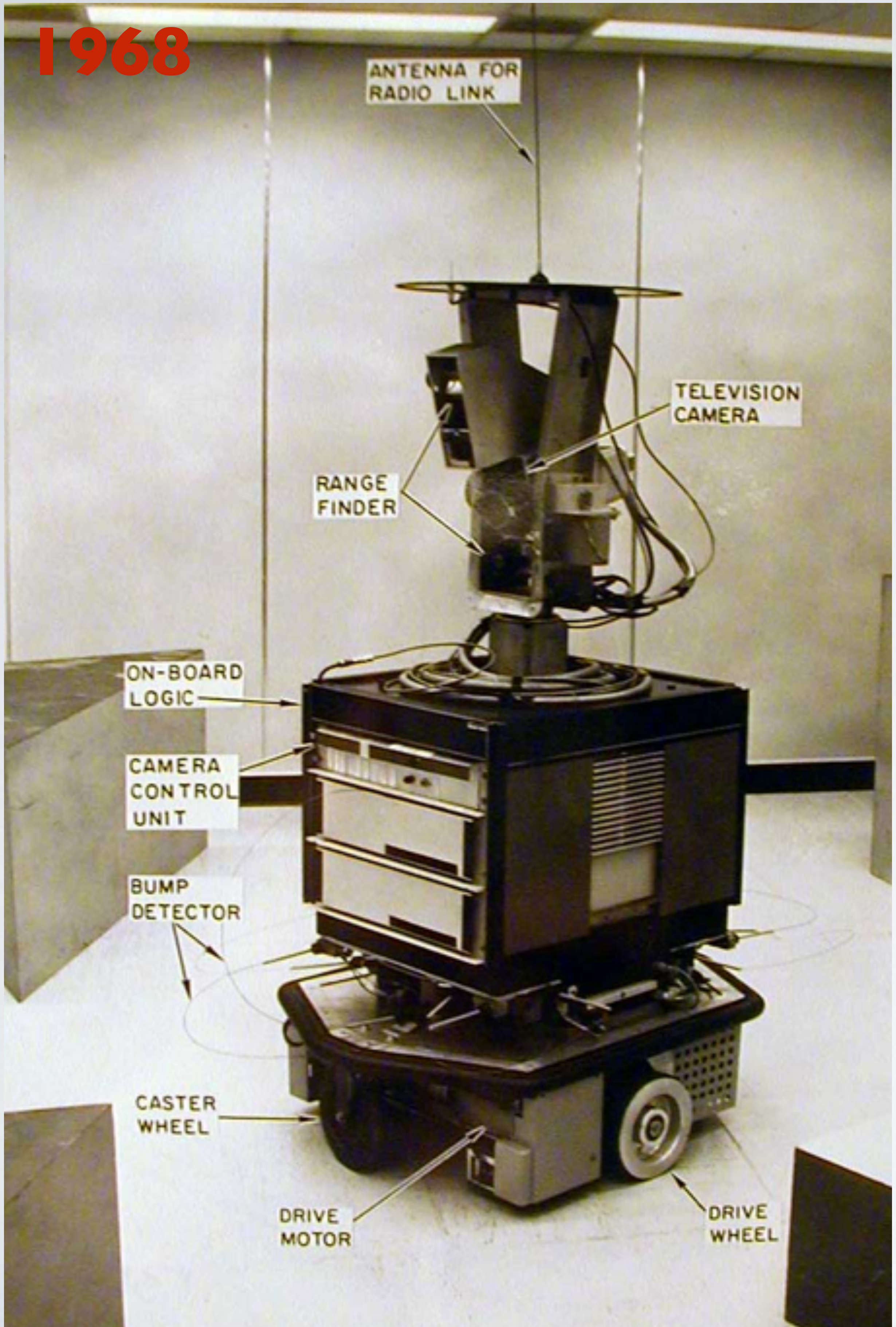
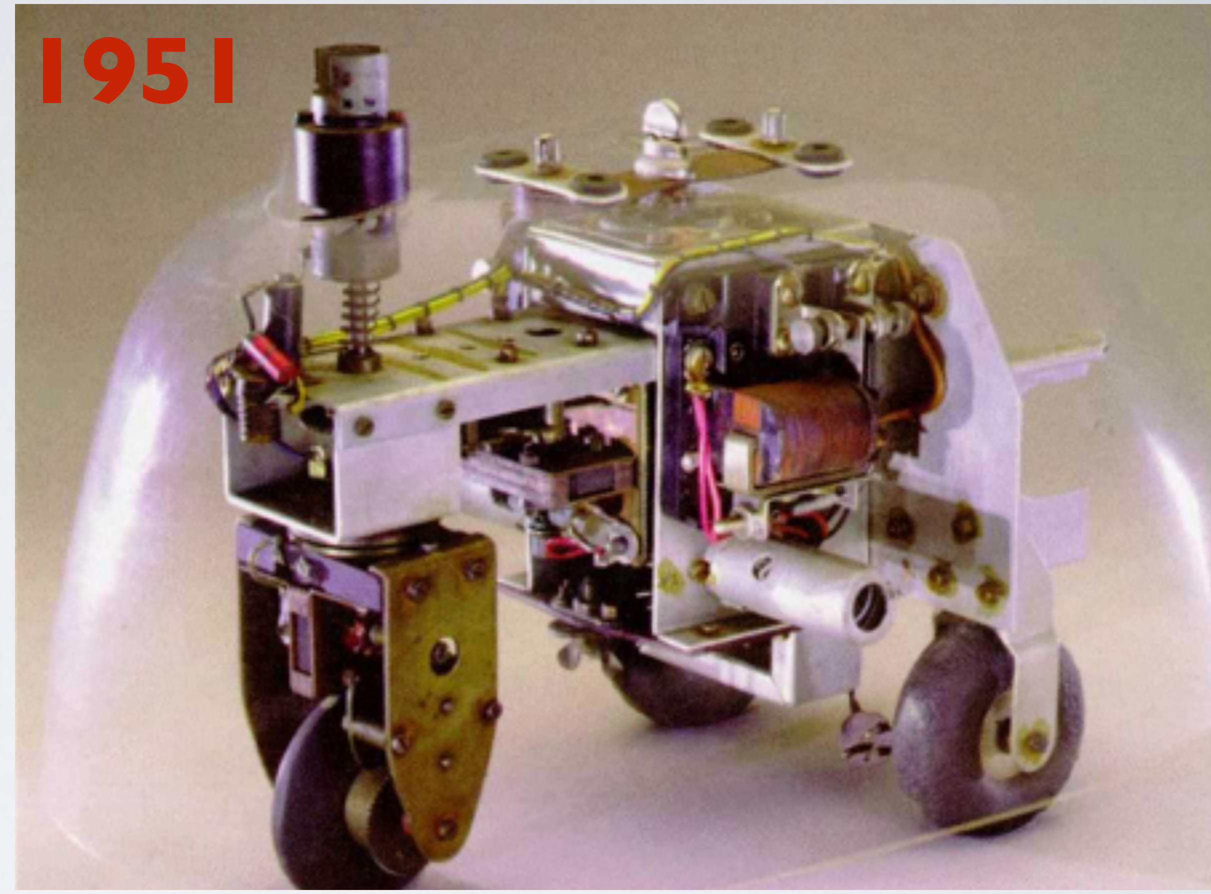
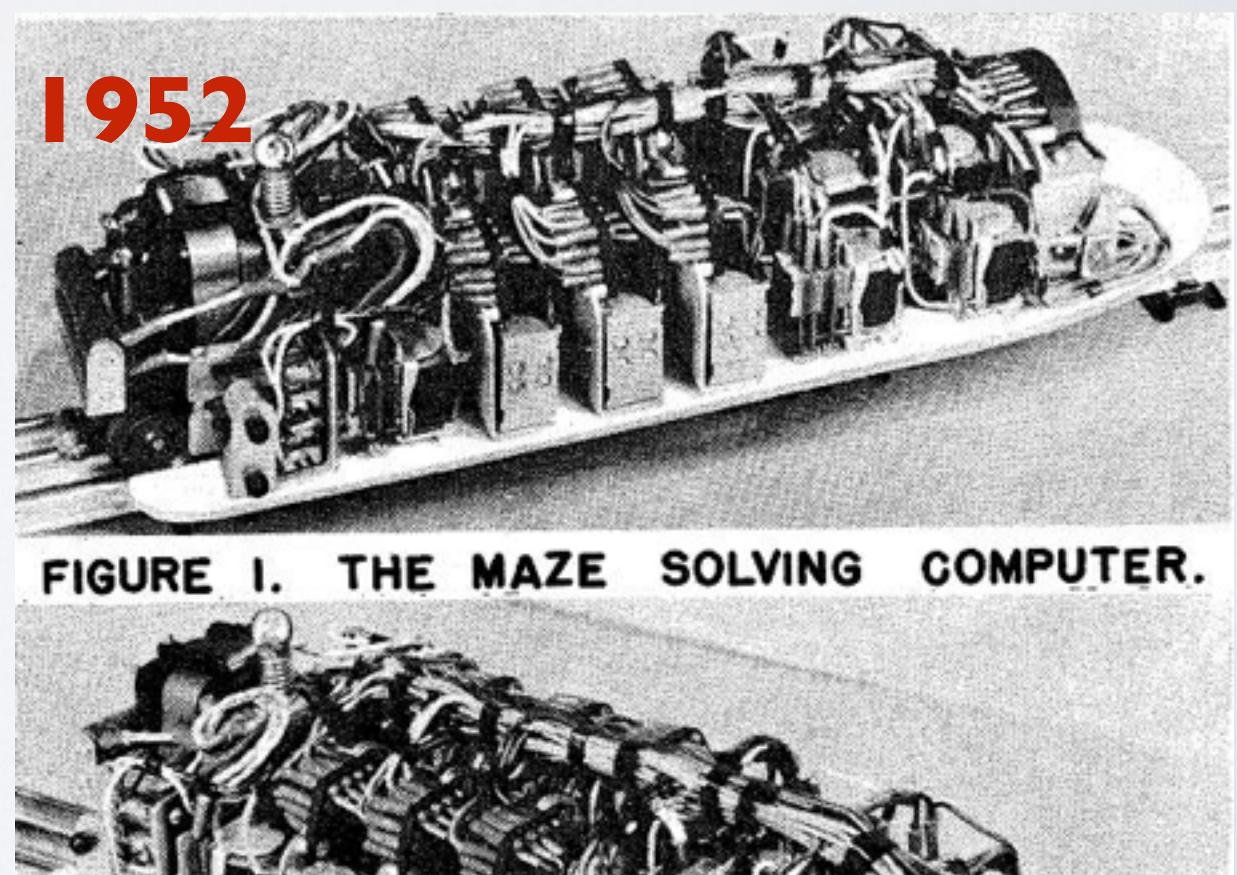
MOBILE ROBOT VEHICLES

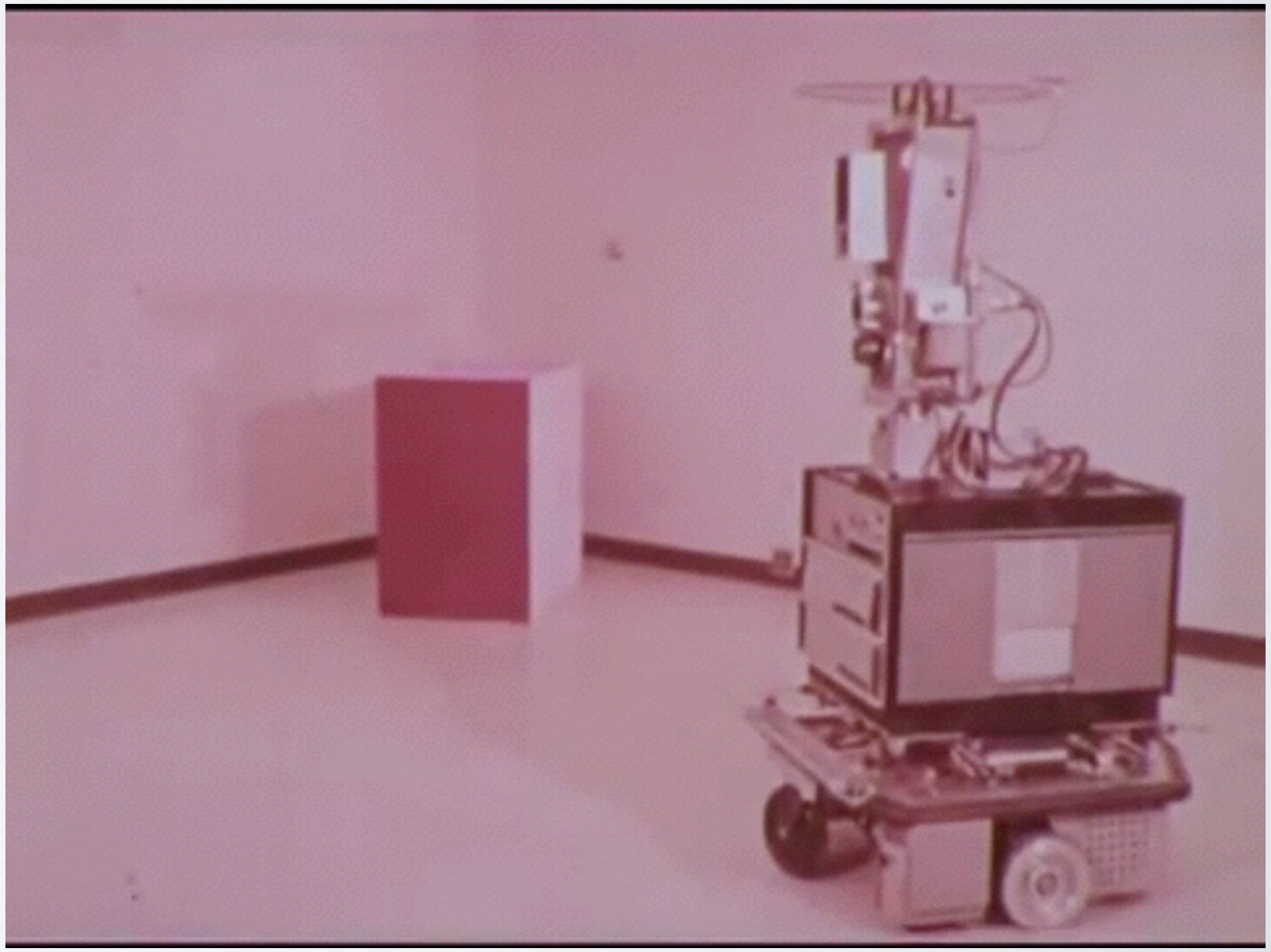
CS 3630 Introduction to Robotics and Perception

