上海科技大学
ShanghaiTech University

CS283: **Robotics Fall 2017: Kinematics**

Andre Rosendo

ShanghaiTech University

Messages

- Publisher does not know about subscribers
- Subscribers do not know publishers
- One topic name: many subscribers and many publishers possible, BUT: same message type (determined by the first publisher)!
- List all topics in the current system:
  - rostopic list
  - Other commands: `rostopic echo`, `rostopic hz`, `rostopic pub` , `rostopic pub /test std_msgs/String “Hello World!"`

Create own message: Text format

- Types:
  - int8, int16, int32, int64 (plus uint*)
  - float32, float64
  - string
  - time, duration
  - other msg files
  - variable-length array[] and fixed-length array[C]
- Save in folder “msg”, start with big letter, end with “.msg”

```
string first_name
string last_name
uint8 age
uint32 score
```

Services

- ROS **service**: send a “message” or command to service provider, wait for reply
- Text format: First message for **request**
  - Separation: three dashes
  - Then message for **response**
- A call to a service blocks

```
float32 x
float32 y
float32 theta
string name
---
string name
```

```
#include "beginner_tutorials/AddTwoInts.h"

bool add(beginner_tutorials::AddTwoInts::Request  &req,
         beginner_tutorials::AddTwoInts::Response &res)
{
  res.sum = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: [%ld]", (long int)res.sum);
  return true;
}
```

```
ros::ServiceServer service = n.advertiseService("add_two_ints", add);
```

Compiler, Linker

- Standard in Linux: gcc: GNU Compiler Collection
- Compiler: Create machine code out of programming language
  - For C++ code: g++
  - g++ -o helloworld -I/homes/me/randomplace/include helloworld.cc
  - Options:
    - -g - turn on debugging (so GDB gives more friendly output)
    - -o <name> - name of the output file
    - -O to -O4 - turn on optimizations
    - -L<library path> - specify a lib directory
    - -Wall - turns on most warnings
    - -c - output an object file (.o)
    - -I<include path> - specify an include directory
    - -l<library> - link with library lib<library>.a
- Linker: Link the machine code with other machine code (provided by libraries)
  - Static link library: executable includes the statically linked library
  - Dynamic link library: upon execution the program is linked against the library: Multiple programs will use the same code => save memory
  - Program: ln
  - Show dynamic linked libraries used by a program: ldd

Makefile, CMake

- Avoid typing g++ and ln
- Makefile:
  - Commands for compiling and linking the program: “make” uses the file “Makefile”
  - May provide additional commands like “make clean”
  - Can be used to run arbitrary commands, e.g. to create pdf files from LaTeX
- Cmake
  - Cross-platform Makefile generator
  - Searches for dependencies (libraries, headers, etc.)
  - Autoconfigure with “cmake .”
  - “CMakeLists.txt”: specify which files to make, etc.

GIT: distributed revision control and source code management

- Every Git working directory is a full-fledged repository
  - => can work without server, two repos can pull/ push from each other
- Working directory has a hidden .git folder in its root
- Automatically merges common changes in same files
- Non-linear development:
  - Create branches, merge them
- Cryptographic authentication of history
- See Cheat Sheet

Recourses:

- http://wiki.ros.org/ROS/Tutorials/
- https://en.wikipedia.org/wiki/Object-oriented_programming
- C++: http://www.cplusplus.com/doc/tutorial/
  - http://www.cplusplus.com/doc/tutorial/templates/
- https://en.wikipedia.org/wiki/Smart_pointer
  - http://en.cppreference.com/w/cpp/memory/shared_ptr
- http://www.cprogramming.com/tutorial/const_correctness.html

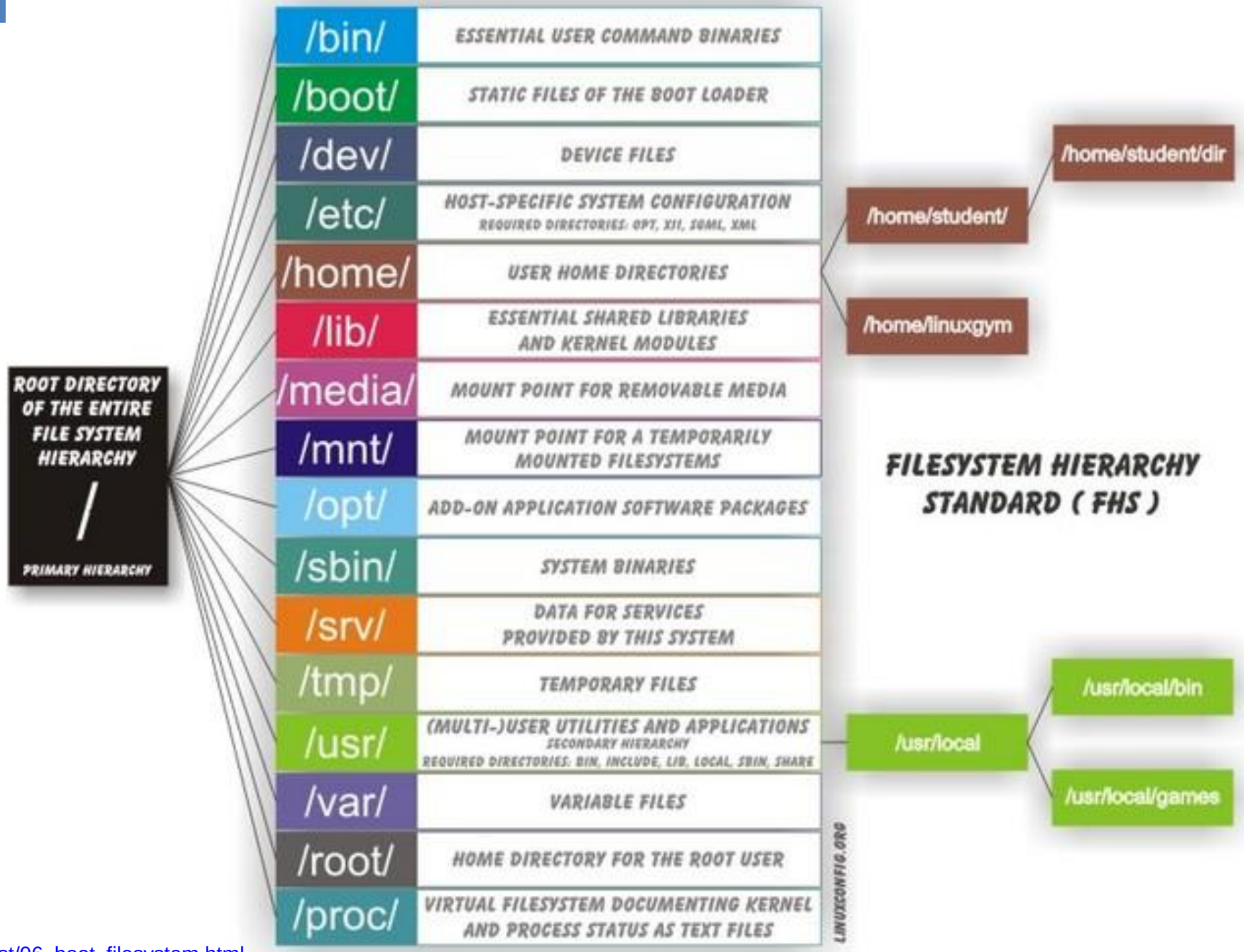
Cheat Sheets

- https://robotics.shanghaitech.edu.cn/static/cheatSheets/bash_cheat_sheet.pdf
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/gitCheatCheet.pdf
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/vim-cheat-sheet.png
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/regular_expressions_cheat_sheet.png
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/cpp_reference_sheet.pdf
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/ROScheatsheet.pdf
- https://robotics.shanghaitech.edu.cn/static/cheatSheets/ROS-Cheat-Sheet-Landscape-v2.pdf

Unix File System

- File types: regular, directory, link, (sockets, named pipes, block devices)
- Slash “/” instead of backslash “\” for folders - distinction between small and big letters!
- One file system tree, beginning with root: “/”
  - Mount partitions (areas of the hard disk): any folder can be the mount point, e.g.: /media/<user_name>/usbDiskName
- Home folders of different users in “/home/<user_name>”
- Hidden files and folders: begin with a dot “.”
- In Unix/ Linux, (almost) everything is a file: devices, partitions, …in “/dev”, e.g. “/dev/video0”
- Show files: “ls”; more info: “ll”; human readable: “-h” – e.g. “ll -h”
- Free space: “df -h”
- Symbolic links (symlink): point to another file or folder. Create with “ln -s from to”

Overview



From: http://devopsbootcamp.readthedocs.org/en/latest/06_boot_filesystem.html

Misc

- Files have access rights: users and groups and others
  - r: read w: write x: execute (for directories: go in)
  - chmod a+w => all (three) are allowed to write
  - chmod o-r => others are not allowed to read
- chown user:group file_name_or_dir change ownership
- Super user: root: can access all files
- sudo <command>: execute a command as root
- sudo su: (one way) to become root
- Compress files: zip + rar for Windows => no support for permissions/ symbolic links
  - tar : tape archive (lol) – sequentially store files and folders (no compression)
  - gzip : compress one file
  - combine: tar gzip: archive.tar.gz

Bash: GNU Unix Shell

- Program that runs in your terminal – executes your commands
- Keyboard up: go through history of last commands
- Tab-complete: any time, press tab to complete the command/ path/ file-name/ … - if a unique solution exists; double tab for list of possible options
- Control C to tell program to stop; Control | to quit;
- Control Z to stop (pause) program: fg to run in foreground again, bg to run in background, kill %1 to kill the last program (in background)
- Start program in background: command &
- Pipe: send output of program 1 as input to program 2: prog1 | prog2; e.g. “ll /dev | less”
- Send standard output to file use “>” e.g.: “ll > file.txt”
- Wildcards: “*” matches anything with any length, “?” matches any one char, e.g. “ll /dev/tty*”

.bashrc

- .bashrc is executed every time a new shell (terminal) is opened
- Execute by hand: “source ~/.bashrc” or “. ~/.bashrc”
- “~” is replaced by your home directory
- Setup variables, e.g.:
  - alias df='df -h' # when calling df, acutally "df -h“ is called – human readable
  - alias ..='cd ..' # executing "..“ will go one level up in the file tree
  - Option: setup ros path always here: “source ~/my_ws/devel/setup.bash”
- Edit input.rc to search history of commands with page up, down:
  - “sudo vi /etc/inputrc” – uncomment “# alternate mappings for "page up" and "page down" to search the history”

vi: editor for the console

- Command mode (press escape) and input mode (press i)
- Install vim for more comfort: sudo apt-get install vim
- Command mode:
  - Press escape to enter command mode
  - “: w” write file
  - “: q” quit
  - “: wq” write file and quit
  - “: q!” quit without writing changes to file
  - Press “d” to delete a char; press “dd” to delete a line
  - Press “/” and enter a regular expression to search
  - Press “n” or “N” for next, previous search result

ssh: secure shell

- Login to remote computer, using encrypted communication
- sudo apt-get install ssh : Installs the ssh server
- Usage: ssh user@host e.g.: ssh schwerti@robotics.shanghaitech.edu.cn
- Option: -X forward X-server: see GUI of remote application on your screen (-Y without encryption)
- ssh-keygen : generate authentication keys – public and private keyfile in .ssh
- ssh-copy-id : copy your public key to remove hose => no login needed anymore!
- Copy files: scp [-r] <from> <to>
  - Either from or to can be remote host: [user@]host:path, e.g. scp hw2.tar.gz test@robotics:homeworks/
  - -r: recursive – copies whole directories

