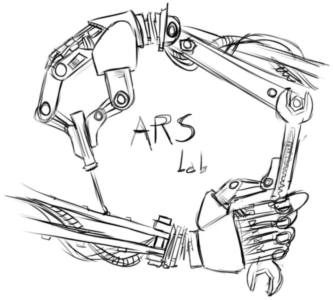
Cinematica e Controllo di un robot mobile

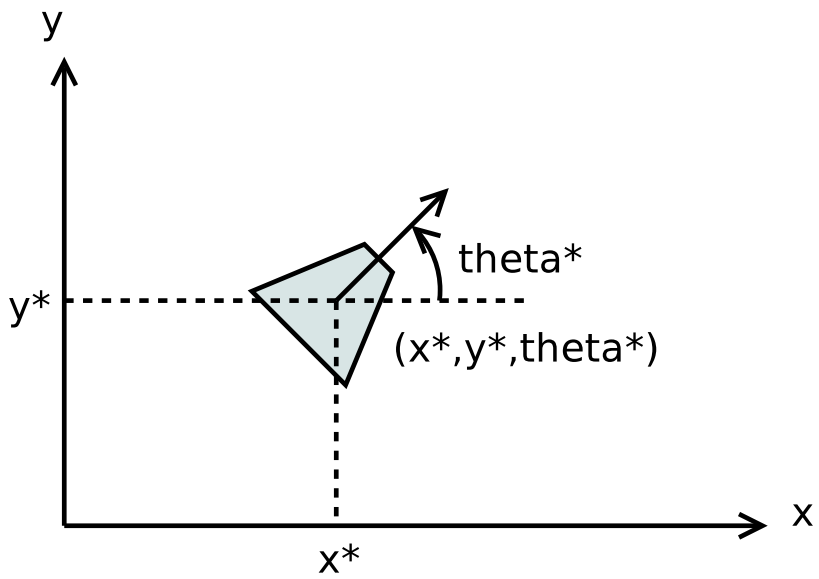
Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



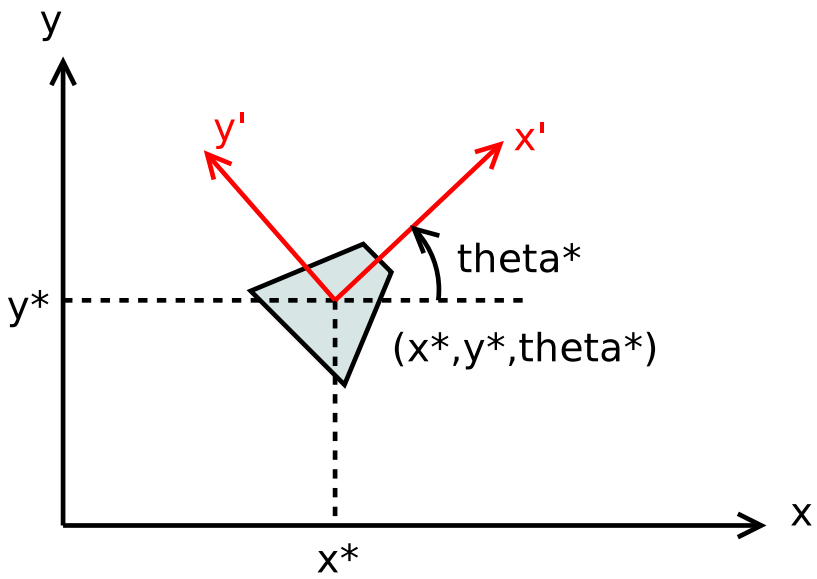
Programmazione Sistemi Robotici

Cinematica di Robot Mobile su Ruote



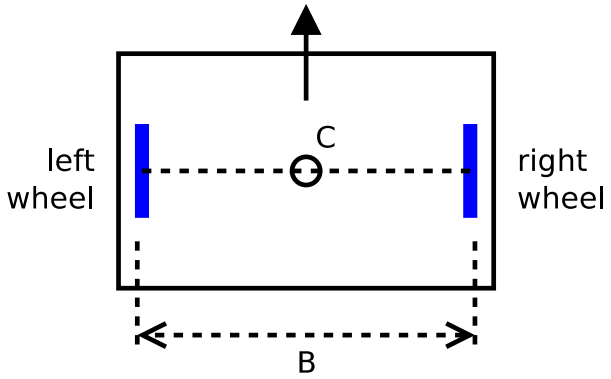
- La **posa** istantanea di un robot su un ambiente **bidimensionale** è caratterizzata da:
  - Posizione *x*
  - Posizione *y*
  - Orientamento *theta* (rispetto all'asse *x*)