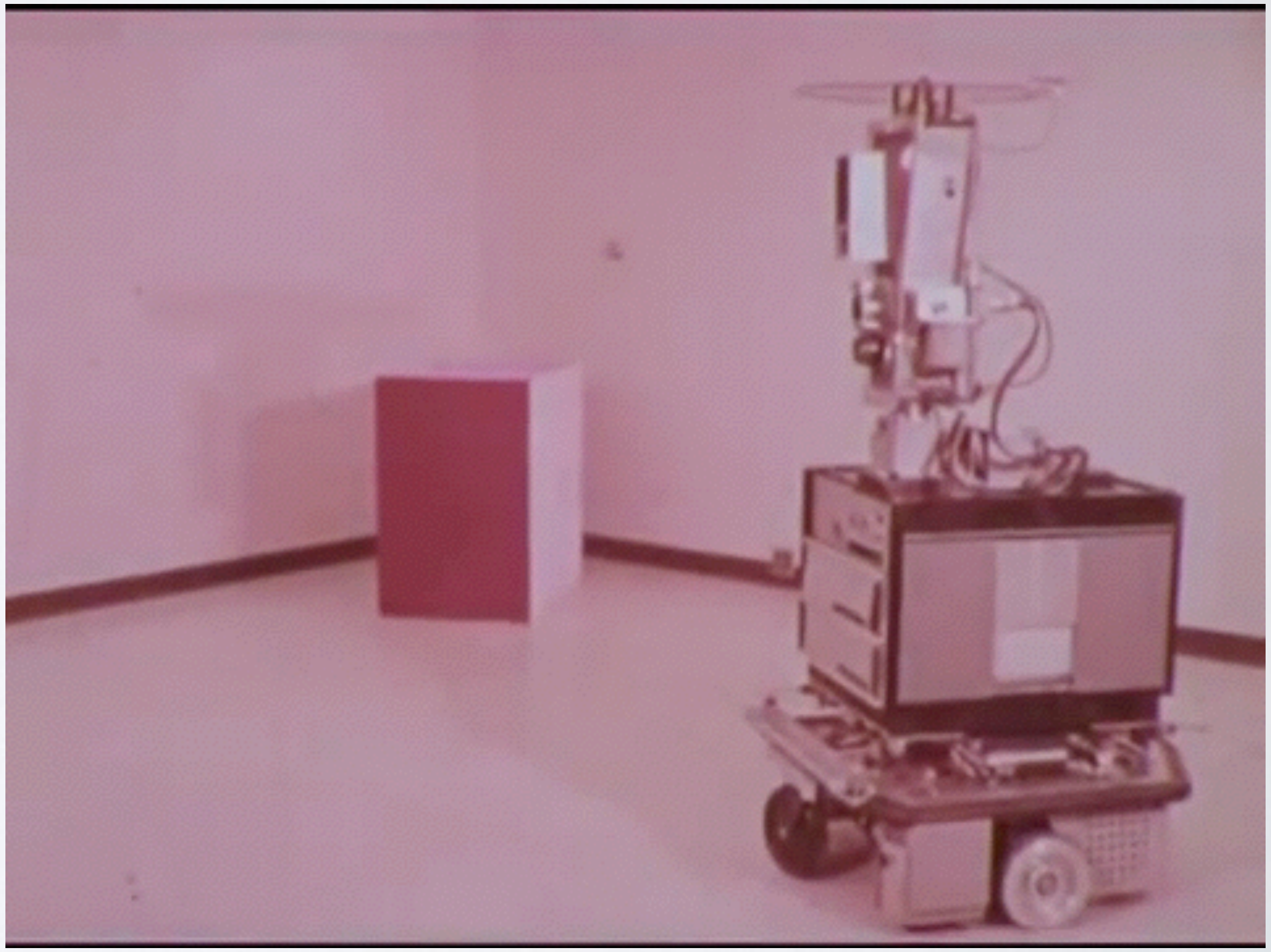
Frank Dellaert

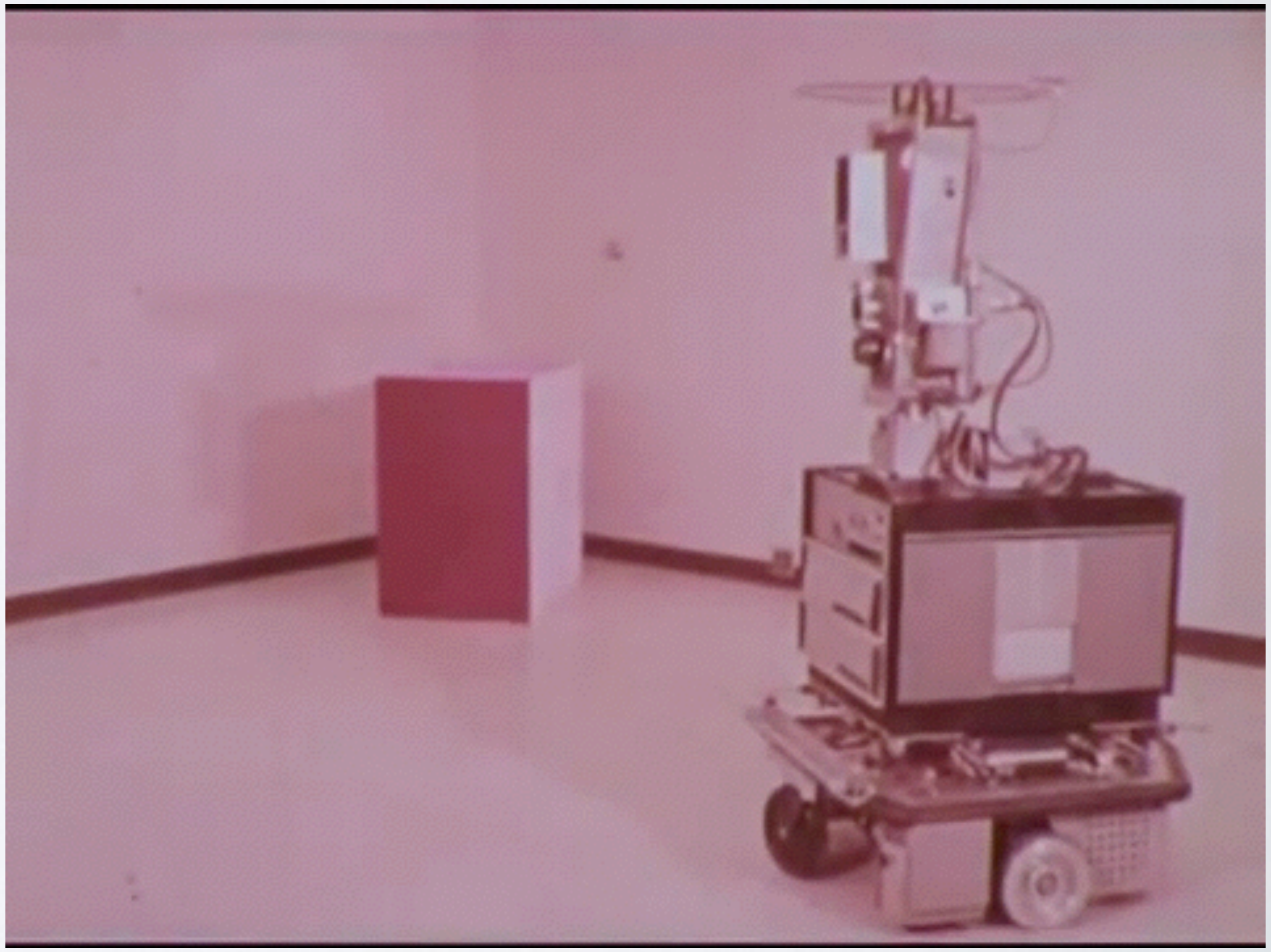

1968
ANTENNA FOR
RADIO LINK
TELEVISION
CAMERA
RANGE
FINDER
ON-BOARD
LOGIC
CAMERA
CONTROL
UNIT
BUMP
DETECTOR
CASTER
WHEEL
DRIVE
MOTOR
DRIVE
WHEEL


1951


1952
FIGURE I. THE MAZE SOLVING COMPUTER.







MORE MODERN AGVS

OTHER MODALITIES

MOBILITY: TRAIN

- Configuration: 1D
- Task Space: R
- C = T

MOBILITY: HOVERCRAFT

- Configuration: (x,y,θ)
- Actuators: 2DOF
- Task Space: SE(2)
- C = T

MOBILITY: HELICOPTER

- Configuration: $(x,y,z,\theta_r,\theta_p,\theta_y)$
- Actuators: 4DOF (thrust,pitch,roll,tail)
- Task Space: SE(3)
- C = T

Boeing A160 Hummingbird

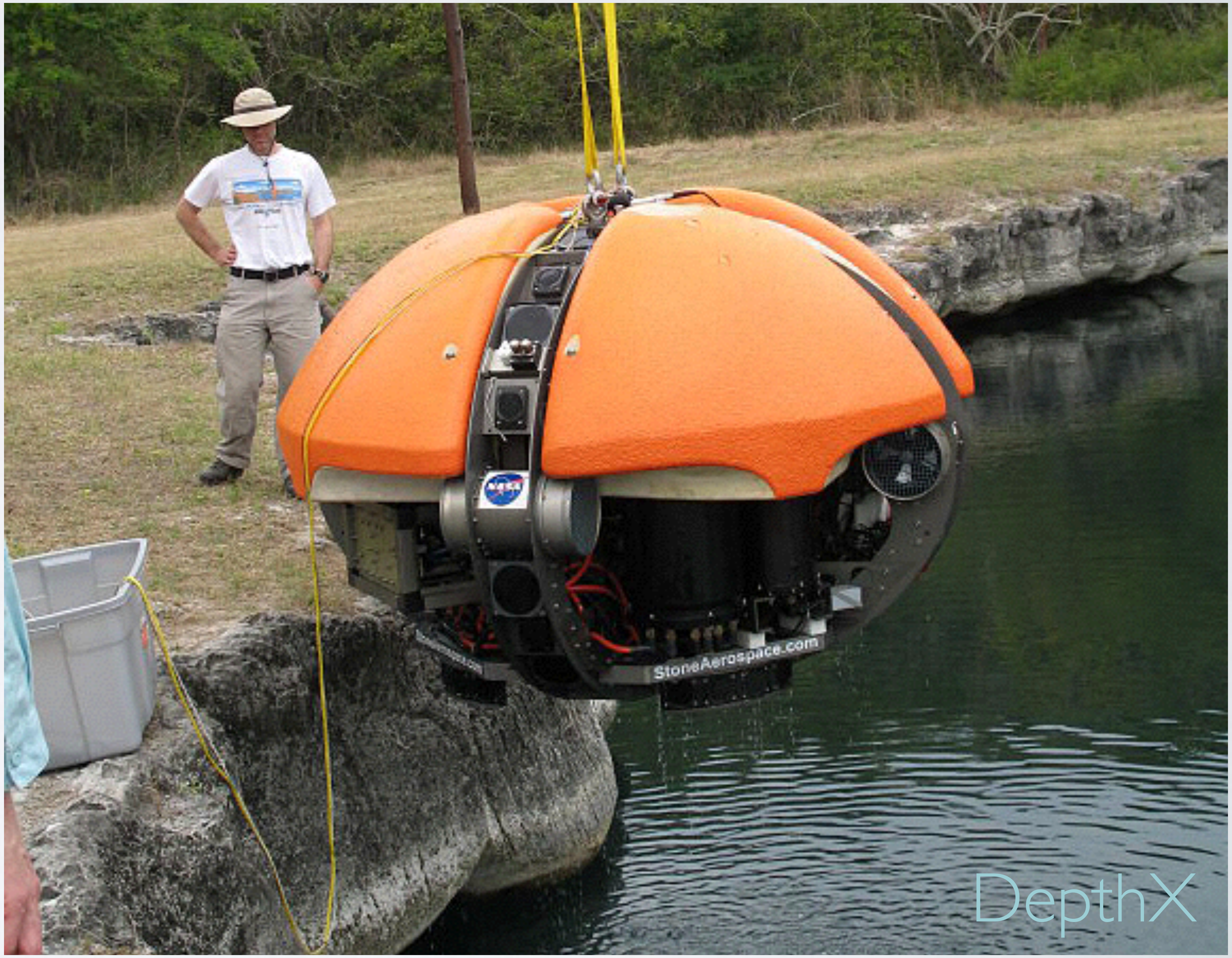
MOBILITY: FIXED WING

- Configuration: $(x,y,z,\theta_r,\theta_p,\theta_y)$
- Actuators: 4DOF (thrust, ail, elev, rud)
- Task Space: SE(3)
- C = T


QF-16

MOBILITY: TRAIN

- Configuration: 6D
- Fully Actuated
- Task Space: SE(3)
- C = T



WHEELS

- Regular wheel
  - Non-holonomic constraint
  - $x' = v, \; y' = 0$
- Omnidirectional wheel
  - No such constraint