ADMIN

Admin

- Questions:
  - Don’t send emails unless your question contains very private info
  - Use piazza instead!
- HW 2 is published – due Oct 9 22:00

KINEMATICS

Motivation

- Autonomous mobile robots move around in the environment.
  Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- **Odometry!**
- Robot:
  - I know how fast the wheels turned =>
  - I know how the robot moved =>
  - I know where I am ☺

Odometry

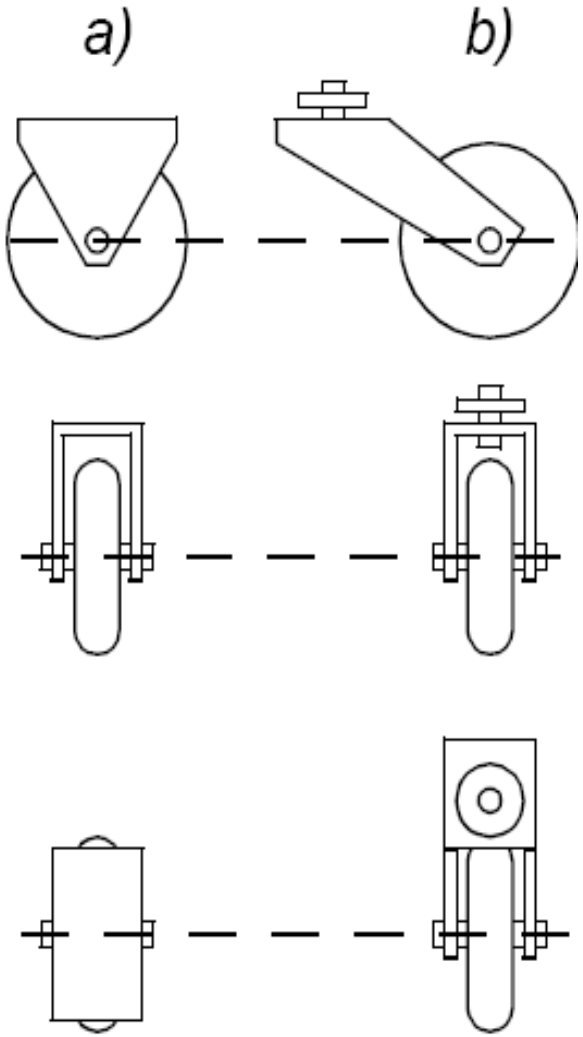
- Robot:
  - I know how fast the wheels turned =>
  - I know how the robot moved =>
  - I know where I am ☺
- Marine Navigation: Dead reckoning (using heading sensor)
- Sources of error (AMR pages 269 - 270):
  - Wheel slip
    - Uneven floor contact (non-planar surface)
    - Robot kinematic: tracked vehicles, 4 wheel differential drive..
  - Integration from speed to position: Limited resolution (time and measurement)
  - Wheel misalignment
  - Wheel diameter uncertainty
  - Variation in contact point of wheel

Mobile Robots with Wheels

- Wheels are the most appropriate solution for most applications
- Three wheels are sufficient to guarantee stability
- With more than three wheels an appropriate suspension is required
- Selection of wheels depends on the application

The Four Basic Wheels Types

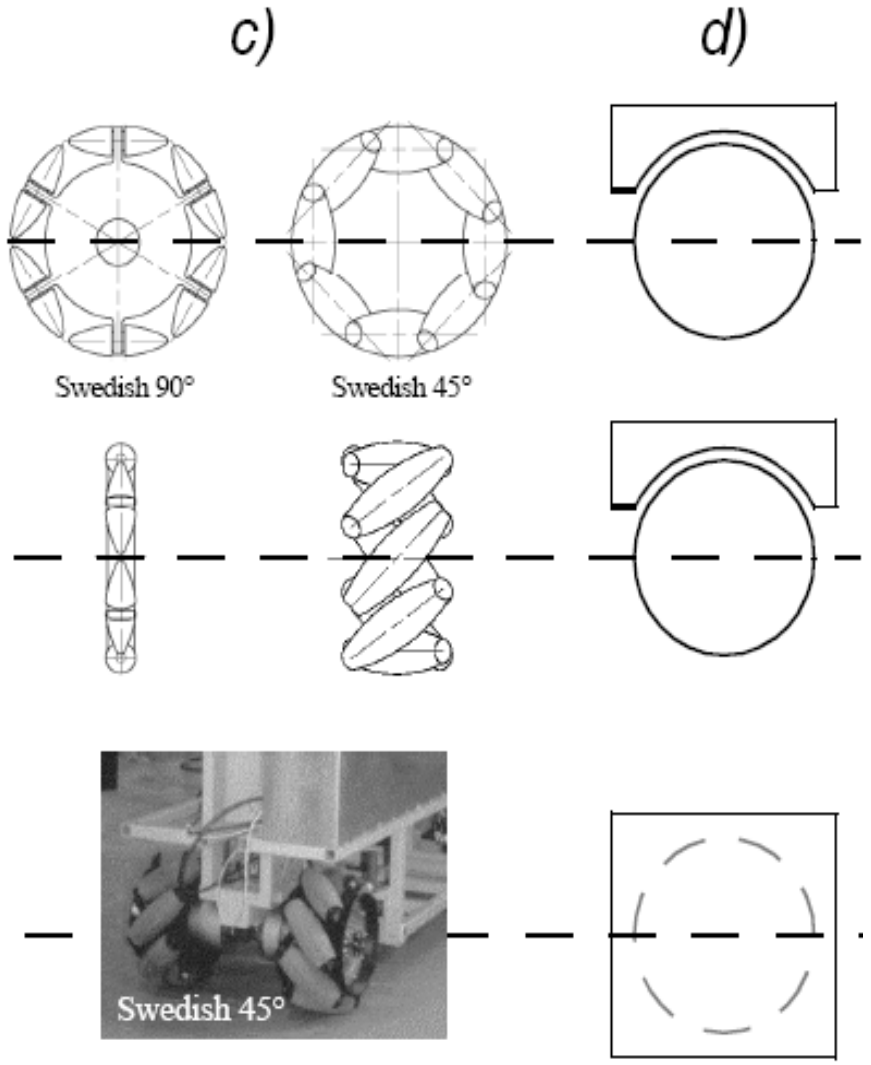
- a) Standard wheel: Two degrees of freedom; rotation around the (motorized) wheel axle and the contact point
- b) Castor wheel: Three degrees of freedom; rotation around the wheel axle, the contact point and the castor axle



The Four Basic Wheels Types

- c) Swedish wheel: Three degrees of freedom; rotation around the (motorized) wheel axle, around the rollers and around the contact point

- d) Ball or spherical wheel: Suspension technically not solved

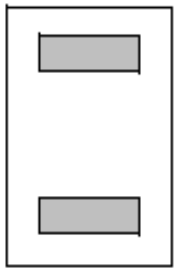


Characteristics of Wheeled Robots and Vehicles

- Stability of a vehicle is be guaranteed with 3 wheels
  - center of gravity is within the triangle with is formed by the ground contact point of the wheels.
- Stability is improved by 4 and more wheel
  - however, this arrangements are hyperstatic and require a flexible suspension system.
- Bigger wheels allow to overcome higher obstacles
  - but they require higher torque or reductions in the gear box.
- Most arrangements are non-holonomic (see chapter 3)
  - require high control effort
- Combining actuation and steering on one wheel makes the design complex and adds additional errors for odometry.