Cinematica di Robot Mobile su Ruote



- Si considerano due sistemi di riferimento:
  - **Body system**, sistema di riferimento *locale* del robot (x', y')
  - **Earth system**, sistema di riferimento *globale* dell'ambiente (x, y)
- La trasformazione tra un sistema e l'altro avviene usando le normali formule di roto-traslazione

Modello di Robot Mobile a due ruote indipendenti



- Due ruote indipendent poste ai lati del robot
- Il punto C è detto **centro di istantanea rotazione** ed è il riferimento per la posizione globale del robot
- I parametri di geometria per il modello cinematico sono:
  - r_L, raggio della ruota sinistra
  - r_R, raggio della ruota destra
  - n_L, numero di tick per giro dell'encoder sinistro
  - n_R, numero di tick per giro dell'encoder destro
  - B, distanza tra i punti di contatto delle due ruote (**wheelbase**)

Cinematica del Robot Mobile a due ruote indipendenti

- Parametri:
  - r_L, raggio della ruota sinistra
  - r_R, raggio della ruota destra
  - n_L, numero di tick per giro dell'encoder sinistro
  - n_R, numero di tick per giro dell'encoder destro
  - B, distanza tra i punti di contatto delle due ruote (**wheelbase**)
- Detti $\Delta tick_L$ e $\Delta tick_R$ il numero di **tick contati**, su ruota destra e sinistra, nell'intervallo di campionamento ΔT, abbiamo:

$$\Delta d_L = \frac{2\pi r_L}{n_L} \Delta tick_L \qquad \Delta d_R = \frac{2\pi r_R}{n_R} \Delta tick_R$$

$$\Delta theta = \frac{\Delta d_R - \Delta d_L}{B}$$

$$v_L = \frac{\Delta d_L}{\Delta T} \qquad v_R = \frac{\Delta d_R}{\Delta T}$$

Cinematica del Robot Mobile a due ruote indipendenti

- Detti $\Delta tick_L$ e $\Delta tick_R$ il numero di **tick contati**, su ruota destra e sinistra, nell'intervallo di campionamento ΔT, abbiamo:

$$\Delta d_L = \frac{2\pi r_L}{n_L}\Delta tick_L \qquad \Delta d_R = \frac{2\pi r_R}{n_R}\Delta tick_R$$

$$\Delta\theta = \frac{\Delta d_R - \Delta d_L}{B}$$

$$v_L = \frac{\Delta d_L}{\Delta T} \qquad v_R = \frac{\Delta d_R}{\Delta T}$$

- Dove:
  - Δd_L, Δd_R, distanza percorsa (in ΔT) dalla ruota sinistra/destra
  - $\Delta theta$, variazione dell'orientamento (in radianti)
  - v_L, v_R, velocità istantanea della routa sinistra/destra

Cinematica del Robot Mobile a due ruote indipendenti

$$\Delta d_L = \frac{2\pi r_L}{n_L}\Delta tick_L \qquad \Delta d_R = \frac{2\pi r_R}{n_R}\Delta tick_R$$

$$\Delta\theta = \frac{\Delta d_R - \Delta d_L}{B}$$

$$v_L = \frac{\Delta d_L}{\Delta T} \qquad v_R = \frac{\Delta d_R}{\Delta T}$$

- Sulla base delle relazioni precedenti, l'aggiornamento della posa si ottiene con:

$$\begin{aligned} x &= x + \frac{\Delta d_L + \Delta d_R}{2}\cos(\theta + \frac{\Delta\theta}{2}) \\ y &= y + \frac{\Delta d_L + \Delta d_R}{2}\sin(\theta + \frac{\Delta\theta}{2}) \\ \theta &= \theta + \Delta\theta \end{aligned}$$