MOBILITY RECAP

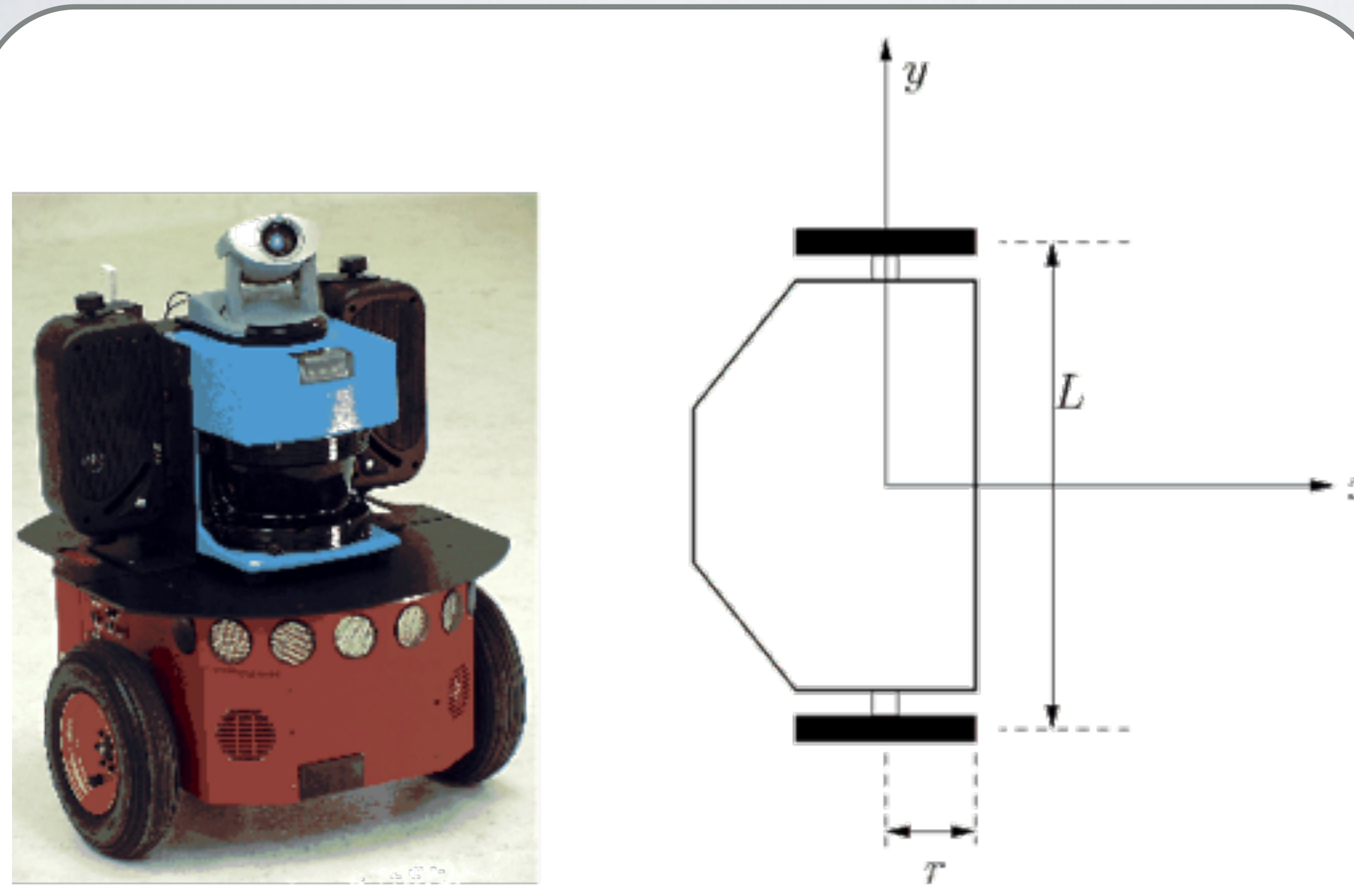
Table 4.1.
Summary of parameters for three different types of vehicle. The $+g$ notation indicates that the gravity field can be considered as an extra actuator

Vehicle	Degrees of freedom	Number of actuators	Fully actuated?
Train	1	1	✓
Hovercraft	3	2	×
Helicopter	6	$4+g$	×
Fixed wing aircraft	6	$4+g$	×
6-thruster AUV	6	6	✓
Car	3	2	×

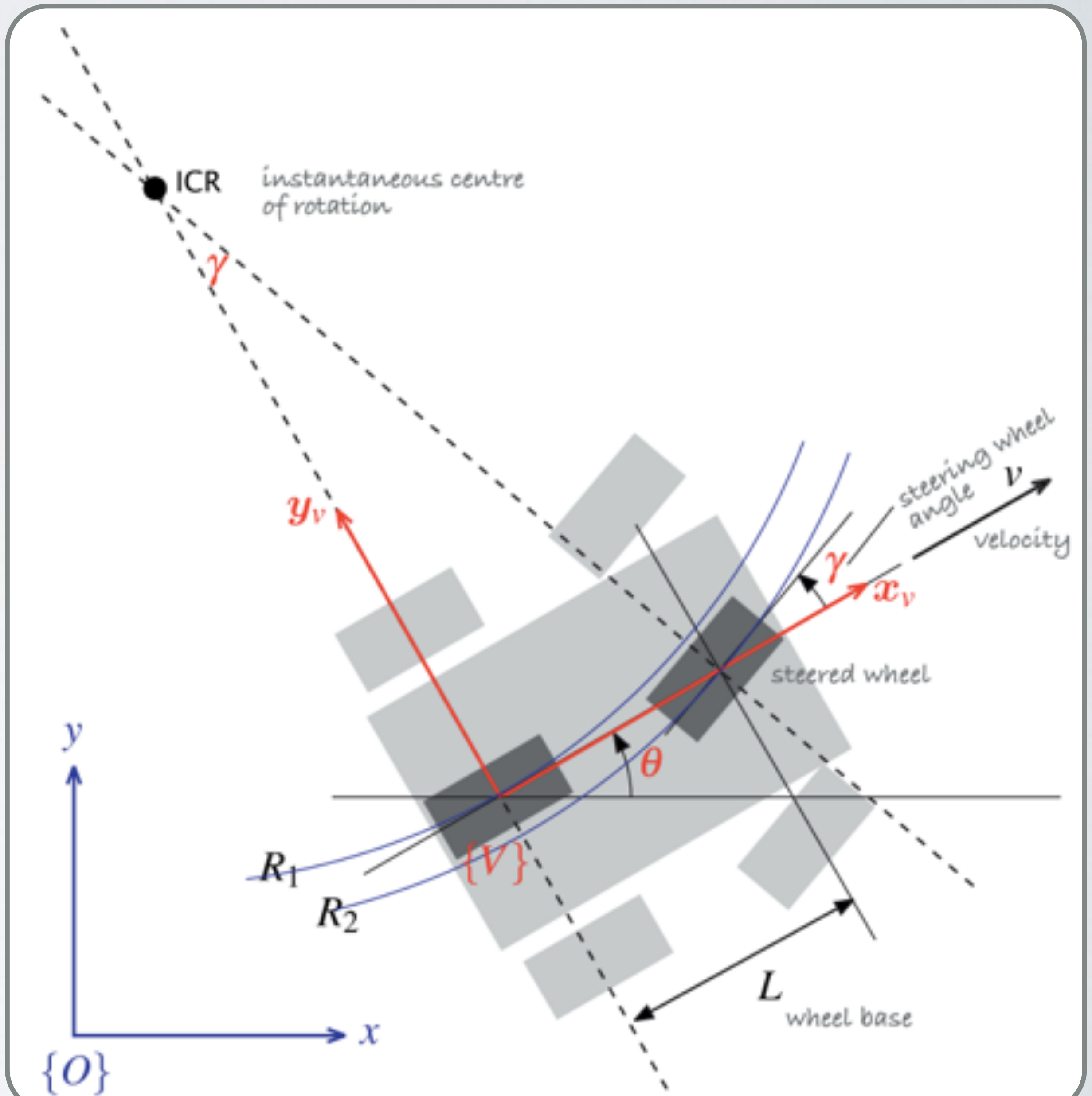
2 KINEMATIC MODELS

BICYCLE MODEL

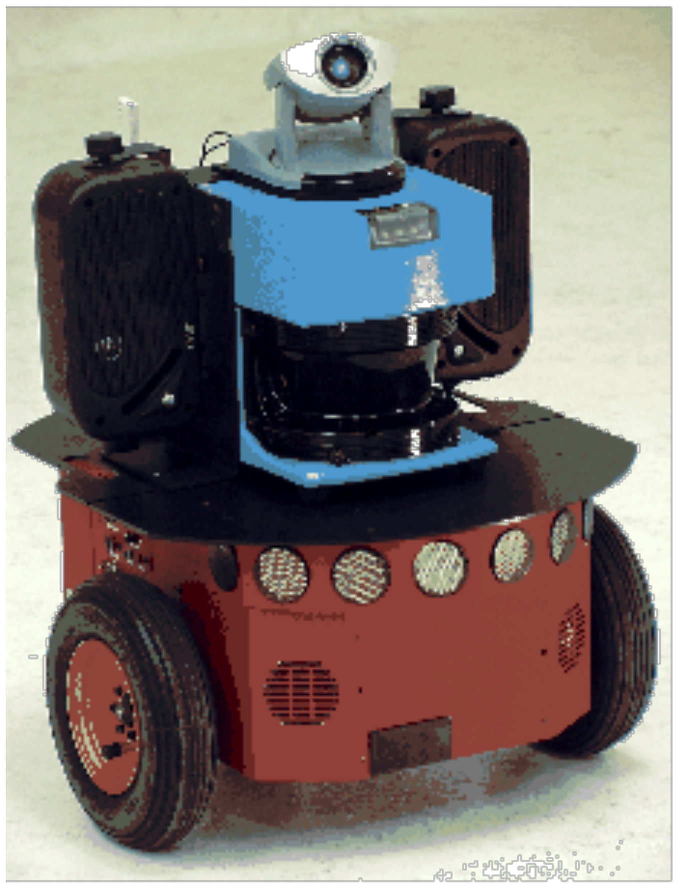
DIFFERENTIAL DRIVE



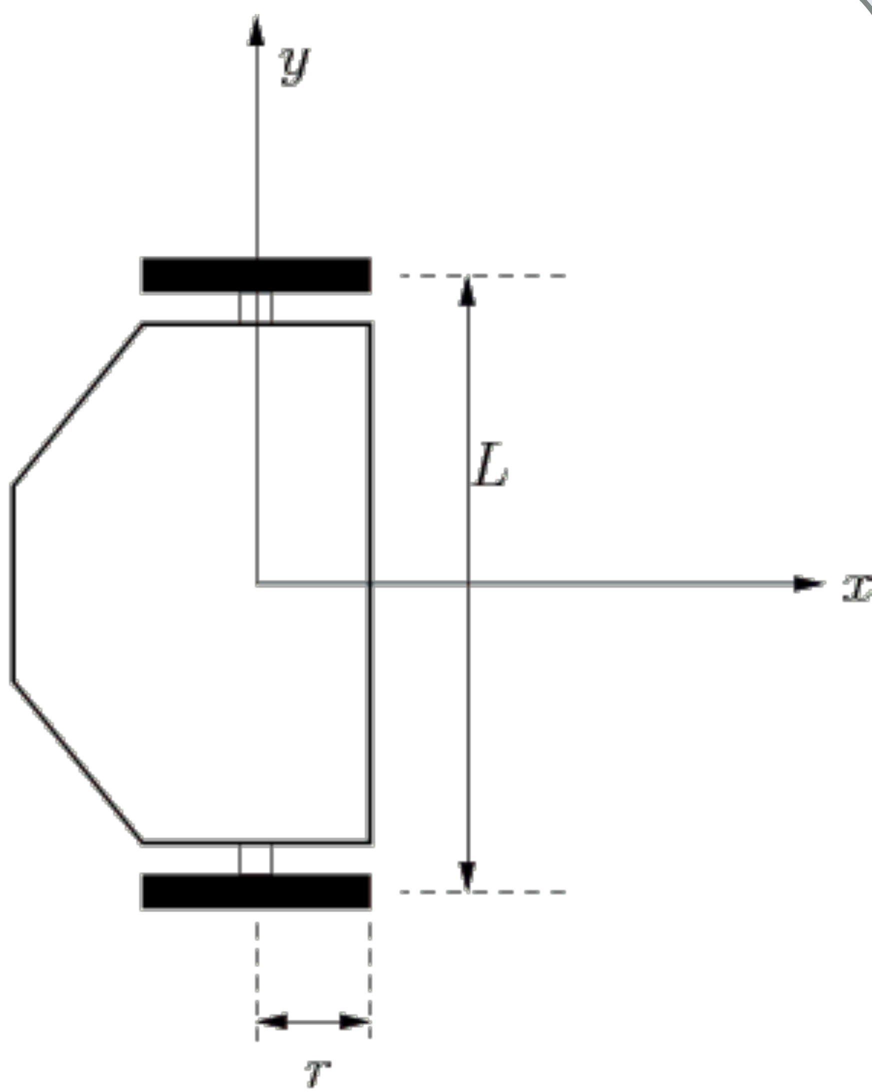
(a) (b)



DIFFERENTIAL DRIVE

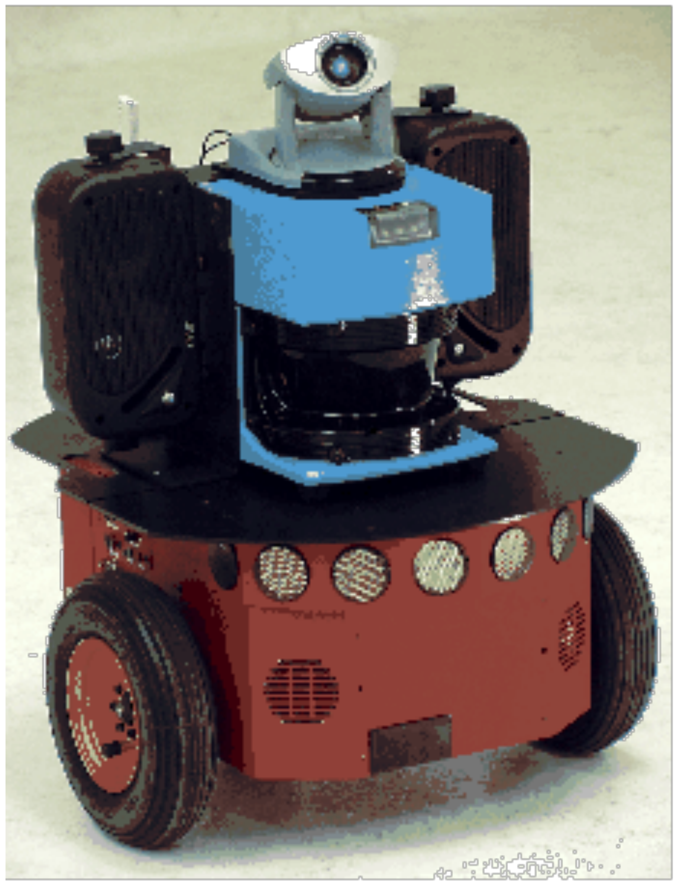


(a)