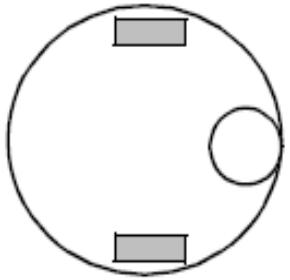
Different Arrangements of Wheels I

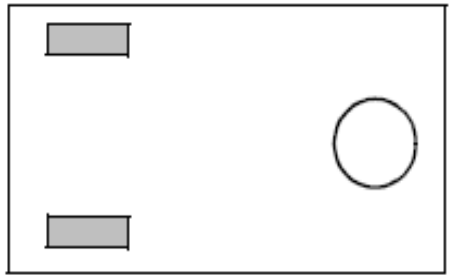
- Two wheels

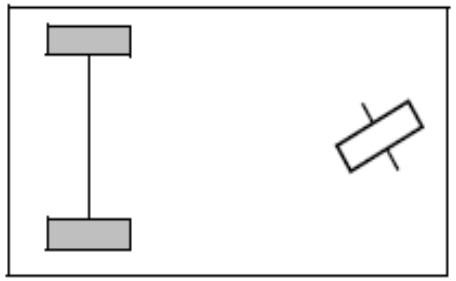


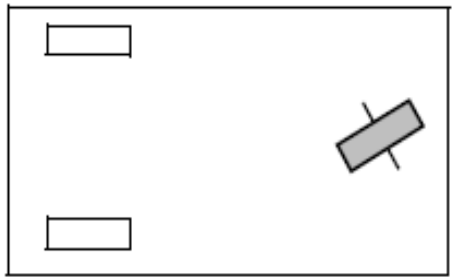
Center of gravity below axle

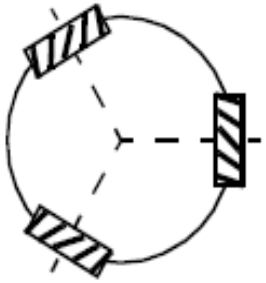
- Three wheels



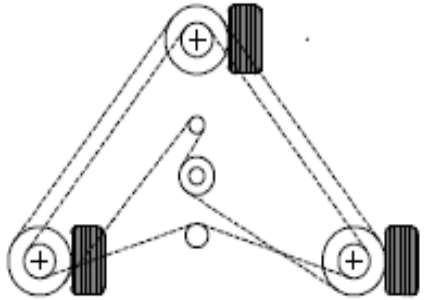








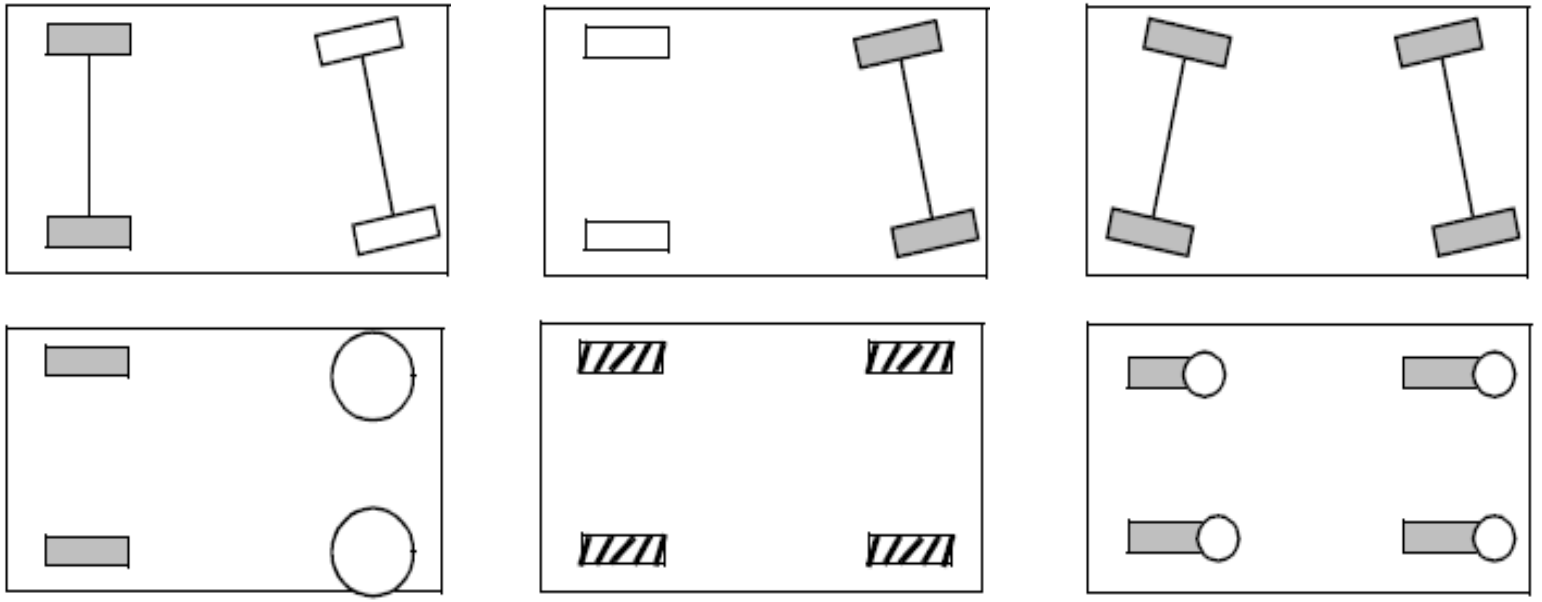
Omnidirectional Drive



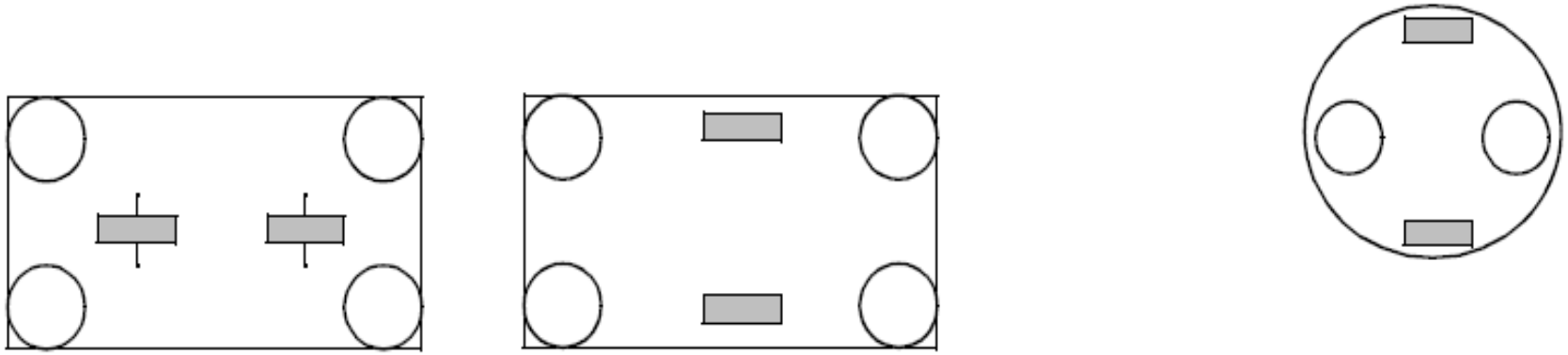
Synchro Drive

Different Arrangements of Wheels II

- Four wheels

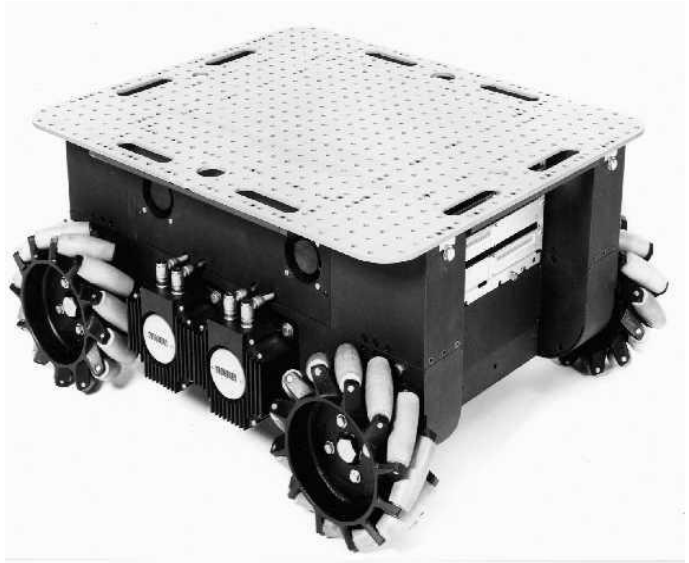


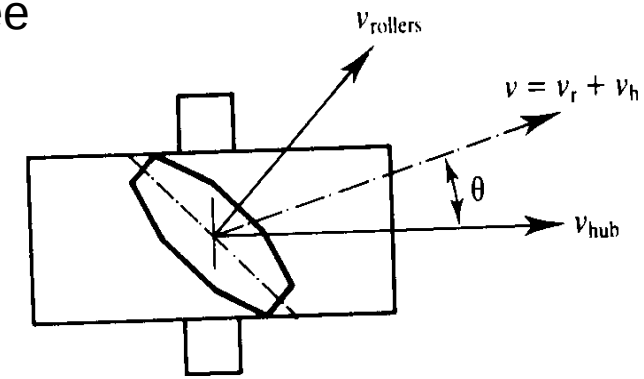
- Six wheels

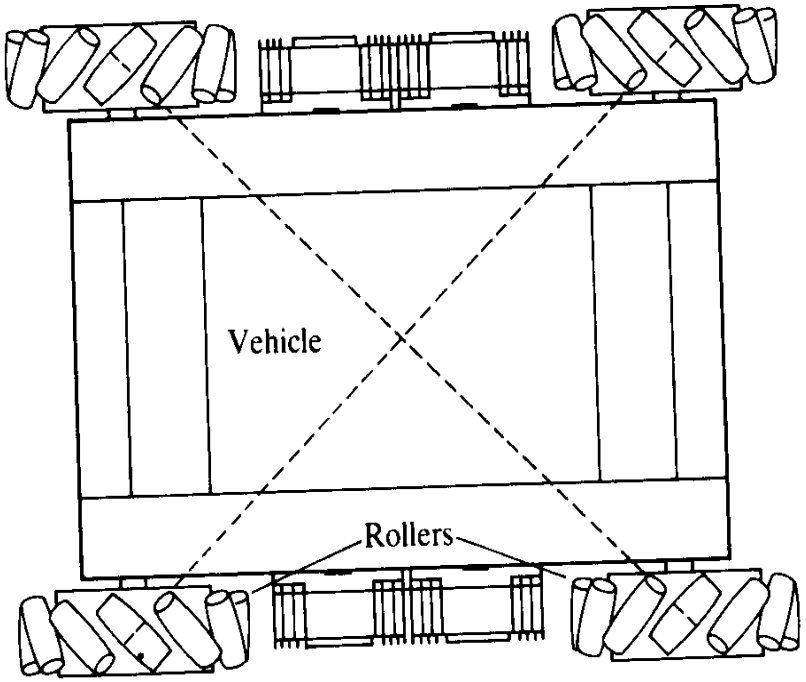


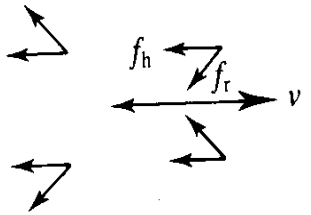
Uranus, CMU: Omnidirectional Drive with 4 Wheels

- Movement in the plane has 3 DOF
  - thus only three wheels can be independently controlled
  - It might be better to arrange three swedish wheels in a triangle

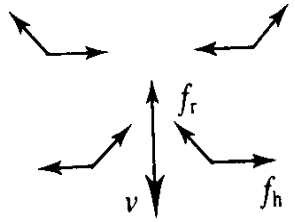




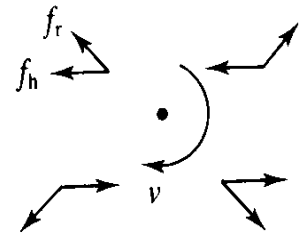




Forward



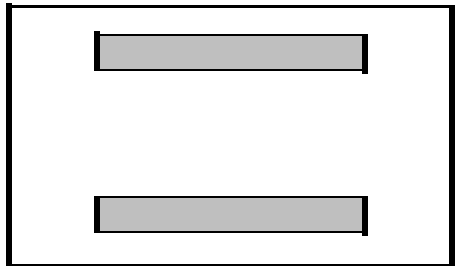
Right



Clockwise

Rugbot, Jacobs Robotics: Tracked Differential Drive

- Kinematic Simplification:
  - 2 Wheels, located at the center



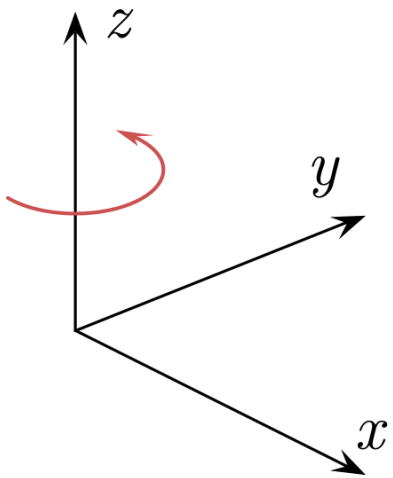
Introduction: Mobile Robot Kinematics

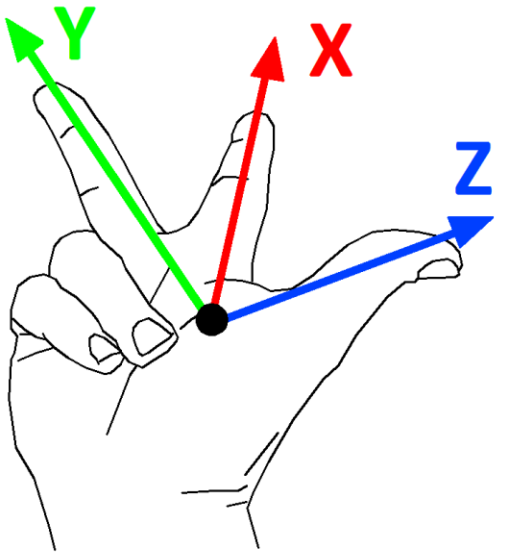
- Aim
  - Description of mechanical behavior of the robot for *design* and *control*
  - Similar to robot manipulator kinematics
  - However, mobile robots can move unbound with respect to its environment
    - there is no direct way to measure the robot's position
    - Position must be integrated over time
    - Leads to inaccuracies of the position (motion) estimate
      -> the number 1 challenge in mobile robotics

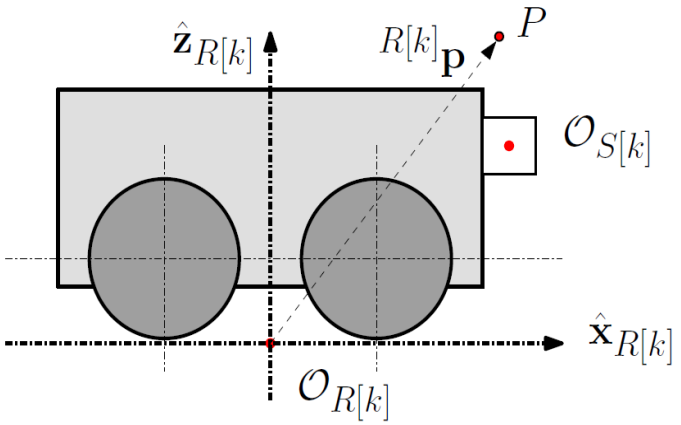
COORDINATE SYSTEM

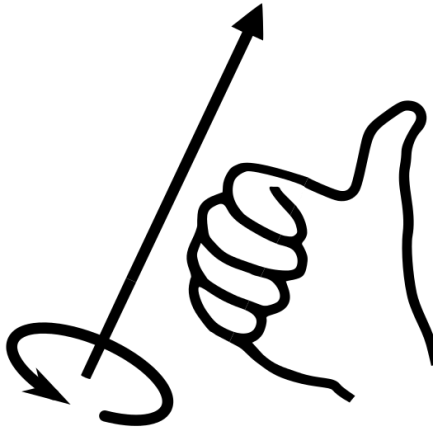
Right Hand Coordinate System

- Standard in Robotics
- Positive rotation around X is anti-clockwise
- Right-hand rule mnemonic:
  - Thumb: z-axis
  - Index finger: x-axis
  - Second finger: y-axis
  - Rotation: Thumb = rotation axis, positive rotation in finger direction
- Robot Coordinate System:
  - X front
  - Z up (Underwater: Z down)
  - Y ???