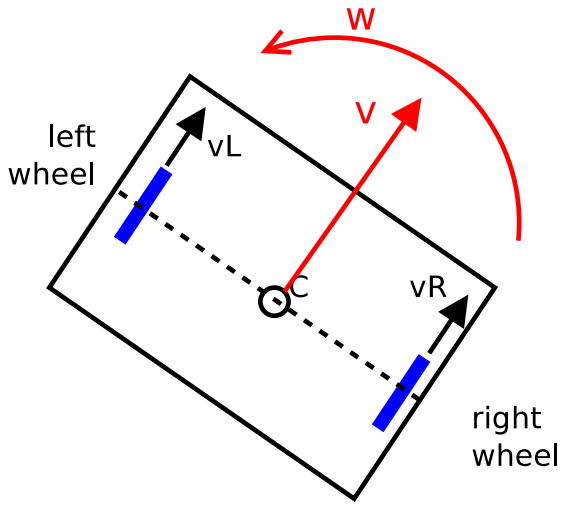
Cinematica del Robot Mobile a due ruote indipendenti

- Sulla base delle relazioni precedenti, l'aggiornamento della posa si ottiene con:

$$\begin{aligned} x &= x + \frac{\Delta d_L + \Delta d_R}{2} \cos(\theta + \frac{\Delta\theta}{2}) \\ y &= y + \frac{\Delta d_L + \Delta d_R}{2} \sin(\theta + \frac{\Delta\theta}{2}) \\ \theta &= \theta + \Delta\theta \end{aligned}$$

- Le relazioni precedenti, note come **formule per l'odometria**, si determinano tramite l'**integrazione dell'equazione del moto**
- Tale integrazione (che non può essere effettuata analiticamente) è effettuata **numericamente** e pertanto contiene un **errore che si propaga in modo additivo** nella stima della posa (x, y, θ)
- La posa stimata (x, y, θ) è pertanto soggetta ad un errore non nullo e non eliminabile che cresce con l'andare del tempo/distanza percorsa

Cinematica del Robot Mobile a due ruote indipendenti



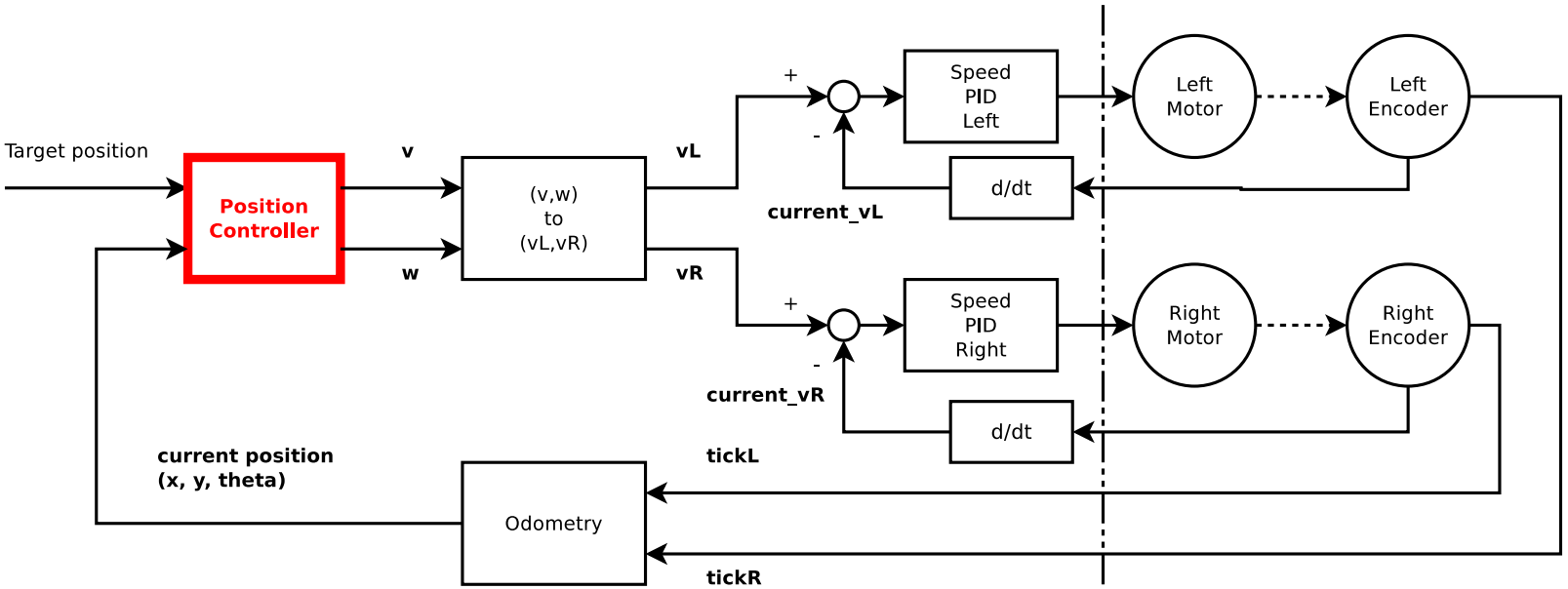
- La **velocità** è invece rappresentata secondo due possibili sistemi di riferimento
  - (v_L, v_R), velocità istantanea delle routa sinistra e destra
  - (v, ω), velocità istantanea di **traslazione** e **rotazione** del punto C
- Tra i due sistemi di riferimento esiste la seguente trasformazione:

$$v = \frac{v_R + v_L}{2} \qquad \omega = \frac{v_R - v_L}{B}$$

$$v_L = v - \frac{B\,\omega}{2} \qquad v_R = v + \frac{B\,\omega}{2}$$

Modello di Controllo in Posizione

- Per il **controllo del moto** è opportuno utilizzare il sistema di riferimento (v, ω), e trasformare nel sistema (v_L, v_R) i valori ottenuti.
- Stabiliti i valori target di (v_L, v_R), ogni ruota implementa poi il controllo in velocità classico.

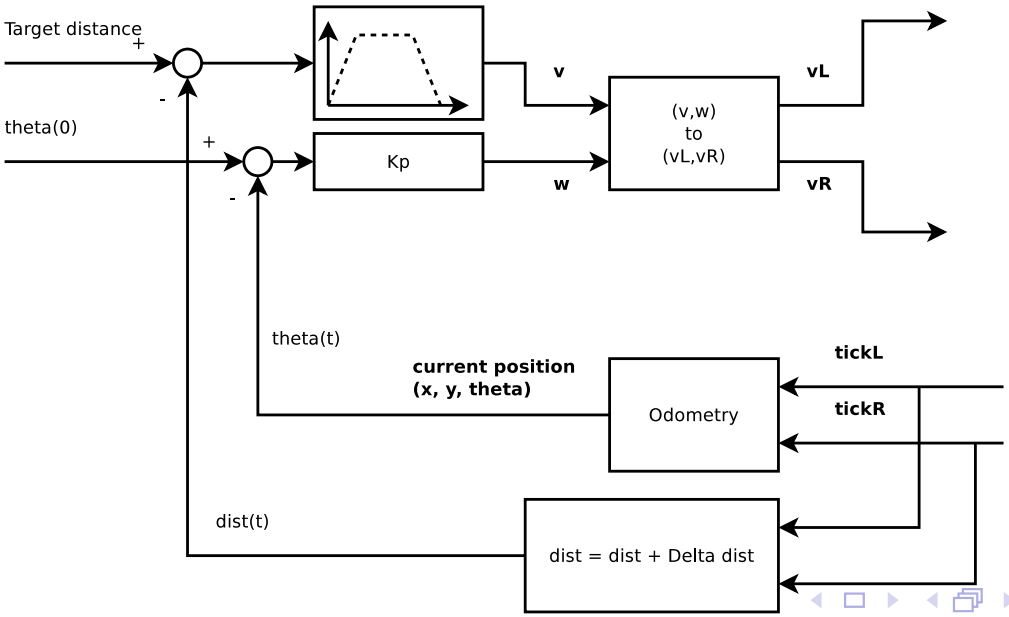


Moto su traiettoria rettilinea

- Desideriamo un controllo del moto in cui il robot percorre una certa distanza *TargetDistance* su una **traiettoria rettilinea**.
- Calcoliamo, a ogni iterazione, la distanza percorsa dall'inizio del moto come:

$$dist(t) = dist(t) + \frac{\Delta d_L + \Delta d_R}{2}$$

- Utilizziamo due controllori:
  - Su v, applichiamo il profilo trapezoidale di velocità, basato sull'**errore lineare** $targetDistance - dist(t)$
  - Su ω, applichiamo un controllore proporzionale, basato sull'**errore angolare** sull'orientamento iniziale $\theta(t) - \theta(0)$



Pseudo-codice

while *True* **do**

On each ΔT;

// Kinematics

$\Delta d_L \leftarrow read\_left\_encoder()$;

$\Delta d_R \leftarrow read\_right\_encoder()$;

$V_L \leftarrow \frac{\Delta d_L}{\Delta T}$; $V_R \leftarrow \frac{\Delta d_R}{\Delta T}$;

$\Delta d \leftarrow \frac{\Delta d_L + \Delta d_R}{2}$;

$\Delta \theta \leftarrow \frac{\Delta d_R - \Delta d_L}{B}$;

// Odometry

$x \leftarrow x + \Delta d \cos(\theta + \frac{\Delta \theta}{2})$;

$y \leftarrow y + \Delta d \sin(\theta + \frac{\Delta \theta}{2})$;

$\theta \leftarrow \theta + \Delta \theta$;

// Position Control

...