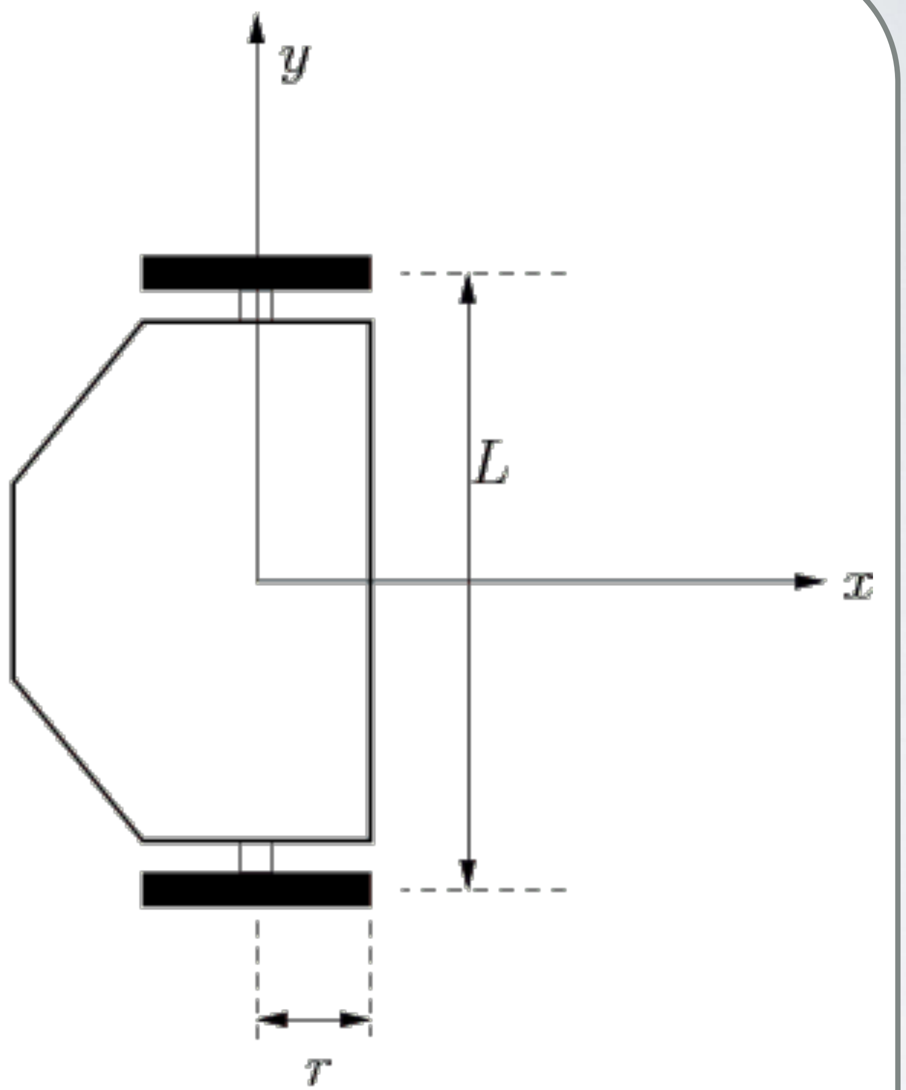


(b)

DIFFERENTIAL DRIVE



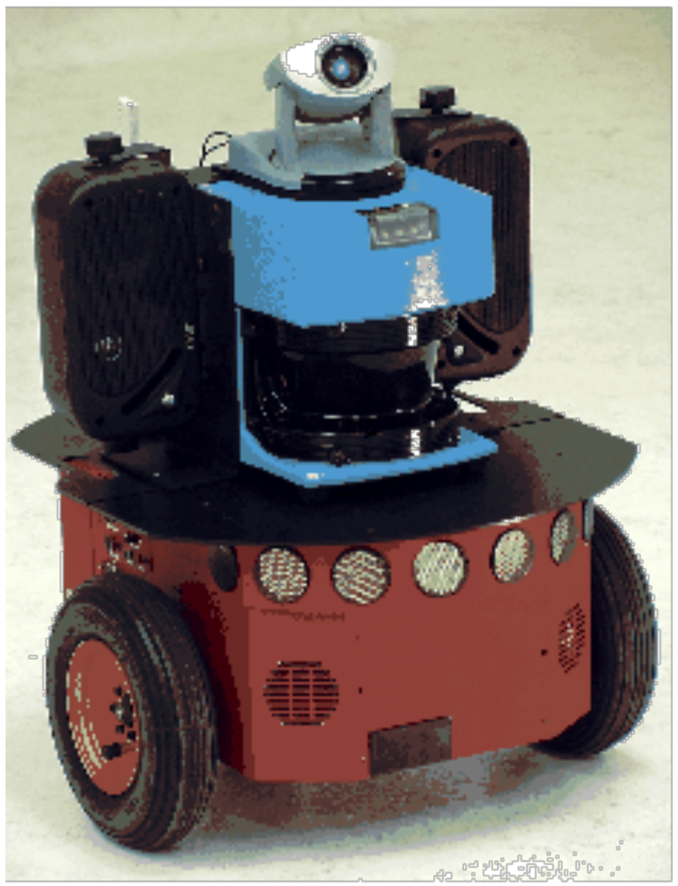
(a)



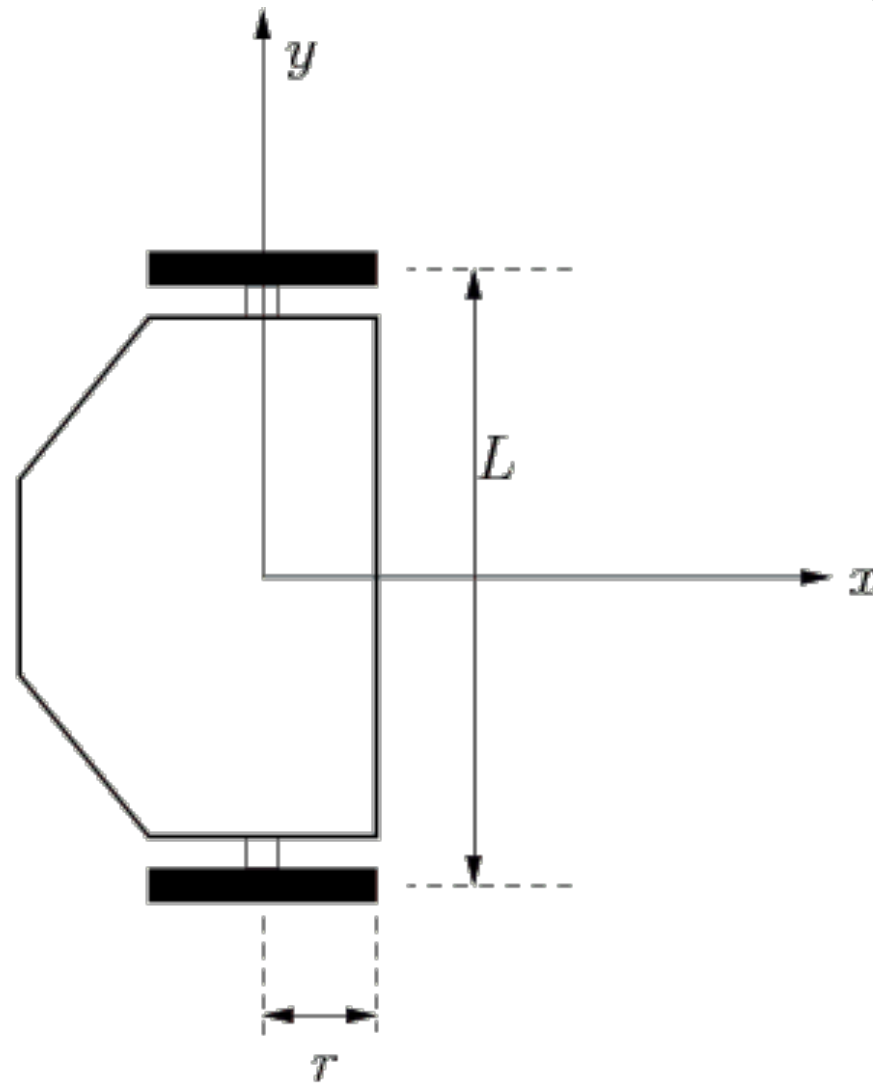
(b)

(this is in the body frame)

DIFFERENTIAL DRIVE



(a)



(b)

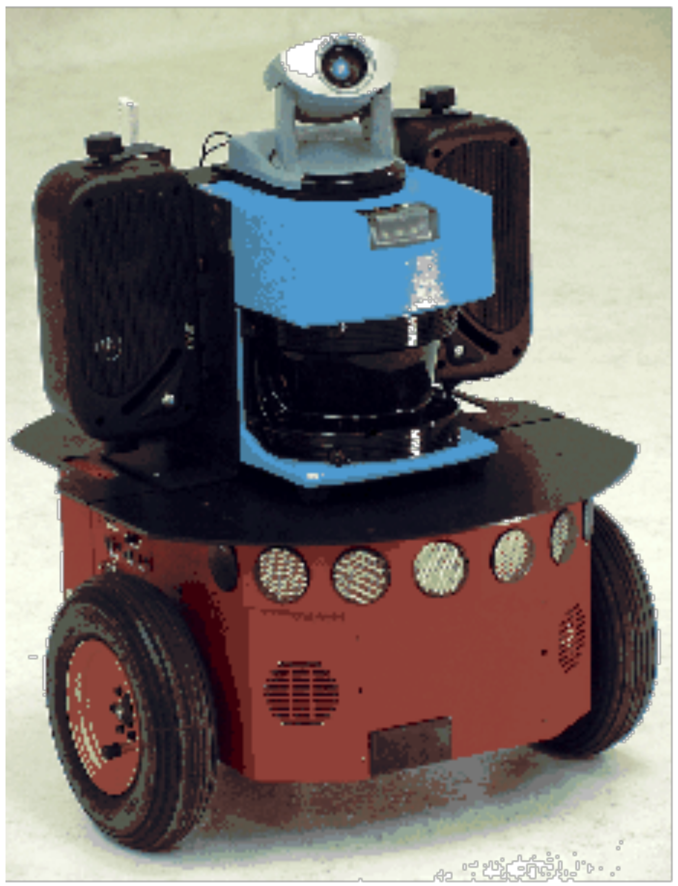
Forward only:

$$\dot{\varphi}_R = \frac{\mathbf{v_x}}{\mathbf{r}}$$

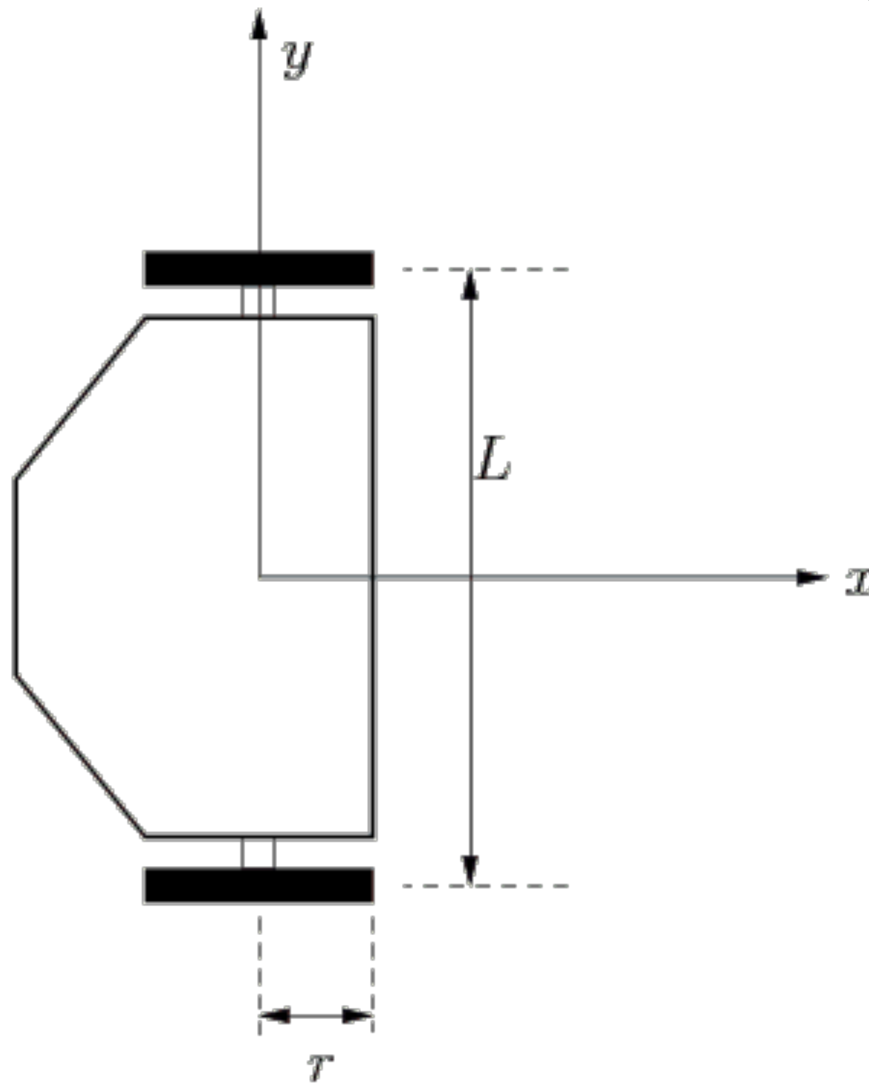
$$\dot{\varphi}_L = \frac{\mathbf{v_x}}{\mathbf{r}}$$

(this is in the body frame)

DIFFERENTIAL DRIVE



(a)



(b)