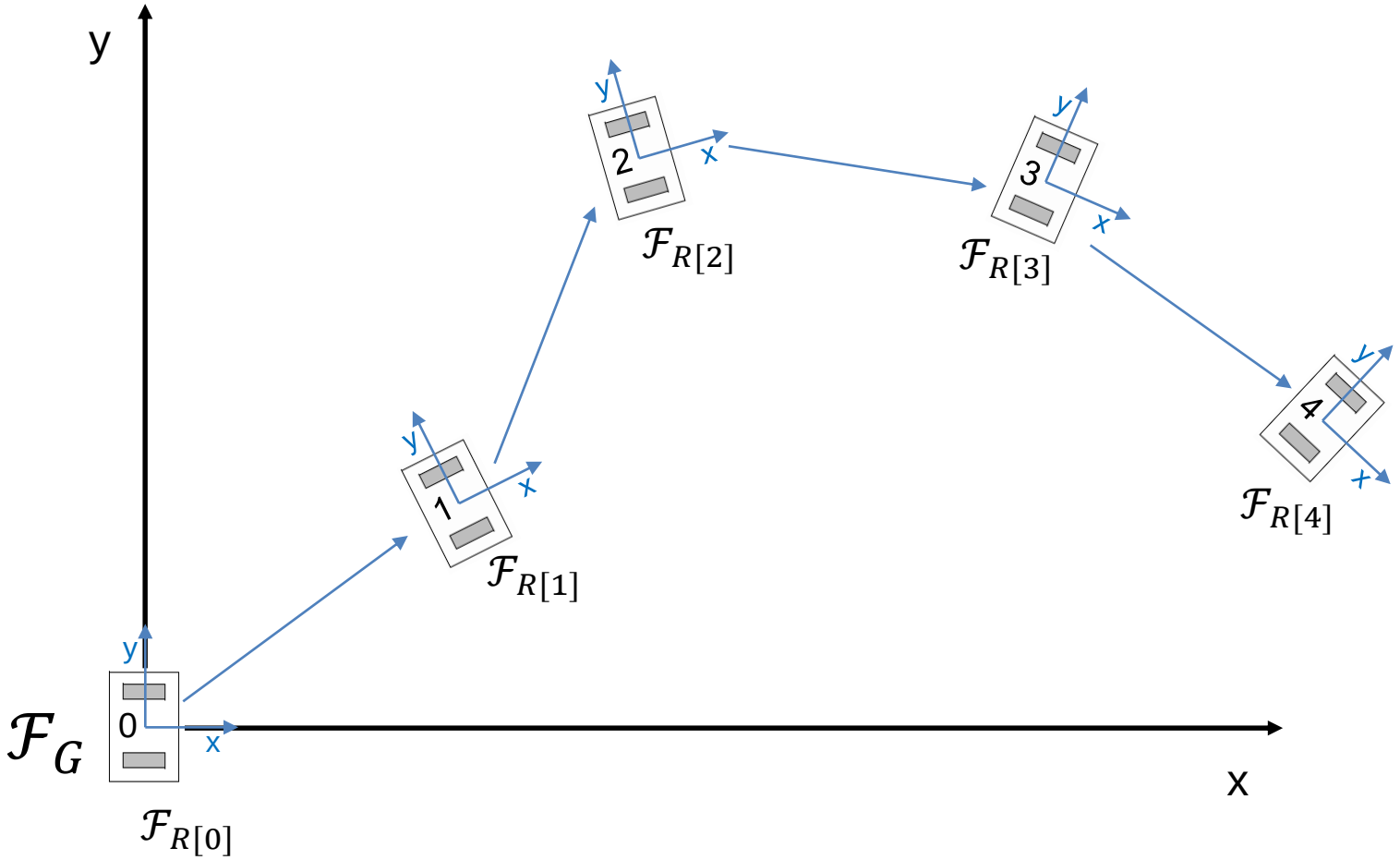






Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

Odometry



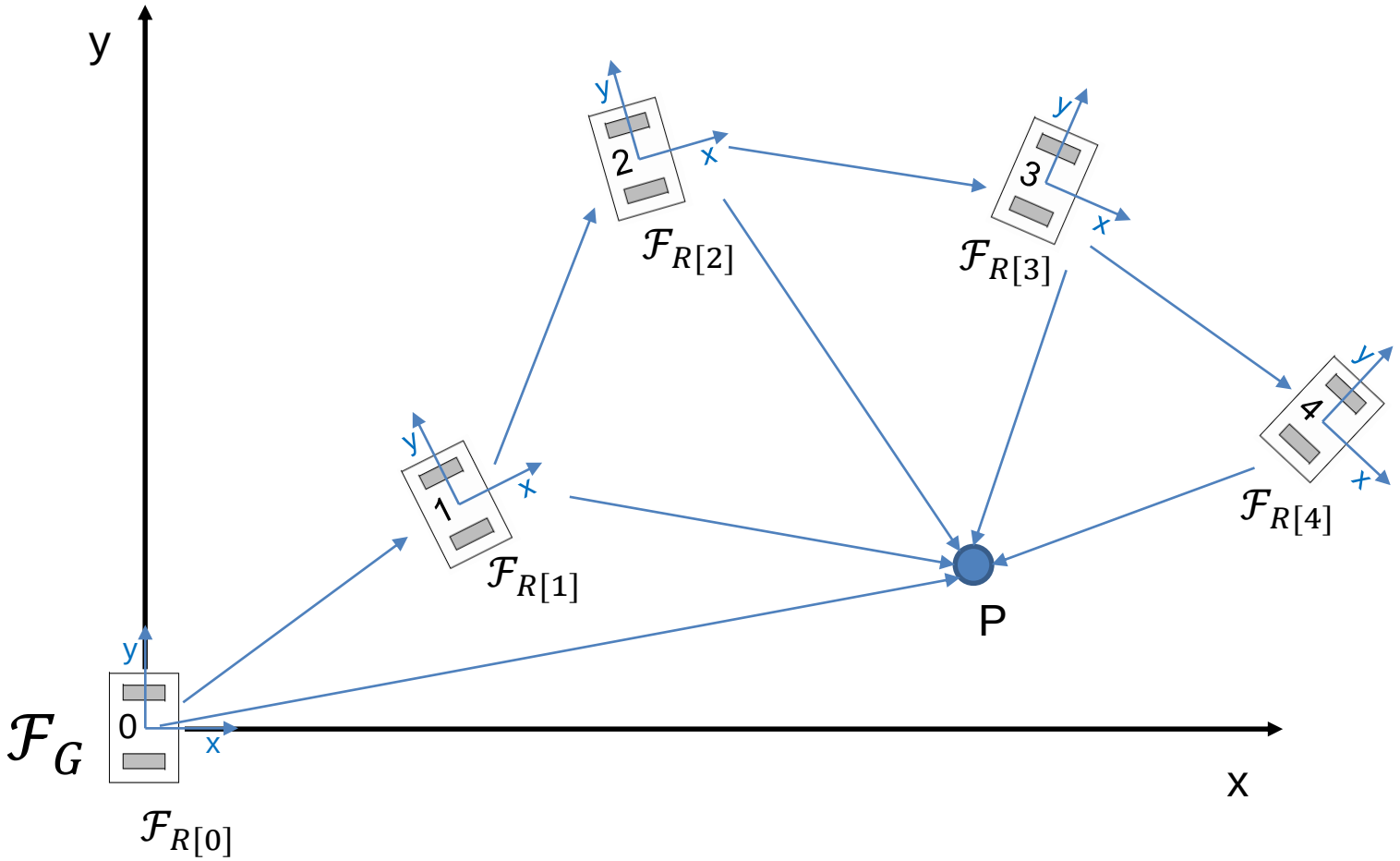
With respect to the robot start pose:
Where is the robot now?

Two approaches – same result:

- Geometry (easy in 2D)
- Transforms (better for 3D)

$\mathcal{F}_{R[X]}$: The ***F***rame of reference (the local coordinate system) of the ***R***obot at the time ***X***

Use of robot frames $\mathcal{F}_{R[X]}$

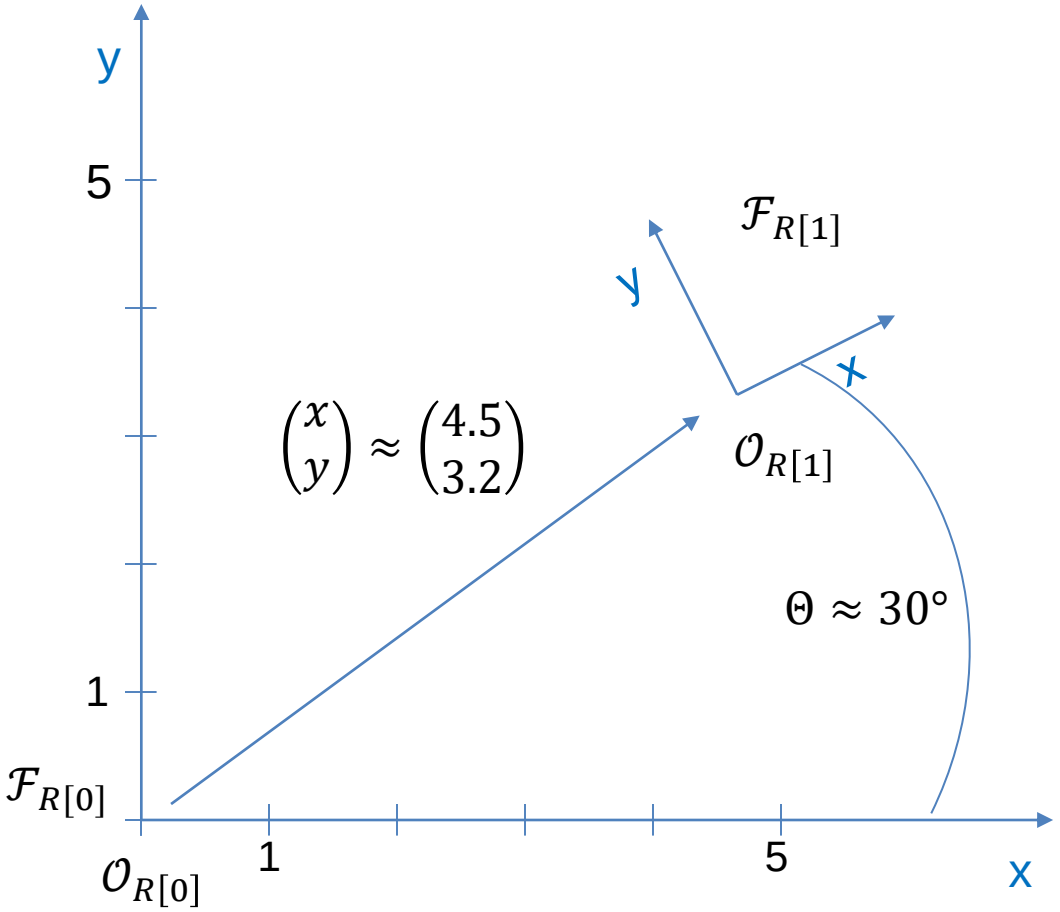


$\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
(coordinates (0, 0)