// Speed Control

...

end

Pseudo-codice

while *True* **do**

On each ΔT;

```
// Kinematics
...
// Odometry
...
// Position Control
```

$distance \leftarrow distance + \Delta d$;

$position\_error \leftarrow target\_distance - distance$;

$v_{target} \leftarrow position\_controller(position\_error)$;

$theta\_error \leftarrow \theta_0 \ominus \theta$;

$\omega_{target} \leftarrow angle\_controller(theta\_error)$;

```
// Speed Control
...
```

end

Moto su traiettoria rettilinea

Pseudo-codice

while *True* **do**

On each ΔT;

```
// Kinematics
...
// Odometry
...
// Position Control
...
// Speed Control
```

$V_{L/target} = v - \frac{B\,\omega}{2}$; $V_{R/target} = v + \frac{B\,\omega}{2}$;

$error \leftarrow V_{L/target} - V_L$;
$PWM_L \leftarrow PI\_left\_speed\_controller(error)$;

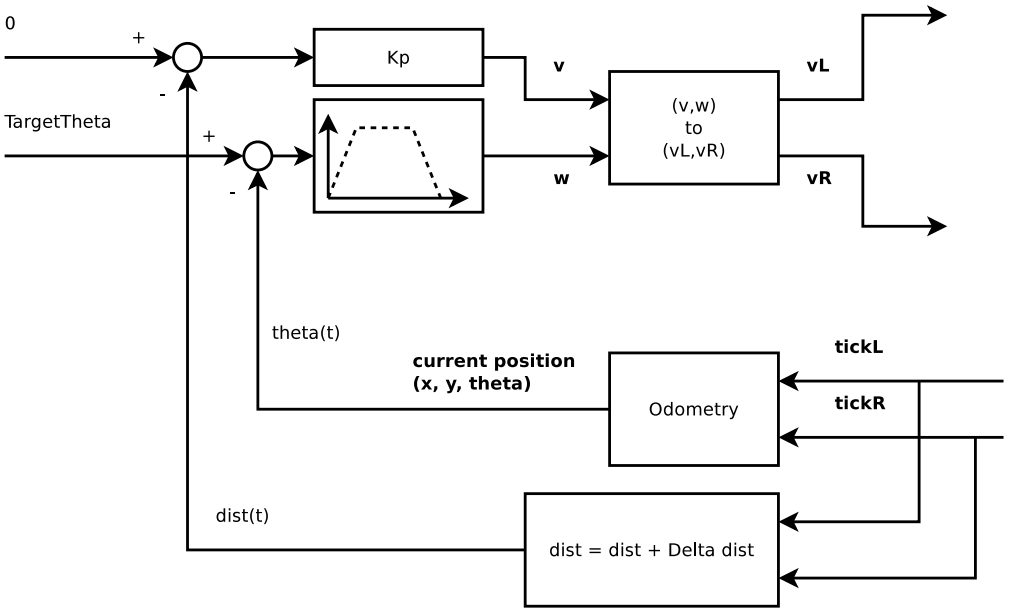
$error \leftarrow V_{R/target} - V_R$;
$PWM_R \leftarrow PI\_right\_speed\_controller(error)$;

$drive\_motor(PWM_L, PWM_R)$;

end

Rotazione sul posto

- Desideriamo ruotare (sul posto) in modo da raggiungere un'orientamento assoluto θ_{target}.
- Utilizziamo due controllori:
  - Su ω, applichiamo il profilo trapezoidale di velocità (angolare), basato sull'**errore angolare** sull'orientamento $\theta(t) - \theta_{target}$
  - Poichè occorre non spostarsi dal punto di partenza, applichiamo un controllore proporzionale basato sull'**errore lineare** $0 - dist(t)$ (0 perchè non vogliamo spostarci dal posto)