Forward only:

$$\dot{\varphi}_R = \frac{\mathbf{v_x}}{\mathbf{r}}$$

$$\dot{\varphi}_L = \frac{\mathbf{v_x}}{\mathbf{r}}$$

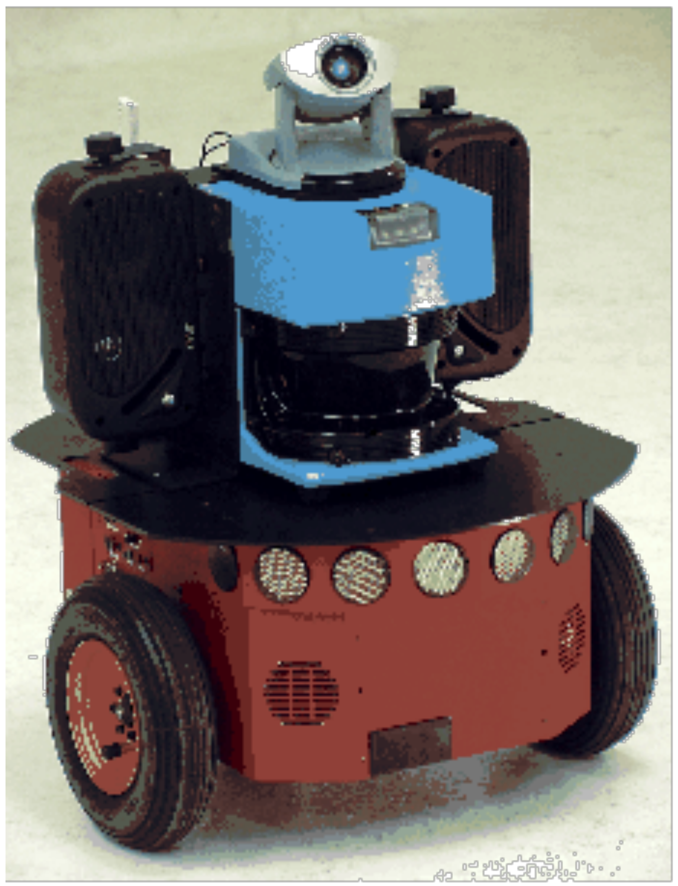
Rotation only:

$$\dot{\varphi}_R = \frac{\omega \mathrm{L}}{2\mathrm{r}}$$

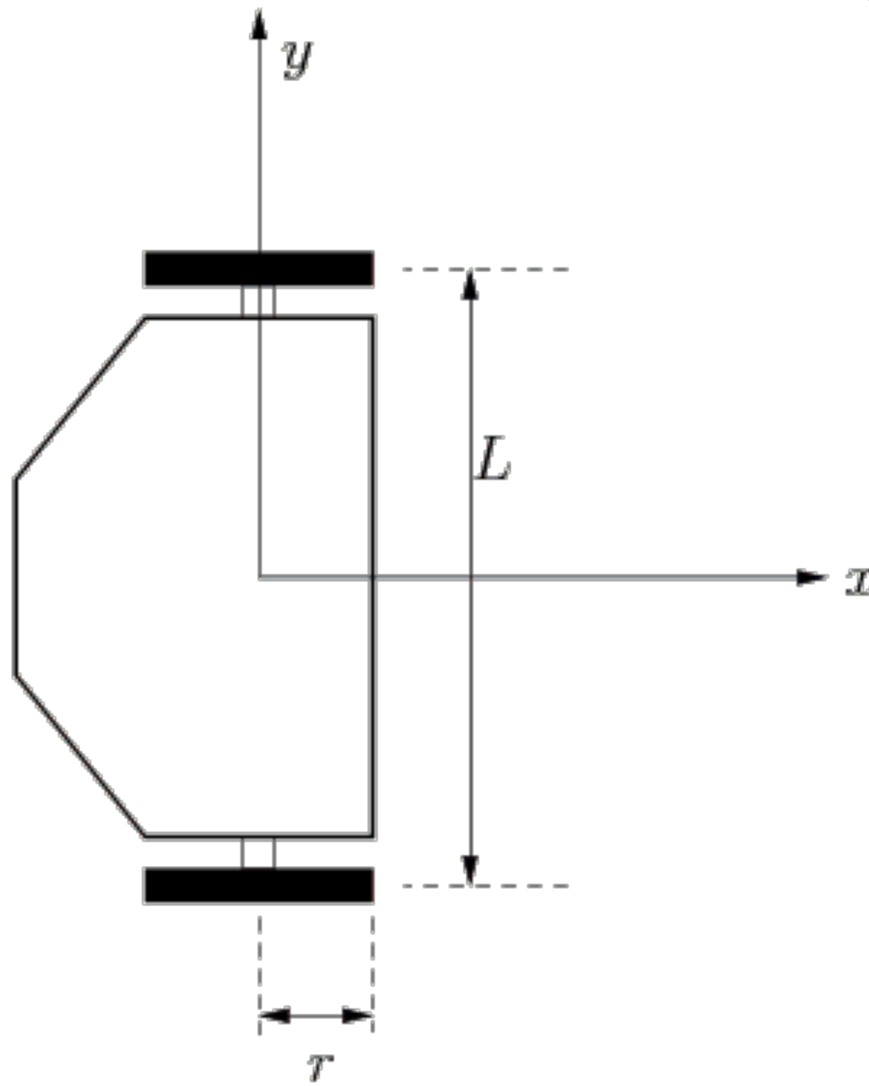
$$\dot{\varphi}_L = -\frac{\omega L}{2r}$$

(this is in the body frame)

DIFFERENTIAL DRIVE



(a)



(b)

Forward only:

$$\dot{\varphi}_R = \frac{\mathrm{v_x}}{\mathrm{r}}$$

$$\dot{\varphi}_L = \frac{\mathrm{v_x}}{\mathrm{r}}$$

Rotation only:

$$\dot{\varphi}_R = \frac{\omega \mathrm{L}}{2\mathrm{r}}$$

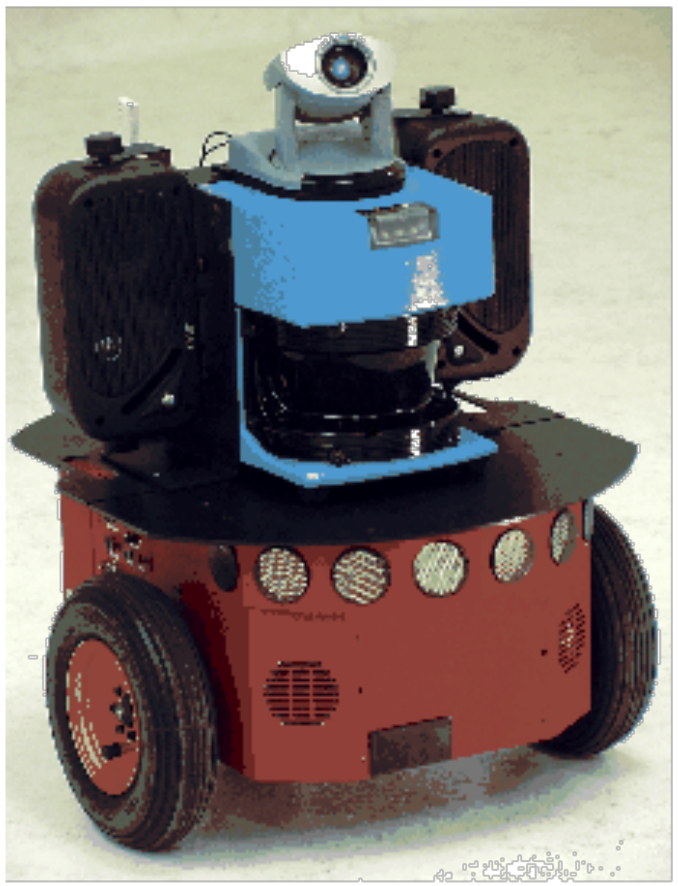
$$\dot{\varphi}_L = -\frac{\omega L}{2r}$$

Both:

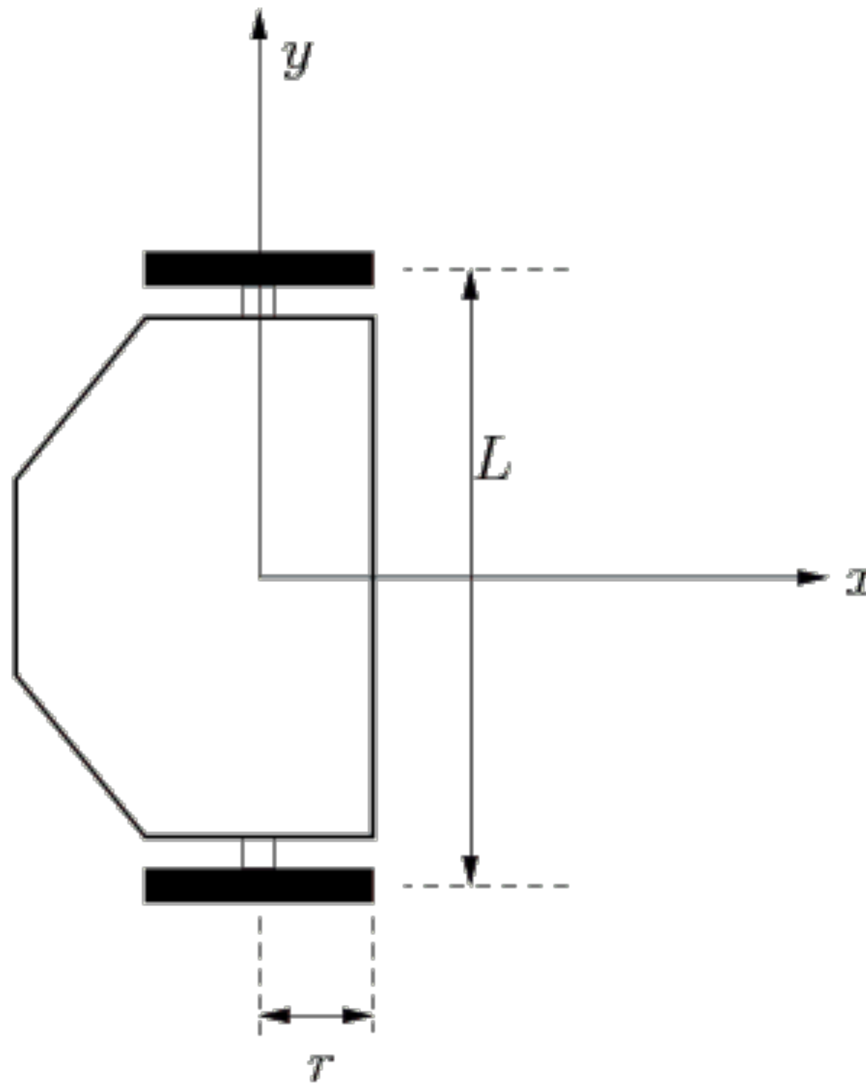
$$\boxed{\begin{aligned} \dot{\varphi}_R &= \frac{\omega \mathrm{L}}{2\mathrm{r}} + \frac{\mathrm{v_x}}{\mathrm{r}} \\ \dot{\varphi}_L &= -\frac{\omega L}{2r} + \frac{\mathrm{v_x}}{\mathrm{r}} \end{aligned}}$$

(this is in the body frame)

DIFFERENTIAL DRIVE



(a)



(b)

$$\dot{\xi}_R = \begin{bmatrix} \frac{r}{2}\left(\dot{\varphi}_R + \dot{\varphi}_L\right) \\ 0 \\ \frac{r}{L}\left(\dot{\varphi}_R - \dot{\varphi}_L\right) \end{bmatrix}$$

(this is in the body frame)

Forward only:

$$\dot{\varphi}_R = \frac{\mathrm{v_x}}{\mathrm{r}}$$

$$\dot{\varphi}_L = \frac{\mathrm{v_x}}{\mathrm{r}}$$

Rotation only:

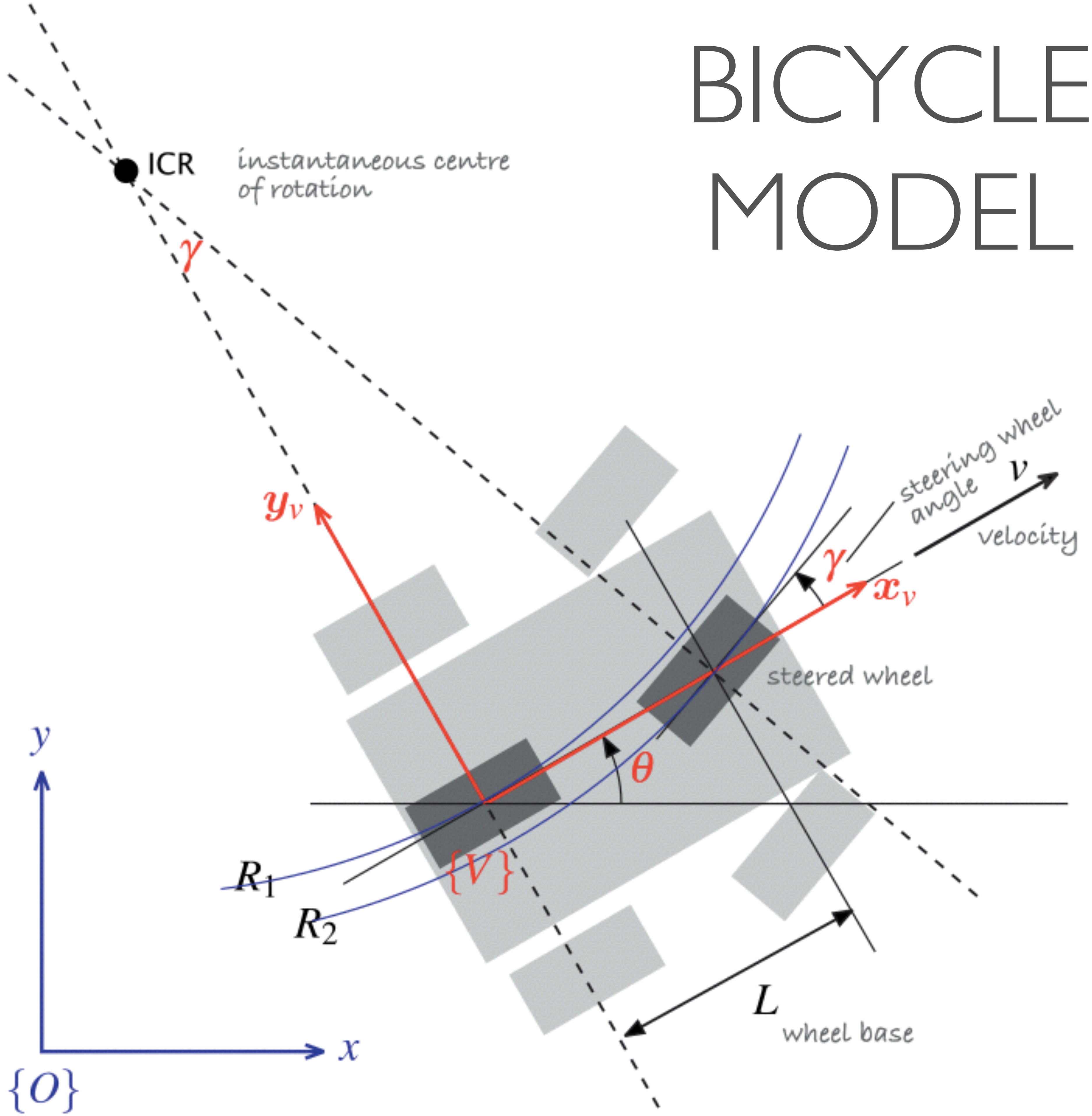
$$\dot{\varphi}_R = \frac{\omega \mathrm{L}}{2\mathrm{r}}$$

$$\dot{\varphi}_L = -\frac{\omega L}{2r}$$

Both:

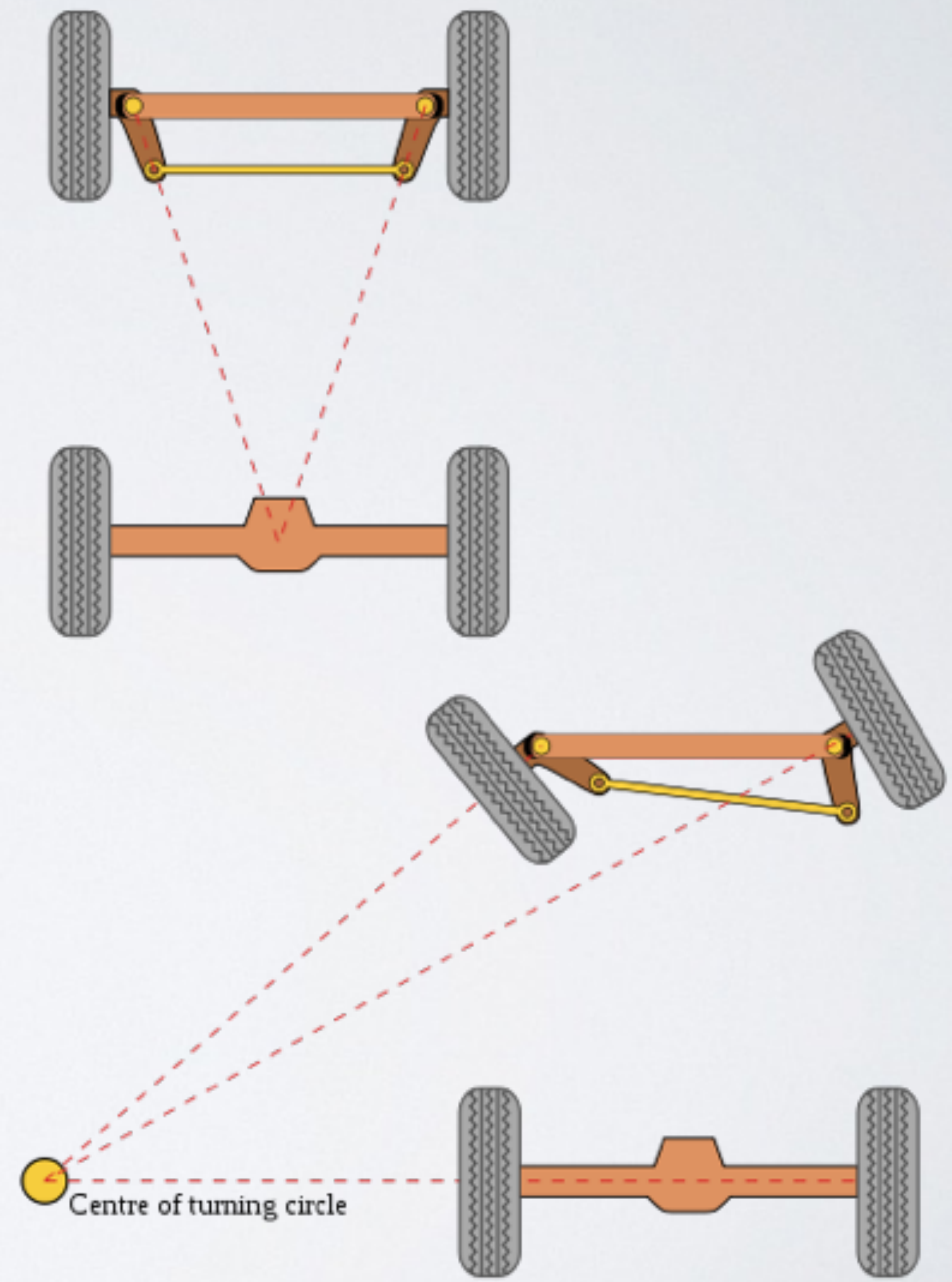
$$\boxed{\dot{\varphi}_R = \frac{\omega \mathrm{L}}{2\mathrm{r}} + \frac{\mathrm{v_x}}{\mathrm{r}} \qquad \dot{\varphi}_L = -\frac{\omega L}{2r} + \frac{\mathrm{v_x}}{\mathrm{r}}}$$

BICYCLE MODEL



ACKERMAN STEERING

- Four-wheeled vehicle
- L and R move on circular paths of different radius
- 1812 patent



EQUATIONS OF MOTION (SEE BOOK)

- Angular velocity
  ~ steering angle γ
- Angular velocity
  ~ velocity *v*
- Non-holonomic constraint
- Undefined for 90° angle

$$\dot{x} = v\cos\theta$$

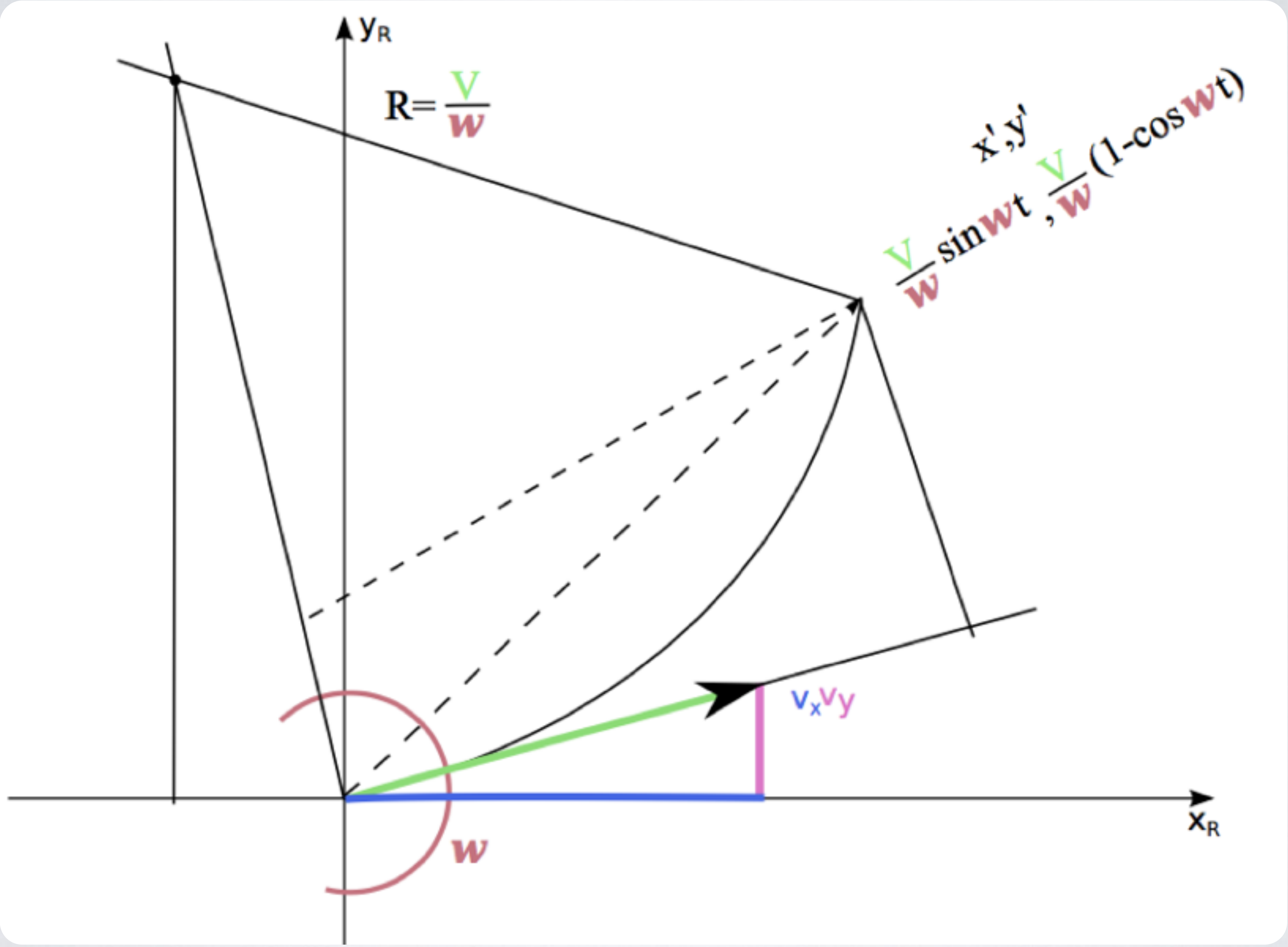
$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \frac{v}{L}\tan\gamma$$

$$\dot{y}\cos\theta - \dot{x}\sin\theta \equiv 0$$

(this is in the world frame)

2D TWIST



2D TWIST

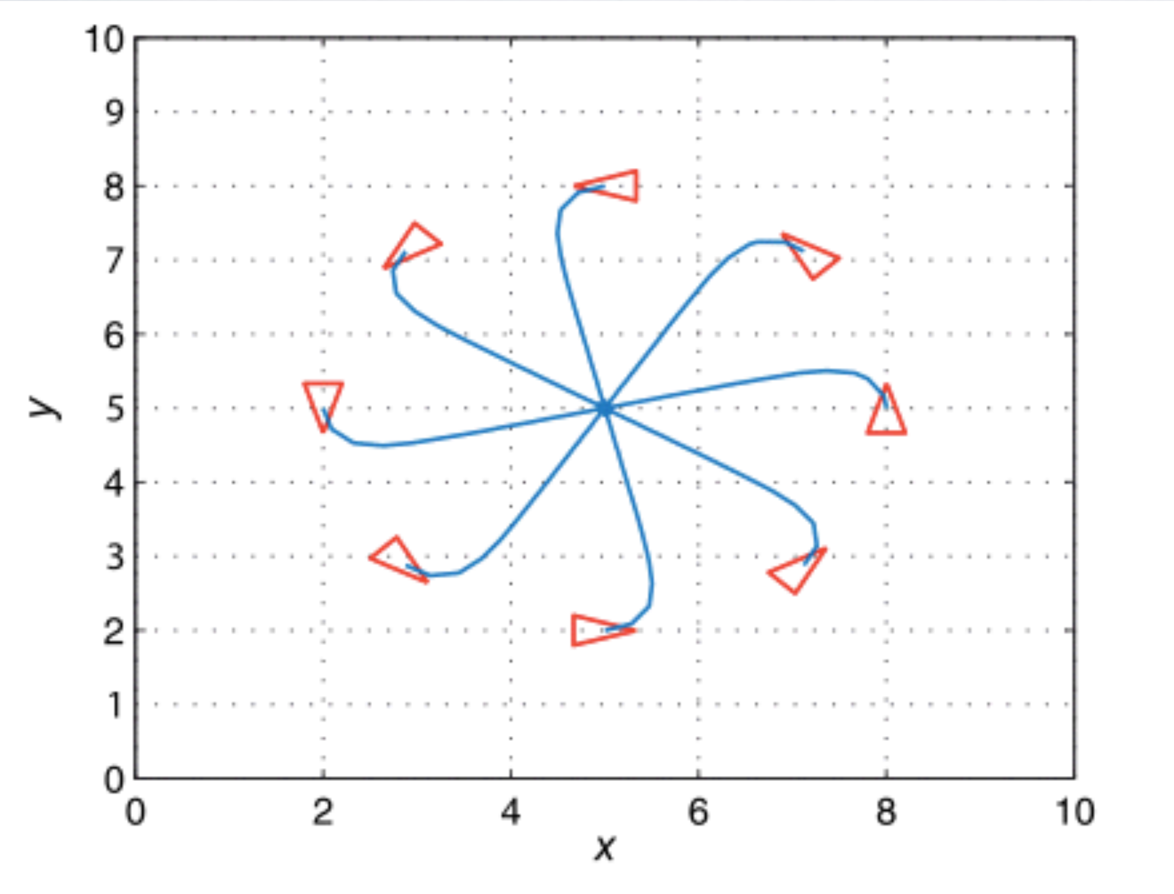
A 2D twist is simply the derivative of a 2D rigid transformation

$$\dot{\xi}_R^W = \begin{bmatrix} \dot{t}_x \\ \dot{t}_y \\ \dot{\theta} \end{bmatrix}$$

A robot undergoing a constant twist, expressed in the robot frame, traces out a circular trajectory with radius $R = v/\omega$. Starting from the origin, after some time T we obtain

$$\xi(T) = \left(\begin{bmatrix} \cos\omega T & -\sin\omega T \\ \sin\omega T & \cos\omega T \end{bmatrix}, \frac{1}{\omega} \begin{bmatrix} 1 - \cos\omega T & \sin\omega T \\ -\sin\omega T & 1 - \cos\omega T \end{bmatrix} \begin{bmatrix} -v_y \\ v_x \end{bmatrix} \right)$$