$\overrightarrow{\mathcal{O}_{R[X]}\mathrm{P}}$: position vector from $\mathcal{O}_{R[X]}$ to point P - $\begin{pmatrix} x \\ y \end{pmatrix}$

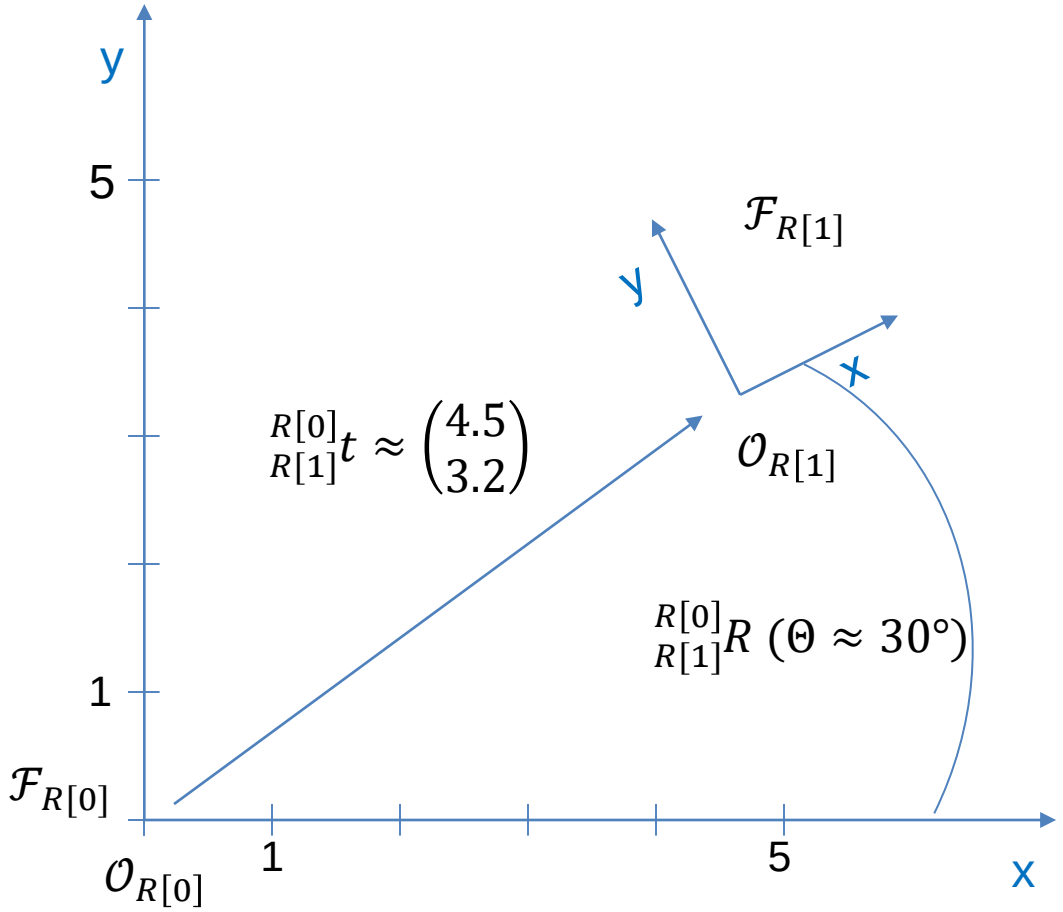
- Object P is observed at times 0 to 4
- Object P is static (does not move)
- The Robot moves
  (e.g. $\mathcal{F}_{R[0]} \neq \mathcal{F}_{R[1]}$)
- => (x, y) coordinates of P are different in all frames, for example:
  - $\overrightarrow{\mathcal{O}_{R[0]}\mathrm{P}} \neq \overrightarrow{\mathcal{O}_{R[1]}\mathrm{P}}$

Position, Orientation & Pose



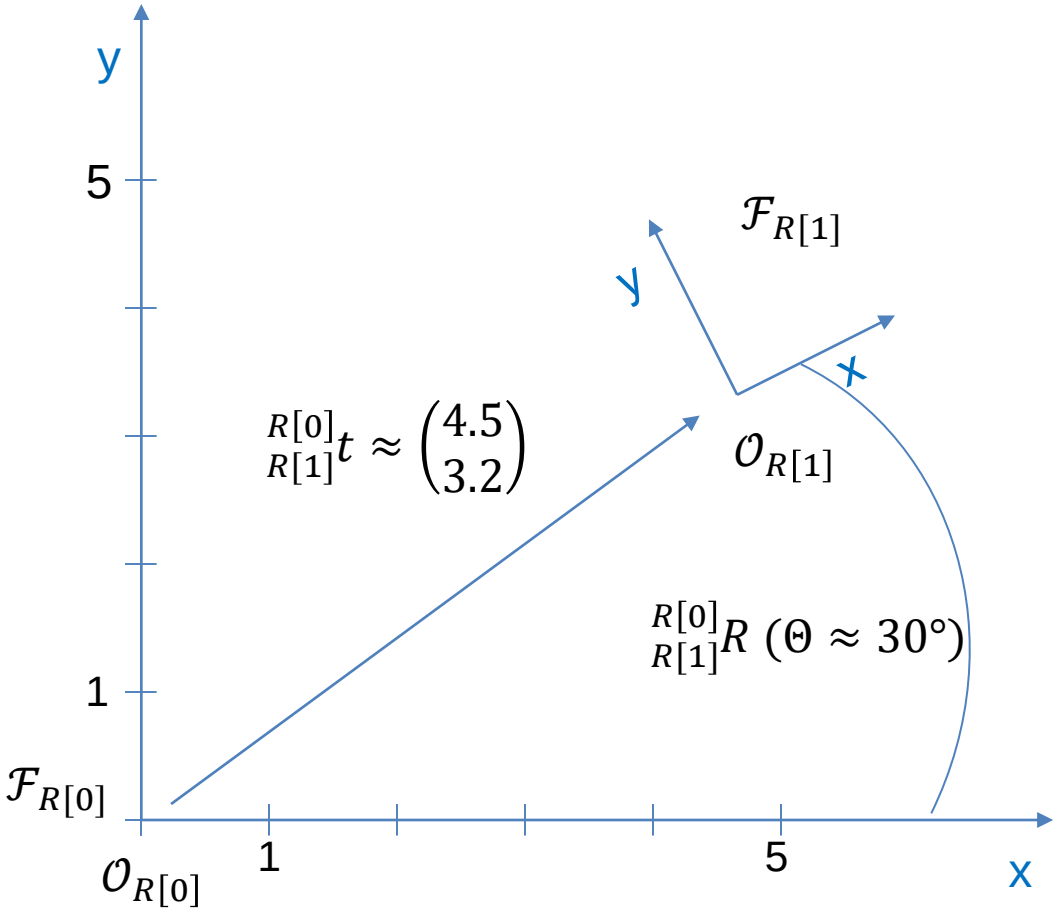
- **Position**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ coordinates of any object or point (or another frame)
  - with respect to (wrt.) a specified frame
- **Orientation**:
  - (Θ) angle of any oriented object (or another frame)
  - with respect to (wrt.) a specified frame
- **Pose**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ position and orientation of any oriented object
  - with respect to (wrt.) a specified frame

Translation, Rotation & Transform



- **Translation**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ difference, change, motion from one reference frame to another reference frame
- **Rotation**:
  - (Θ) difference in angle, rotation between one reference frame and another reference frame
- **Transform**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ difference, motion between one reference frame and another reference frame

Position & Translation, Orientation & Rotation



- $\mathcal{F}_{R[X]}$: Frame of reference of the robot at time X
- Where is that frame $\mathcal{F}_{R[X]}$?
  - Can only be expressed with respect to (wrt.) another frame (e.g. global Frame $\mathcal{F}_G$) =>
  - Pose of $\mathcal{F}_{R[X]}$ wrt. $\mathcal{F}_G$

- $\mathcal{O}_{R[X]}$: Origin of $\mathcal{F}_{R[X]}$
  - $\overrightarrow{\mathcal{O}_{R[X]}\mathcal{O}_{R[X+1]}}$: **Position** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

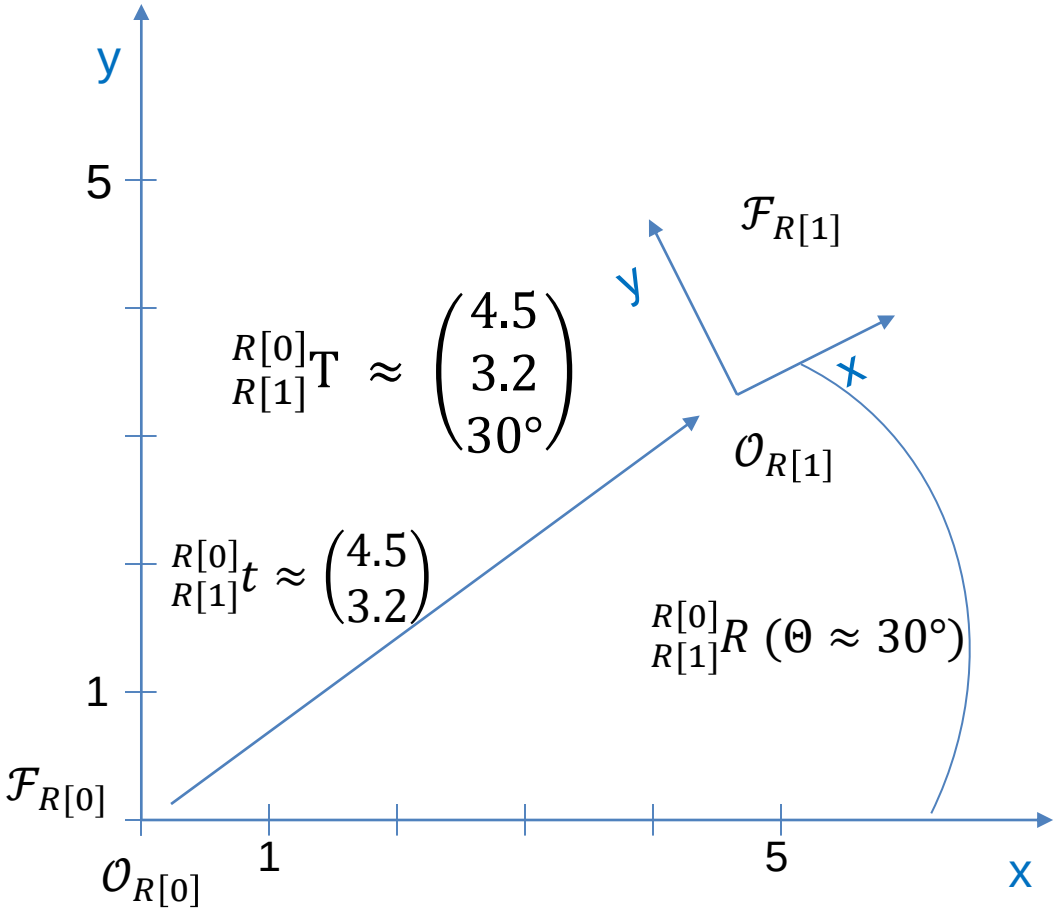
    to $\mathcal{O}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