Rotazione sul posto

Pseudo-codice

while *True* **do**

On each ΔT;

```
// Kinematics
...
// Odometry
...
// Position Control
```

$distance \leftarrow distance + \Delta d$;

$position\_error \leftarrow 0 - distance$;

$v_{target} \leftarrow position\_controller(position\_error)$;

$theta\_error \leftarrow \theta_{target} \ominus \theta$;

$\omega_{target} \leftarrow angle\_controller(theta\_error)$;

```
// Speed Control
...
```

end

Rotazioni, angoli e differenze

- Nell'usare gli angoli, occorre considerare che la loro artimetica è **modulare**
- Ad esempio, i valori (in gradi) 350 e -10 si riferiscono allo **stesso angolo**
- Tuttavia, il **differente segno** implica un comportamento diverso del controllore che genererà velocità positive o negative
- Nel calcolo degli angoli, è pertanto opportuno **normalizzare** sempre il risultato, riconducendolo all'intervallo $[-\pi, \pi]$ (risp. $[-180, 180]$)

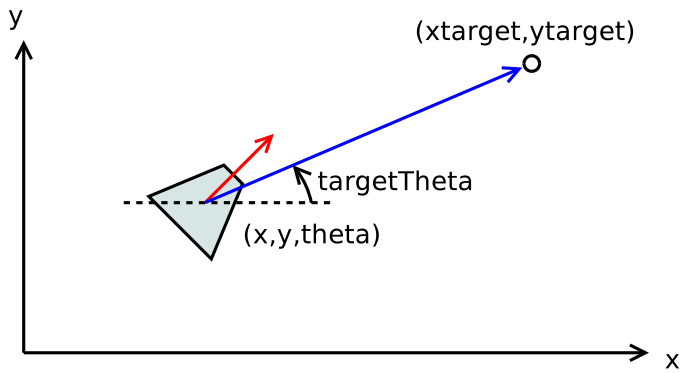
Raggiungimento di un punto (controllo polare)

- Desideriamo raggiungere un punto specifico del piano di coordinate (x_{target}, y_{target}).
- In ogni istante dobbiamo:
  - Percorrere la distanza (lineare) che ci separa dal punto target:

$$dist = \sqrt{(x_{robot} - x_{target})^2 + (y_{robot} - y_{target})^2}$$

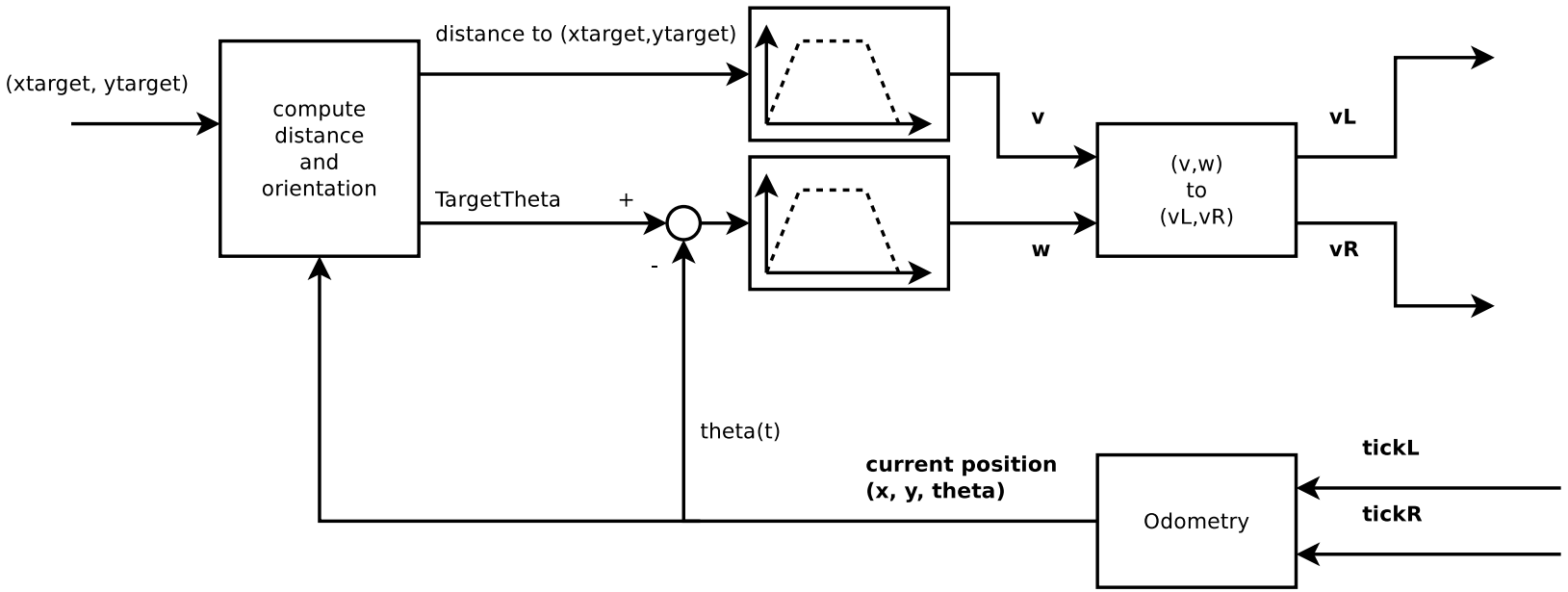
  - Assicurare che l'orientamento del robot sia coerente con l'orientamento della traiettoria:

$$\theta_{trajectory} = \arctan \frac{y_{target} - y_{robot}}{x_{target} - x_{robot}}$$



Raggiungimento di un punto (controllo polare)

- Usiamo due controllori, in entrambi i casi applichiamo il profilo trapezoidale della velocità
- Il controllore su v utilizza, come errore, proprio la **distanza** dal punto target
- Il controllore su ω utilizza, come errore, la **differenza** tra orientamento della traiettoria e orientamento del robot $\theta_{trajectory} - \theta_{robot}$



Raggiungimento di un punto (controllo polare)

Pseudo-codice

while *True* **do**

On each ΔT;

```
// Kinematics
...
// Odometry
...
// Position Control
```

$position\_error \leftarrow \sqrt{(x_{target} - x)^2 + (y_{target} - y)^2}$;

$v_{target} \leftarrow position\_controller(position\_error)$;

$\theta_{target} \leftarrow \text{atan2}(y_{target} - y, x_{target} - x)$;

$theta\_error \leftarrow \theta_{target} \ominus \theta$;

$\omega_{target} \leftarrow angle\_controller(theta\_error)$;

```
// Speed Control
...
```

end

Inseguimento di una traiettoria generica (virtual robot)

- Desideriamo muoverci su una traiettoria generica
- Specifichiamo la **legge oraria** $\widetilde{x}(t), \widetilde{y}(t)$ delle coordinate della traiettoria che un ipotetico robot (virtual robot) sta seguendo
- Discretizziamo la legge: $\widetilde{x}(k\Delta T), \widetilde{y}(k\Delta T)$
- Utilizziamo i punti $\widetilde{x}(k\Delta T), \widetilde{y}(k\Delta T)$ come **target temporanei** e applichiamo il **controllo polare**
- Il robot reale procederà via via "inseguendo" il robot virtuale agganciando la sua traiettoria ed arrivando all'eventuale punto finale

Inseguimento di una traiettoria generica (virtual robot)

Pseudo-codice

while *True* **do**

On each ΔT;

```
// Kinematics
```

...

```
// Odometry
```

...

```
// Position Control
```

$x_{target} \leftarrow \tilde{x}(k\Delta t)$; $y_{target} \leftarrow \tilde{y}(k\Delta t)$;

$k \leftarrow k + 1$;

$position\_error \leftarrow \sqrt{(x_{target} - x)^2 + (y_{target} - y)^2}$;

$v_{target} \leftarrow position\_controller(position\_error)$;

$\theta_{target} \leftarrow \text{atan2}(y_{target} - y, x_{target} - x)$;

$theta\_error \leftarrow \theta_{target} \ominus \theta$;

$\omega_{target} \leftarrow angle\_controller(theta\_error)$;

```
// Speed Control
```

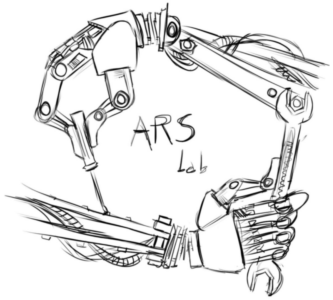
...

end

Cinematica e Controllo di un robot mobile

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



Programmazione Sistemi Robotici