MOVING TO A POINT

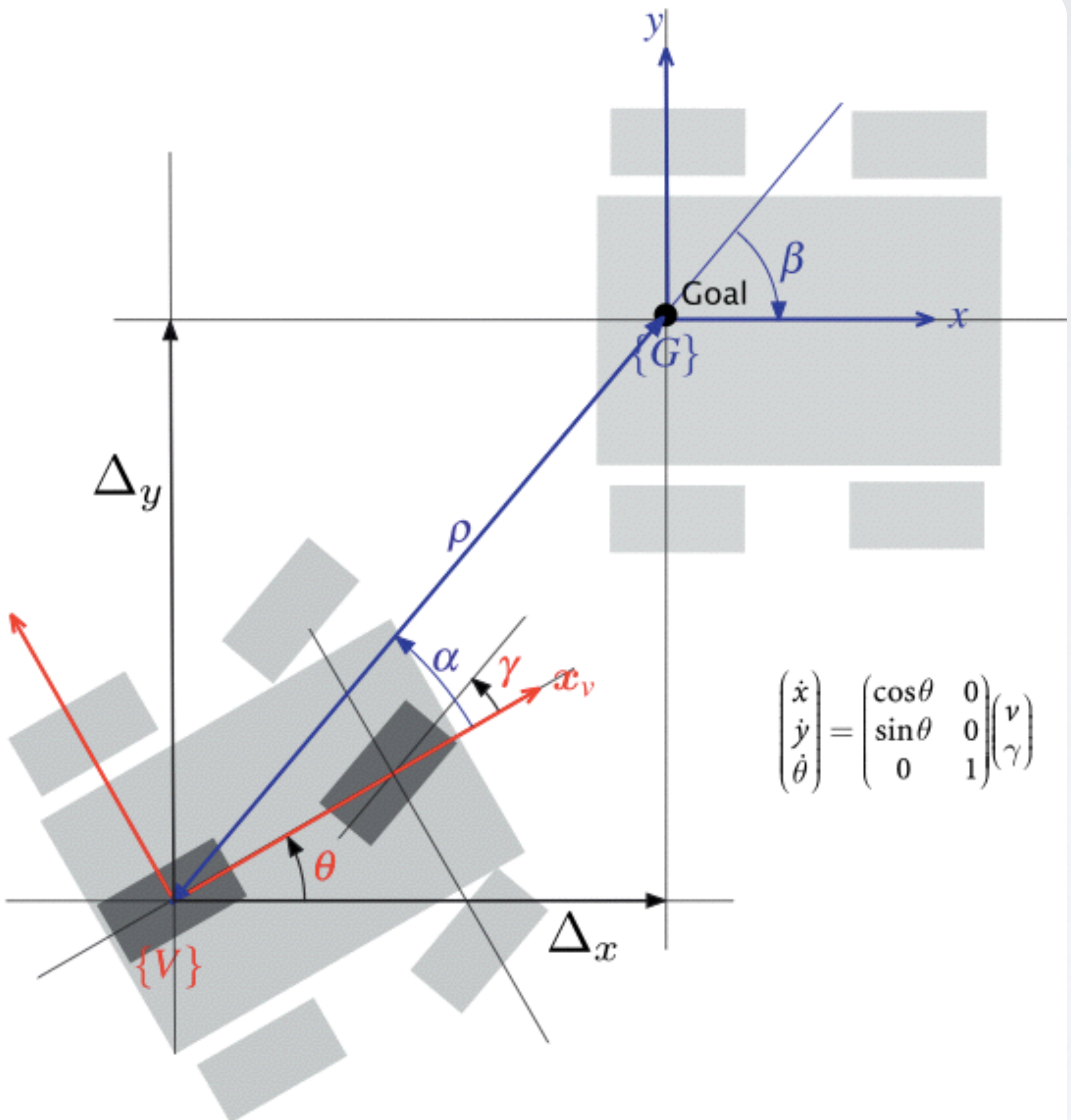


$$v^* = K_v \sqrt{(x^* - x)^2 + (y^* - y)^2}$$

$$\theta^* = \tan^{-1} \frac{y^* - y}{x^* - x}$$

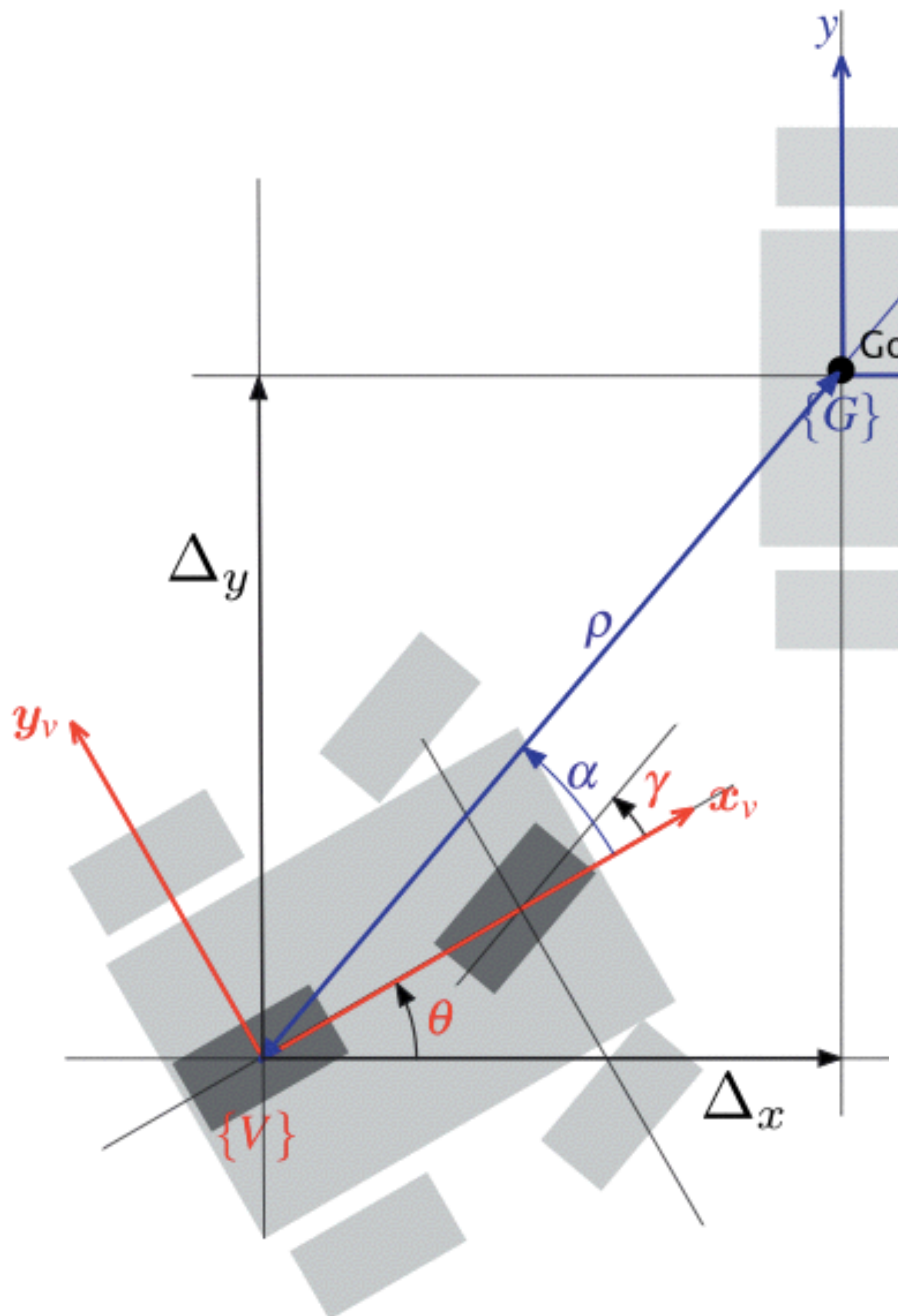
$$\gamma = K_h(\theta^* \ominus \theta), \quad K_h > 0$$

MOVING TO A POSE



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

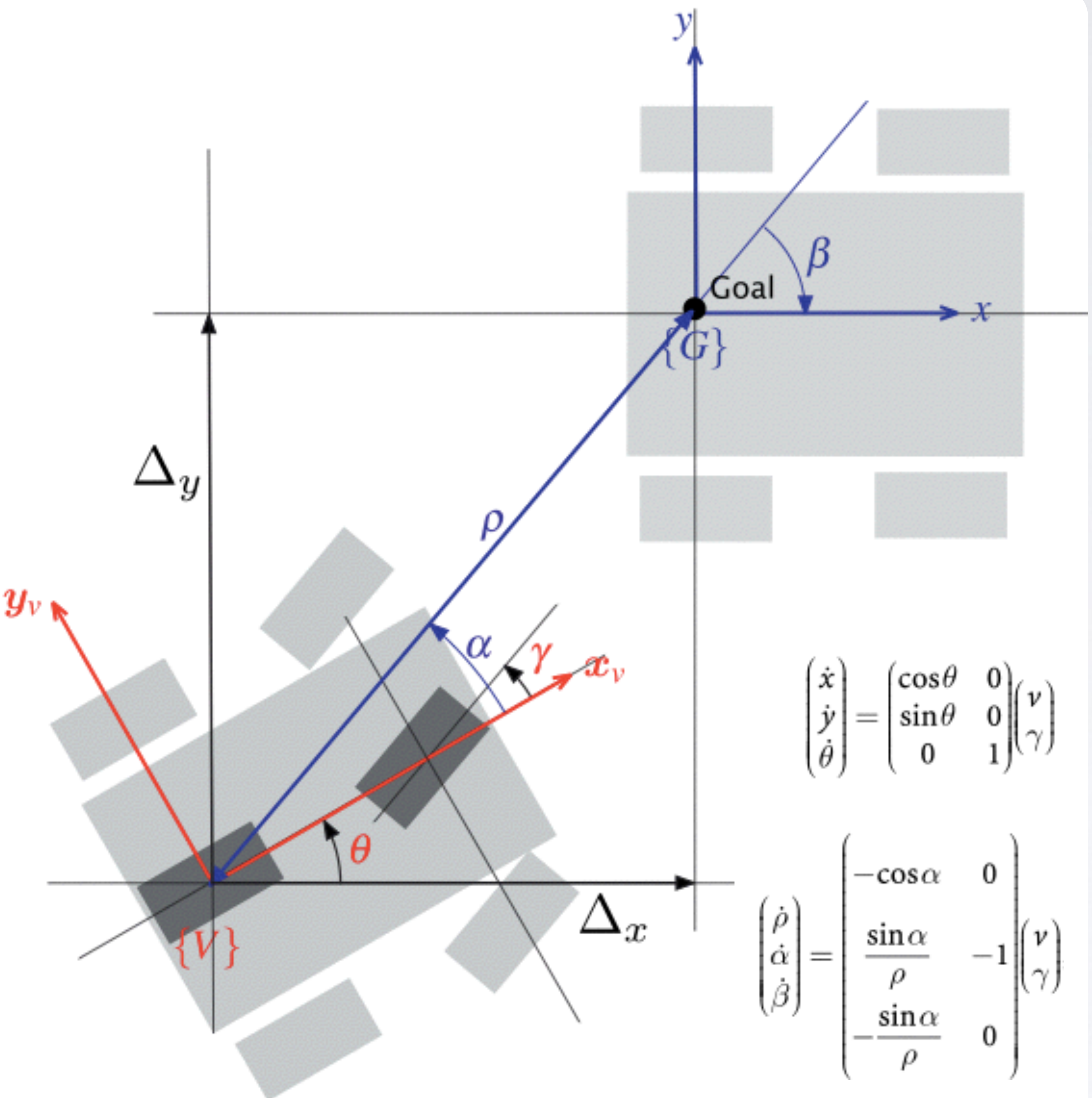
MOVING TO A POSE



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

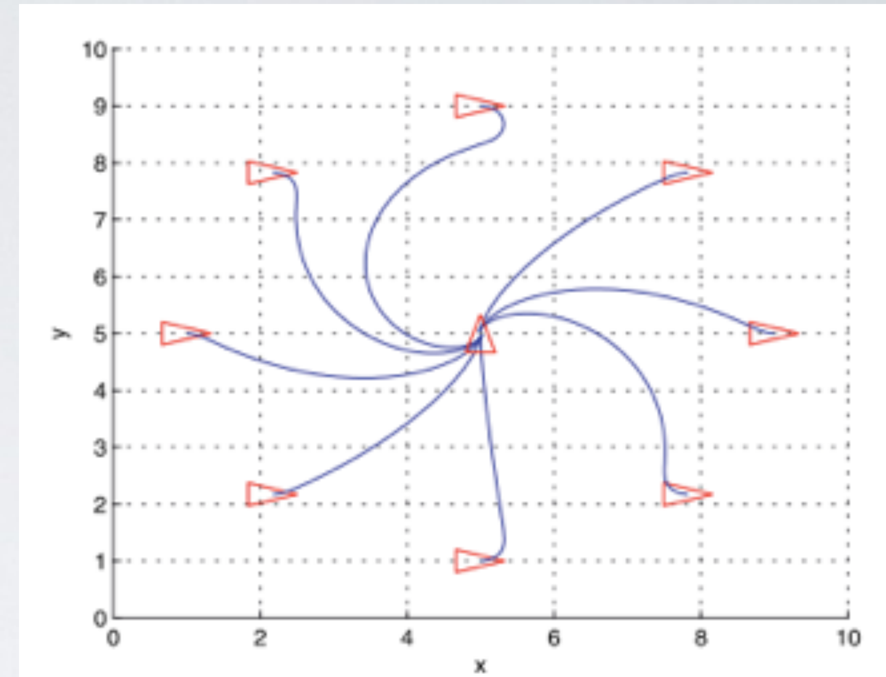
$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{pmatrix} \begin{pmatrix} v \\ \gamma \end{pmatrix}$$