    $\triangleq {}^{R[X]}_{R[X+1]}t$: **Translation**

- The angle Θ between the x-Axes:
  - **Orientation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

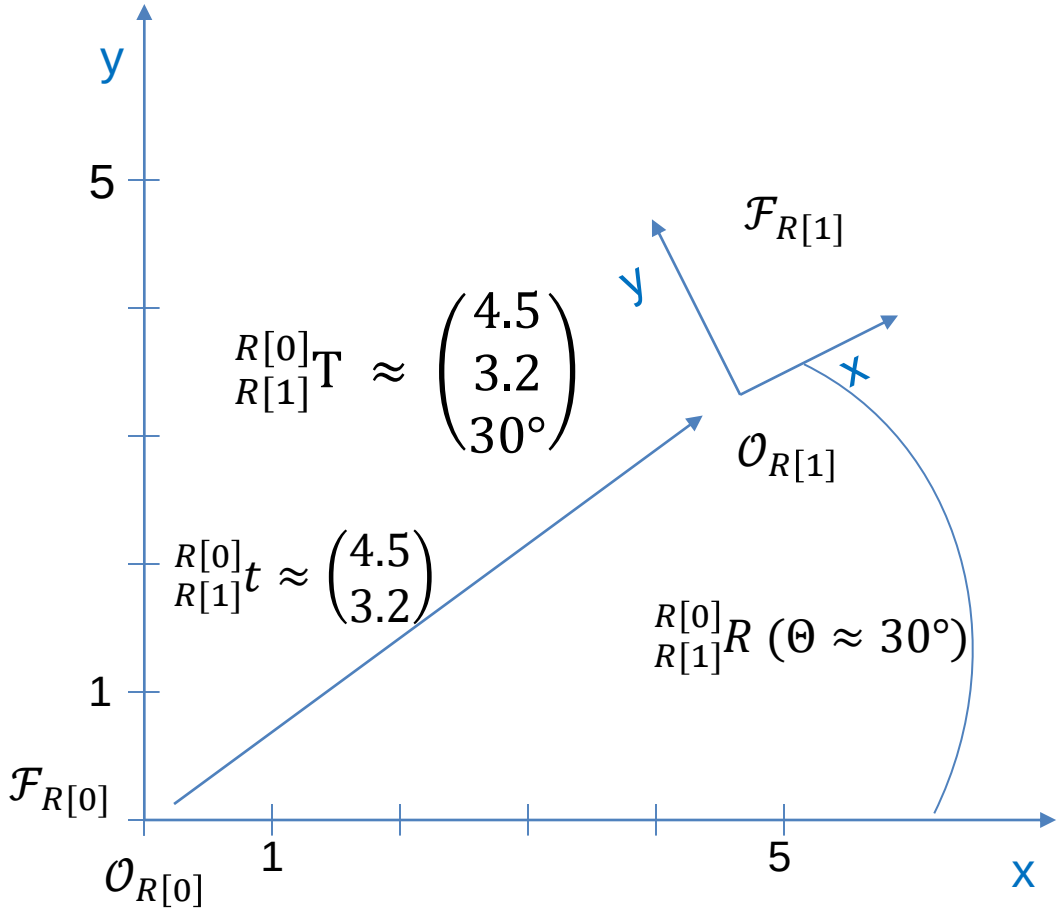
    $\triangleq {}^{R[X]}_{R[X+1]}R$: **Rotation** of $\mathcal{F}_{R[X+1]}$ wrt. $\mathcal{F}_{R[X]}$

Transform



- ${}^{R[X]}_{R[X+1]}t$: **Translation**
  - Position vector (x, y) of $R[X+1]$ wrt. $R[X]$
- ${}^{R[X]}_{R[X+1]}R$ **: Rotation**
  - Angle (Θ) of $R[X+1]$ wrt. $R[X]$
- **Transform:** ${}^{R[X]}_{R[X+1]}\mathrm{T} \equiv \begin{Bmatrix} {}^{R[X]}_{R[X+1]}t \\ {}^{R[X]}_{R[X+1]}R \end{Bmatrix}$

Geometry approach to Odometry



We want to know:

- Position of the robot (x, y)
- Orientation of the robot (Θ)
- => together: Pose $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$

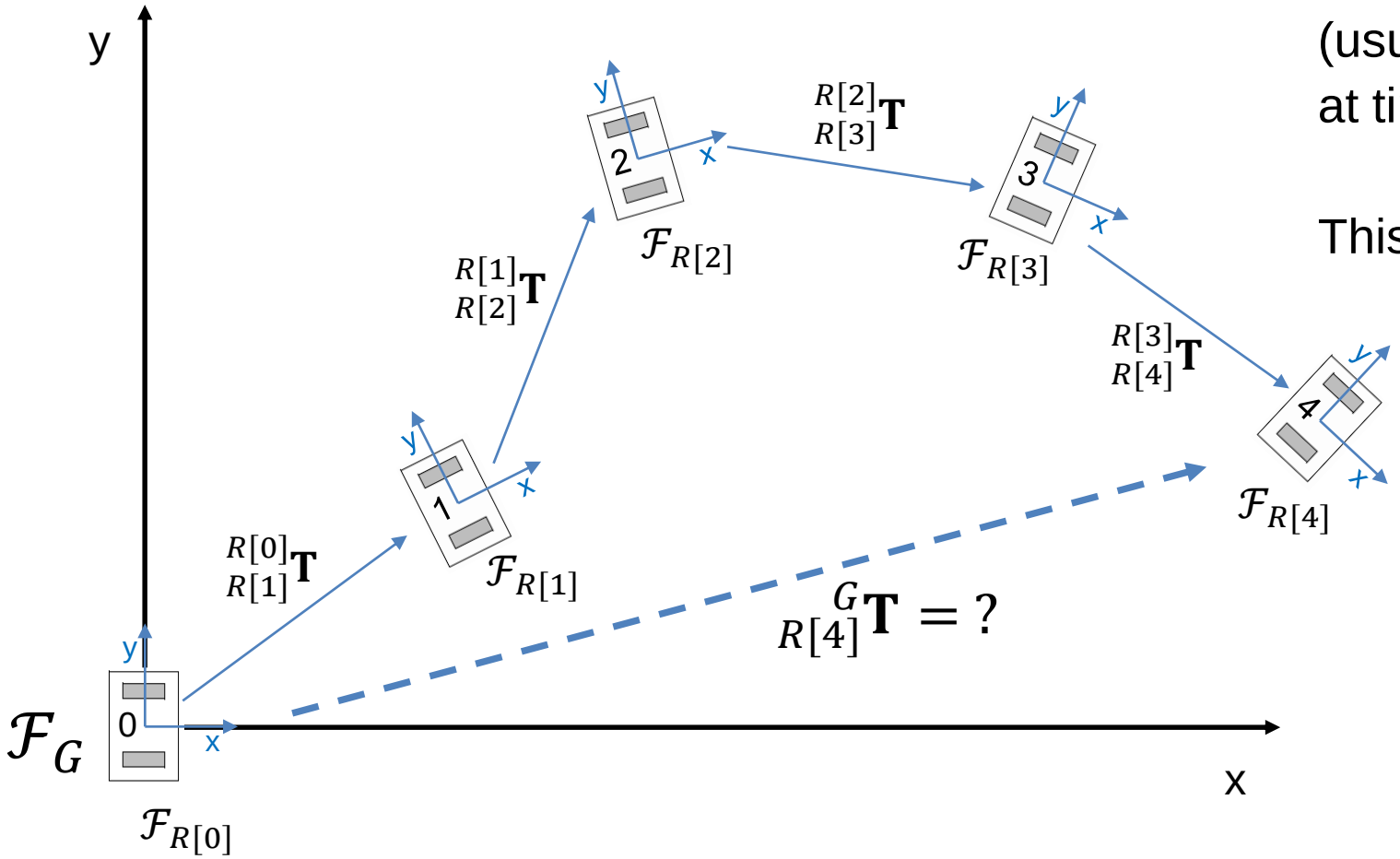
With respect to (wrt.) $\mathcal{F}_G$: The global frame; global coordinate system

$$\mathcal{F}_{R[0]} = \mathcal{F}_G \Rightarrow {}^{G}\mathcal{F}_{R[0]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$${}^{G}\mathcal{F}_{R[1]} = {}^{R[0]}_{R[1]}\mathrm{T} \approx \begin{pmatrix} 4.5 \\ 3.2 \\ 30° \end{pmatrix}$$

Blackboard: $^{R[1]}_{R[2]}\mathrm{T} \approx \begin{pmatrix} 2 \\ 3 \\ 60° \end{pmatrix}$

Mathematical approach: Transforms



Where is the Robot now?

The pose of $\mathcal{F}_{R[X]}$ with respect to $\mathcal{F}_G$ (usually = $\mathcal{F}_{R[0]}$) is the pose of the robot at time X.

