatrix} = \begin{pmatrix} -k_\rho & 0 & 0 \\ 0 & -(k_\alpha - k_\rho) & -k_\beta \\ 0 & -k_\rho & 0 \end{pmatrix} \begin{pmatrix} \rho \\ \alpha \\ \beta \end{pmatrix}$$

# Proof of Stability

- The system is locally exponentially stable if the eigenvalues of matrix $A$ all have a negative real part

$$A = \begin{pmatrix} -k_\rho & 0 & 0 \\ 0 & -(k_\alpha - k_\rho) & -k_\beta \\ 0 & -k_\rho & 0 \end{pmatrix}$$

- The characteristic polynomial $\det(\lambda I - A)$ is

$$(\lambda + k_\rho)\left(\lambda^2 + \lambda(k_\alpha - k_\rho) - k_\rho k_\beta\right) \rightarrow k_\rho > 0, -k_\beta > 0, k_\alpha - k_\rho > 0$$

strong stability condition, if $k_\rho > 0$, $k_\beta < 0$, $k_\alpha + \frac{5}{3}k_\beta - \frac{2}{\pi}k_\rho > 0$ no change in direction of $v$ (always head for goal)

# Coming Next

Feature Extraction
and Object Recognition