MOVING TO A POSE



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

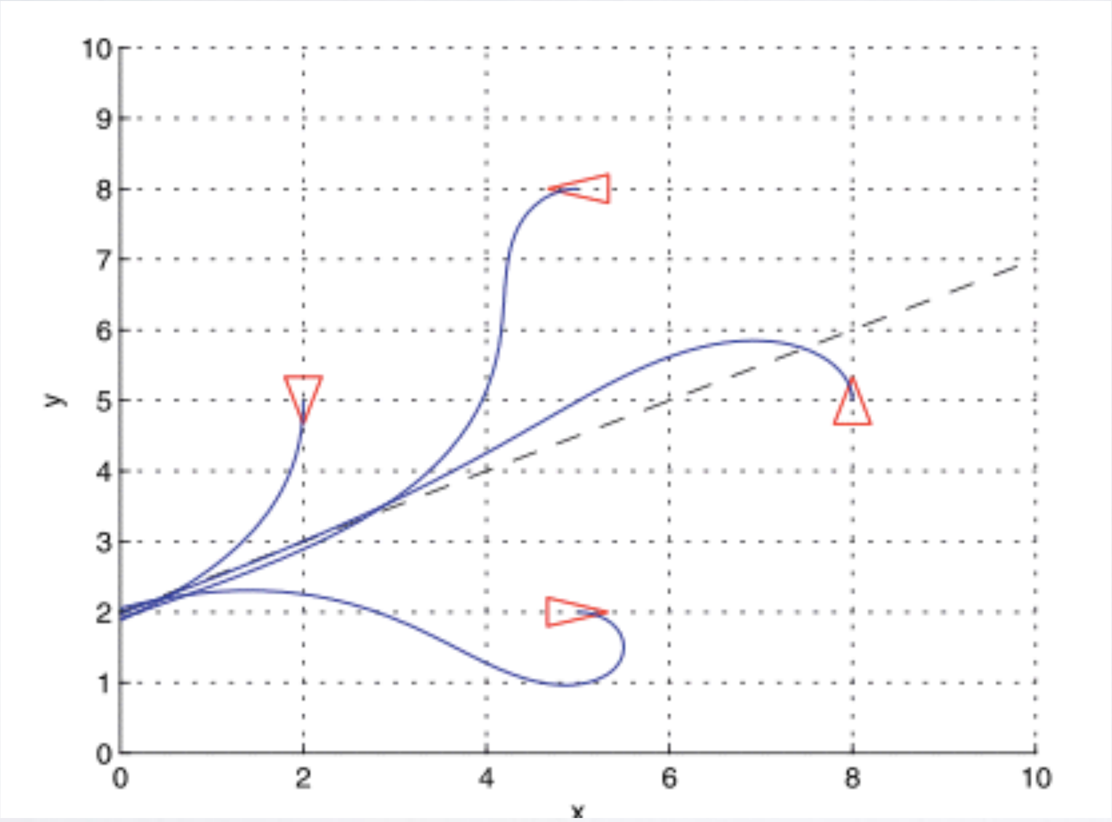
$$\begin{pmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{pmatrix} = \begin{pmatrix}-\cos\alpha & 0\\ \dfrac{\sin\alpha}{\rho} & -1\\ -\dfrac{\sin\alpha}{\rho} & 0\end{pmatrix}\begin{pmatrix}v\\ \gamma\end{pmatrix}$$



$$v = k_\rho \rho$$
$$\gamma = k_\alpha \alpha + k_\beta \beta$$

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

FOLLOWING A LINE



$$d = \frac{(a,b,c)\cdot(x,y,1)}{\sqrt{a^2+b^2}}$$

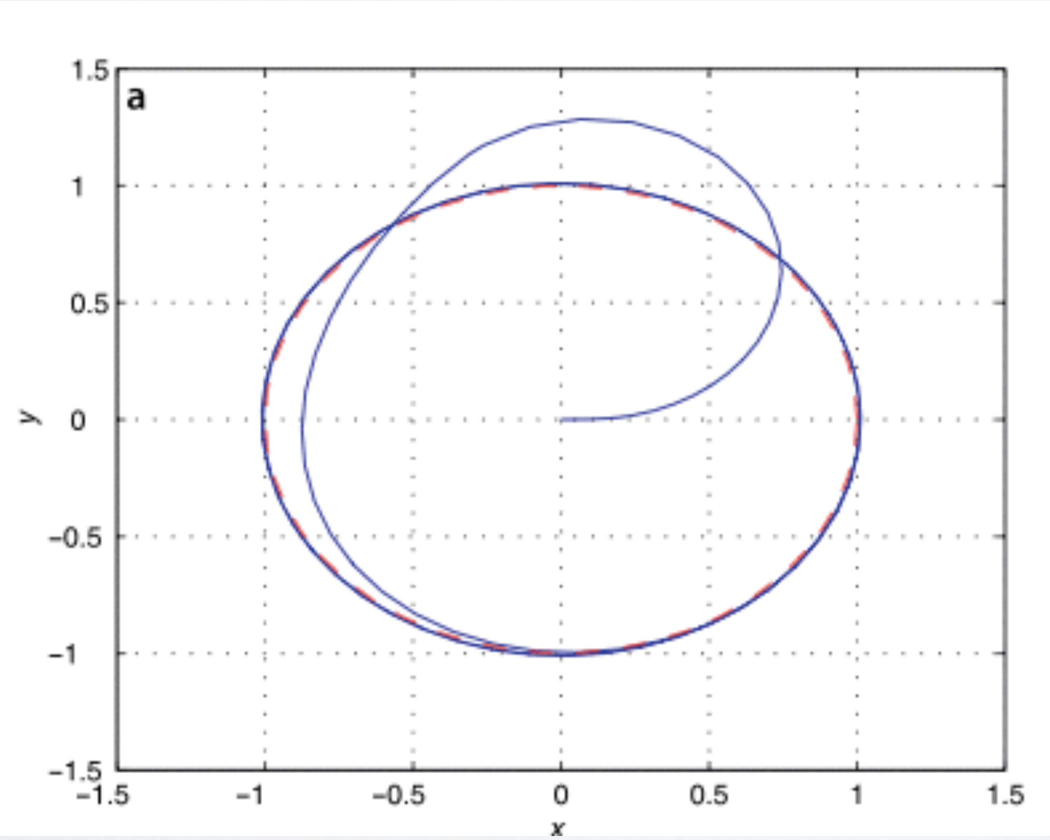
$$\alpha_d = -K_d d,\ K_d > 0$$

$$\theta^* = \tan^{-1}\frac{-a}{b}$$

$$\alpha_h = K_h(\theta^* \ominus \theta),\ K_h > 0$$

$$\gamma = -K_d d + K_h(\theta^* \ominus \theta)$$

FOLLOWING A PATH



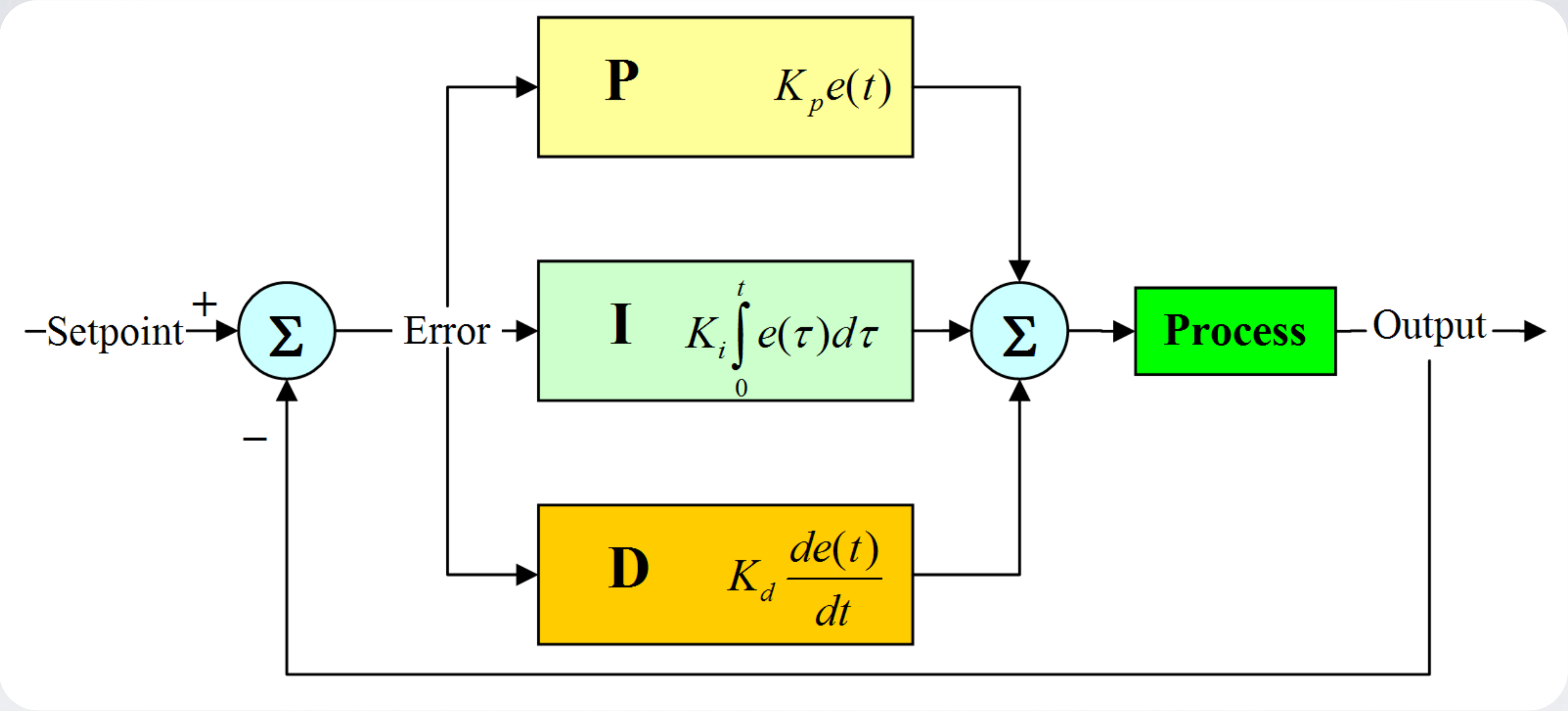
$$e = \sqrt{(x^* - x)^2 + (y^* - y)^2} - d^*$$

$$v^* = K_v e + K_i \int e \, \mathrm{d}t$$

$$\theta^* = \tan^{-1} \frac{y^* - y}{x^* - x}$$

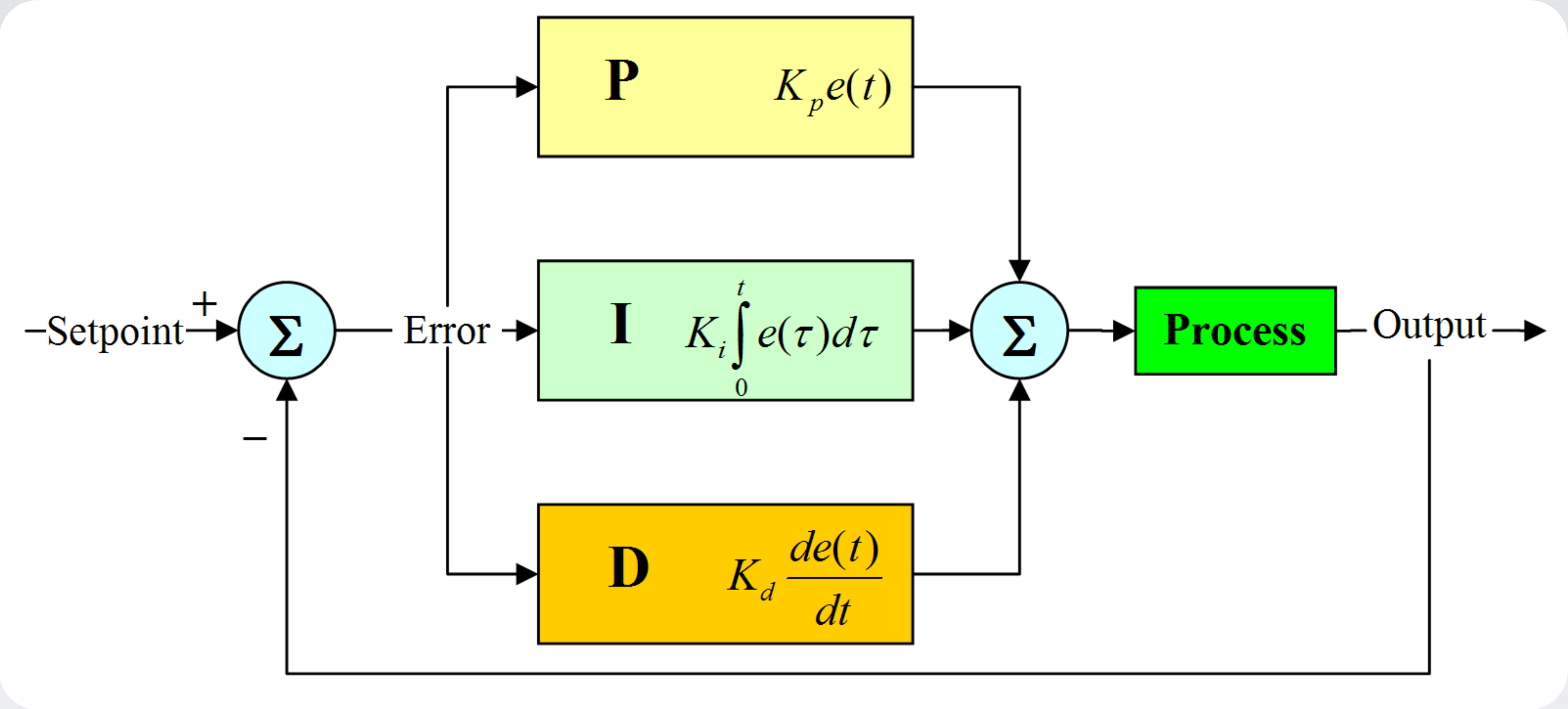
$$\alpha = K_h(\theta^* \ominus \theta), \;\; K_h > 0$$

PID CONTROL



- Simple
- Robust, excellent results in many cases
- Controls 95% of all industrial processes

PID CONTROL



- **P** = Proportional: correct
- **I** = Integral: reduce tracking error
- **D** = Derivative: stabilize (anticipate)

CODE EXAMPLE

```
class PID:
    def __init__(self, Kp, Ki, Kd):
        self.previous_error = 0
        self.integral = 0
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd

    def calc(self, dt, setpoint, y):
        error = setpoint - y
        self.integral += error*dt
        derivative = (error - self.previous_error)/dt
        u = self.Kp*error + self.Ki*self.integral + self.Kd*derivative
        self.previous_error = error
        return u
```

Python example due to Andy Henshaw at GTRI

PID TUNING

- **Ziegler-Nichols**
  - Start with P only
  - Increase $\boldsymbol{K_p}$ until output oscillates
  - Call that "Ultimate gain" $\boldsymbol{K_u}$,and call the oscillation period $\boldsymbol{T_u}$

Ziegler–Nichols method[1]			
Control Type	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$1.2K_p/T_u$	-
PD	$0.8K_u$	-	$K_pT_u/8$
classic PID[2]	$0.60K_u$	$2K_p/T_u$	$K_pT_u/8$

Source: Wikipedia