This is equivalent to $^{G}_{R[X]}\mathbf{T}$

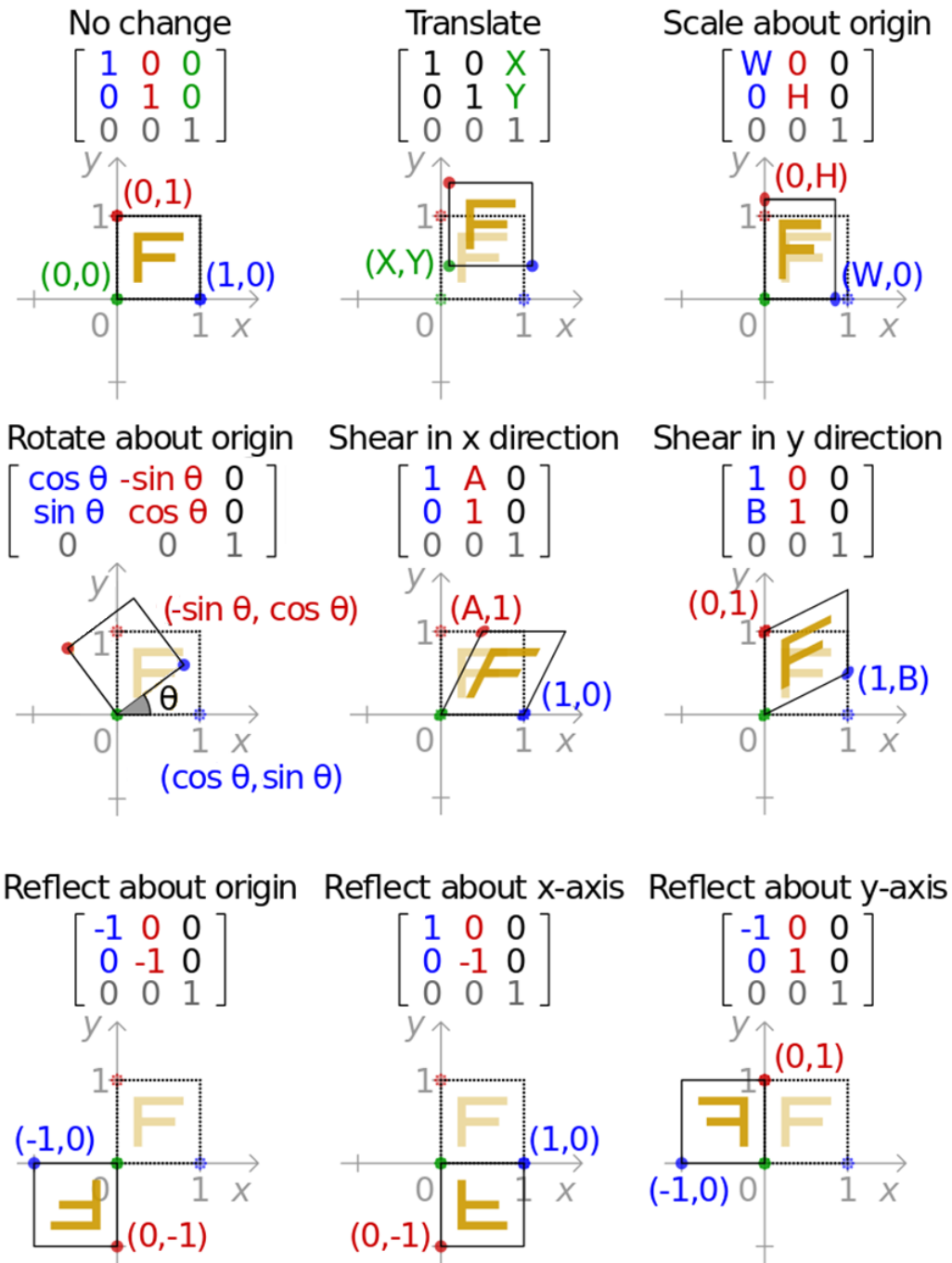
Chaining of Transforms

$$^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T}\ {}^{R[X]}_{R[X+1]}\mathbf{T}$$

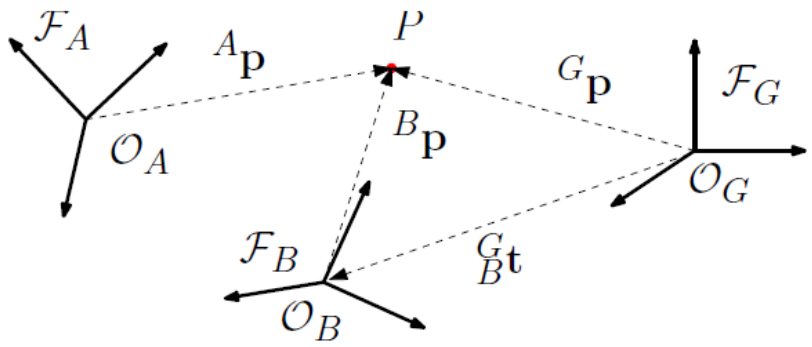
often: $\mathcal{F}_G \equiv \mathcal{F}_{R[0]} \Rightarrow {}^{G}_{R[0]}\mathbf{T} = id$

Affine Transformation

- Function between affine spaces. Preserves:
  - points,
  - straight lines
  - planes
  - sets of parallel lines remain parallel
- Allows:
  - Interesting for Robotics: translation, rotation, (scaling), and chaining of those
  - Not so interesting for Robotics: reflection, shearing, homothetic transforms
- Rotation and Translation: $\begin{bmatrix} \cos\theta & -\sin\theta & X \\ \sin\theta & \cos\theta & Y \\ 0 & 0 & 1 \end{bmatrix}$



Transform



Notation	Meaning
$\mathcal{F}_{\mathrm{R}[k]}$	Coordinate frame attached to object 'R' (usually the robot) at sample time-instant k.
$\mathcal{O}_{\mathrm{R}[k]}$	Origin of $\mathcal{F}_{\mathrm{R}[k]}$.
$^{\mathrm{R}[k]}\mathbf{p}$	For any general point P, the position vector $\overrightarrow{\mathcal{O}_{\mathrm{R}[k]}P}$ resolved in $\mathcal{F}_{\mathrm{R}[k]}$.
$^{\mathrm{H}}\hat{\mathbf{x}}_{\mathrm{R}}$	The x-axis direction of $\mathcal{F}_{\mathrm{R}}$ resolved in $\mathcal{F}_{\mathrm{H}}$. Similarly, $^{\mathrm{H}}\hat{\mathbf{y}}_{\mathrm{R}}$, $^{\mathrm{H}}\hat{\mathbf{z}}_{\mathrm{R}}$ can be defined. Obviously, $^{\mathrm{R}}\hat{\mathbf{x}}_{\mathrm{R}} = \hat{\mathbf{e}}_1$. Time indices can be added to the frames, if necessary.
$^{\mathrm{R}[k]}_{\mathrm{S}[k']}\mathbf{R}$	The rotation-matrix of $\mathcal{F}_{\mathrm{S}[k']}$ with respect to $\mathcal{F}_{\mathrm{R}[k]}$.
$^{\mathrm{R}}_{\mathrm{S}}\mathbf{t}$	The translation vector $\overrightarrow{\mathcal{O}_{\mathrm{R}}\mathcal{O}_{\mathrm{S}}}$ resolved in $\mathcal{F}_{\mathrm{R}}$.

Transform between two coordinate frames

$$^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_{\mathrm{G}}\mathcal{O}_{\mathrm{A}}} \text{ resolved in } \mathcal{F}_{\mathrm{G}}$$

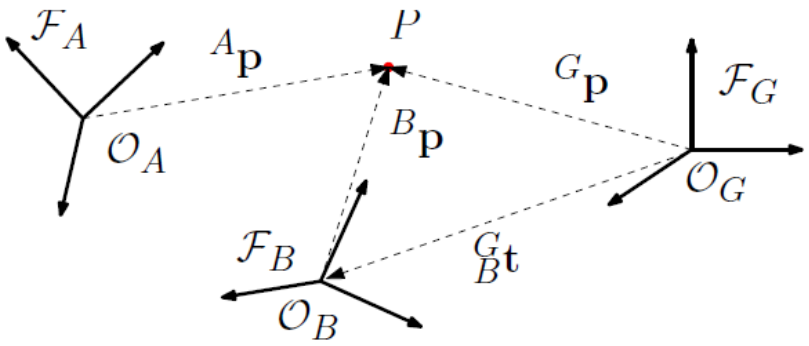
$$\begin{pmatrix} ^{\mathrm{G}}\mathbf{p} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} ^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} & ^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ \mathbf{0}_{1\times[2,3]} & 1 \end{pmatrix} \begin{pmatrix} ^{\mathrm{A}}\mathbf{p} \\ 1 \end{pmatrix} \qquad ^{\mathrm{G}}_{\mathrm{A}}\mathbf{T} \equiv \left\{ \begin{matrix} ^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \\ ^{\mathrm{G}}_{\mathrm{A}}\mathbf{R} \end{matrix} \right\}$$

$$^{\mathrm{G}}\mathbf{p} = ^{\mathrm{G}}_{\mathrm{A}}\mathbf{R}\ ^{\mathrm{A}}\mathbf{p} + ^{\mathrm{G}}_{\mathrm{A}}\mathbf{t} \triangleq ^{\mathrm{G}}_{\mathrm{A}}\mathbf{T}(\ ^{\mathrm{A}}\mathbf{p}).$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & ^{\mathrm{G}}_{\mathrm{A}}\mathrm{t}_{\mathrm{x}} \\ \sin\theta & \cos\theta & ^{\mathrm{G}}_{\mathrm{A}}\mathrm{t}_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix}$$

Example

Transform: Operations



Transform between two coordinate frames (chaining, compounding):

$$ {}^G_B\mathbf{T} = {}^G_A\mathbf{T}\,{}^A_B\mathbf{T} \equiv \begin{Bmatrix} {}^G_A\mathbf{R}\,{}^A_B\mathbf{t} + {}^G_A\mathbf{t} \\ {}^G_A\mathbf{R}\,{}^A_B\mathbf{R} \end{Bmatrix} $$

Inverse of a Transform :

$$ {}^B_A\mathbf{T} = {}^A_B\mathbf{T}^{-1} \equiv \begin{Bmatrix} -{}^A_B\mathbf{R}^\mathsf{T}\,{}^A_B\mathbf{t} \\ {}^A_B\mathbf{R}^\mathsf{T} \end{Bmatrix} $$

Relative (Difference) Transform :

$$ {}^B_A\mathbf{T} = {}^G_B\mathbf{T}^{-1}\,{}^G_A\mathbf{T} $$

See: **Quick Reference to Geometric Transforms in Robotics** by Kaustubh Pathak on the webpage!

Chaining :

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T}\ {}^{R[X]}_{R[X+1]}\mathbf{T} \equiv \begin{Bmatrix} {}^{G}_{R[X]}\mathrm{R} & {}^{R[X]}_{R[X+1]}t + {}^{G}_{R[X]}t \\ {}^{G}_{R[X]}\mathrm{R} & {}^{R[X]}_{R[X+1]}\mathrm{R} \end{Bmatrix} = \begin{Bmatrix} {}^{G}_{R[X+1]}t \\ {}^{G}_{R[X+1]}\mathrm{R} \end{Bmatrix} $$

In 2D Translation:

$$ \begin{bmatrix} {}^{G}_{R[X+1]}t_x \\ {}^{G}_{R[X+1]}t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos {}^{G}_{R[X]}\theta & -\sin {}^{G}_{R[X]}\theta & {}^{G}_{R[X]}\mathrm{t_x} \\ \sin {}^{G}_{R[X]}\theta & \cos {}^{G}_{R[X]}\theta & {}^{G}_{R[X]}\mathrm{t_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{R[X]}_{R[X+1]}\mathrm{t_x} \\ {}^{R[X]}_{R[X+1]}\mathrm{t}_y \\ 1 \end{bmatrix} $$

In 2D Rotation:

$$ {}^{G}_{R[X+1]}\mathrm{R} = \begin{bmatrix} \cos {}^{G}_{R[X+1]}\theta & -\sin {}^{G}_{R[X+1]}\theta \\ \sin {}^{G}_{R[X+1]}\theta & \cos {}^{G}_{R[X+1]}\theta \end{bmatrix} = \begin{bmatrix} \cos {}^{G}_{R[X]}\theta & -\sin {}^{G}_{R[X]}\theta \\ \sin {}^{G}_{R[X]}\theta & \cos {}^{G}_{R[X]}\theta \end{bmatrix} \begin{bmatrix} \cos {}^{R[X]}_{R[X+1]}\theta & -\sin {}^{R[X]}_{R[X+1]}\theta \\ \sin {}^{R[X]}_{R[X+1]}\theta & \cos {}^{R[X]}_{R[X+1]}\theta \end{bmatrix} $$

In 2D Rotation (simple): $ {}^{G}_{R[X+1]}\theta = {}^{G}_{R[X]}\theta + {}^{R[X]}_{R[X+1]}\theta $

In ROS

- First Message at time 97 : G
- Message at time 103 : X
- Next Message at time 107 : X+1

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T} \; {}^{R[X]}_{R[X+1]}\mathbf{T} $$

$$ {}^{R[X]}_{R[X+1]}t_x $$

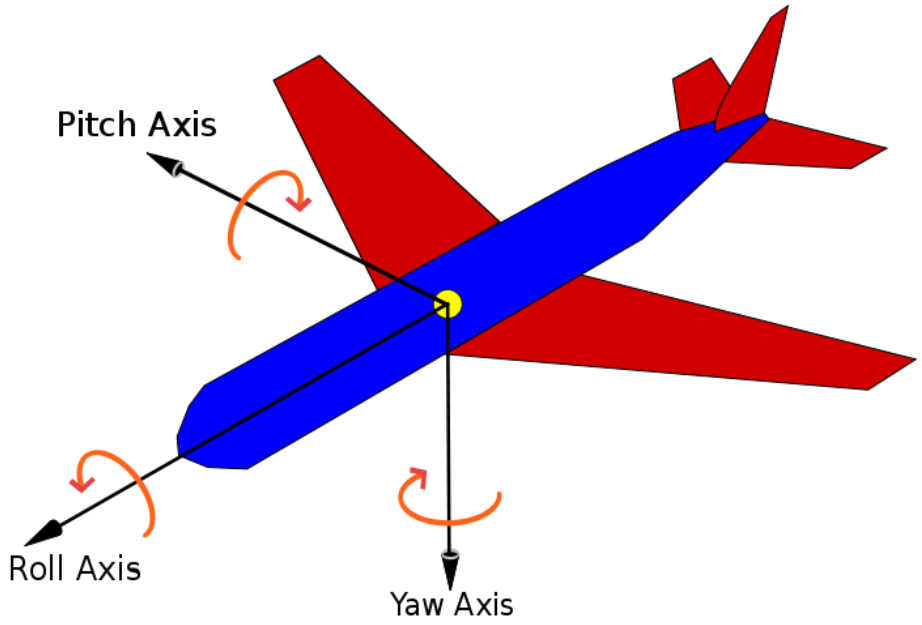
$$ {}^{R[X]}_{R[X+1]}t_y $$

$$ {}^{R[X]}_{R[X+1]}\Theta $$

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D pose2D
  float64 x
  float64 y
  float64 theta
```

3D Rotation

- Euler angles: Roll, Pitch, Yaw
  - ☹ Singularities
- Quaternions:
  - Concatenating rotations is computationally faster and numerically more stable
  - Extracting the angle and axis of rotation is simpler
  - Interpolation is more straightforward
  - Unit Quaternion: norm = 1
  - Scalar (real) part: q_0 , sometimes q_w
  - Vector (imaginary) part: q
  - Over determined: 4 variables for 3 DoF



$$\check{\mathbf{p}} \equiv p_0 + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

$$\check{\mathbf{q}} = \begin{pmatrix} q_0 & q_x & q_y & q_z \end{pmatrix}^\mathsf{T} \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}$$

Transform in 3D

Matrix Euler Quaternion

$$ {}^{G}_{A}\mathbf{T} = \begin{bmatrix} {}^{G}_{A}\mathrm{R} & {}^{G}_{A}\mathrm{t} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\Theta \end{pmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\check{\mathrm{q}} \end{pmatrix} $$

$$ {}^{G}_{A}\Theta \triangleq (\theta_r, \theta_p, \theta_y)^T $$

In ROS: Quaternions! (w, x, y, z)
Uses Bullet library for Transforms

Rotation Matrix 3x3

$$ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} $$

$$ R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} $$

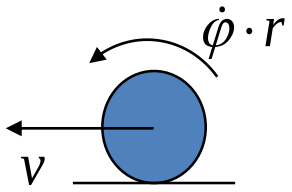
$$ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

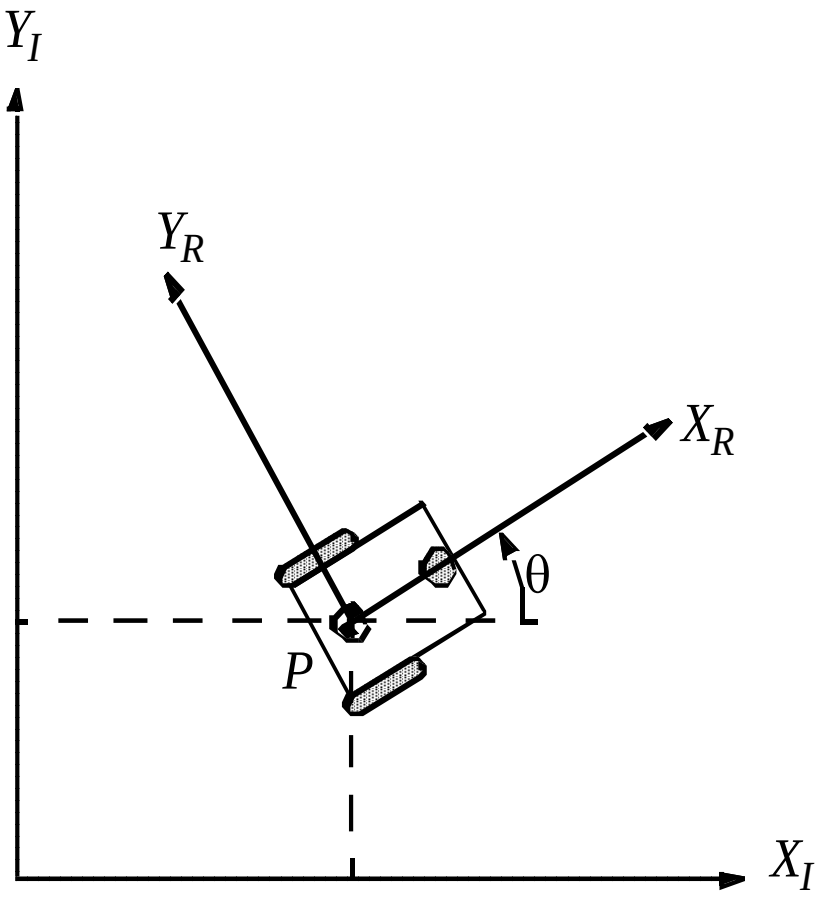
$$ R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) $$

yaw = α, pitch = β, roll = γ

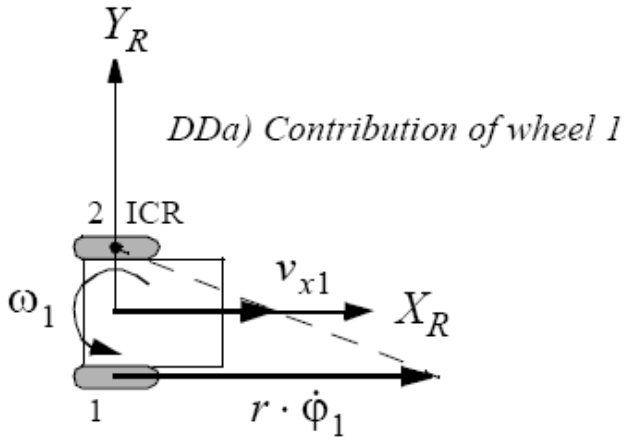
Wheel Kinematic Constraints: Assumptions

- Movement on a horizontal plane
- Point contact of the wheels
- Wheels not deformable
- Pure rolling
  - $v_c = 0$ at contact point
- No slipping, skidding or sliding
- No friction for rotation around contact point
- Steering axes orthogonal to the surface
- Wheels connected by rigid frame (chassis)





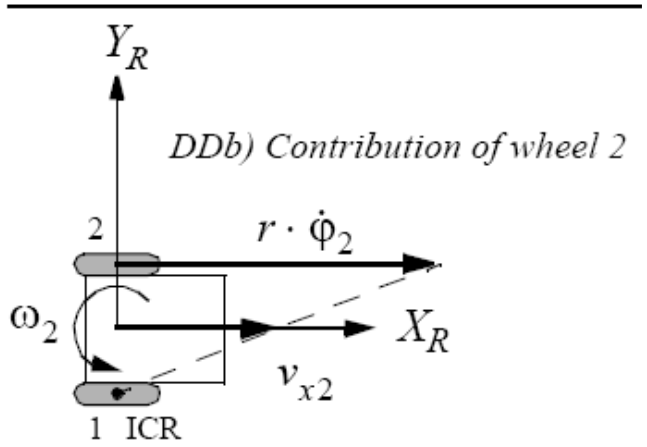
Forward Kinematic Model: Geometric Approach



Differential-Drive:

DDa) $v_{x1} = \frac{1}{2} r\dot{\varphi}_1 \;;\; v_{y1} = 0 \;;\; \omega_1 = \frac{1}{2l} r\dot{\varphi}_1$

DDb) $v_{x2} = \frac{1}{2} r\dot{\varphi}_2 \;;\; v_{y2} = 0 \;;\; \omega_2 = -\frac{1}{2l} r\dot{\varphi}_2$



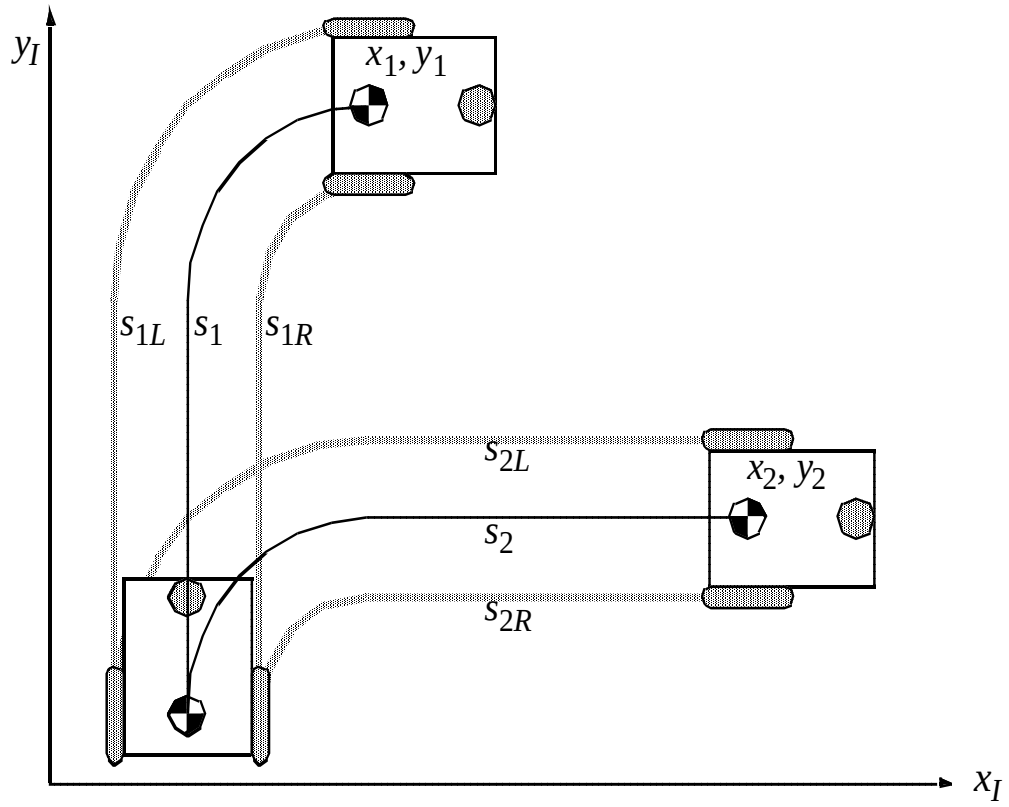
$$\rightarrow \dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_I = R(\theta)^{-1} \begin{bmatrix} v_{x1} + v_{x2} \\ v_{y1} + v_{y2} \\ \omega_1 + \omega_2 \end{bmatrix} = R(\theta)^{-1} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix}$$

Inverse of R => Active and Passive Transform:
http://en.wikipedia.org/wiki/Active_and_passive_transformation

Mobile Robot Kinematics: Non-Holonomic Systems

$s_1 = s_2$; $s_{1R} = s_{2R}$; $s_{1L} = s_{2L}$

but: $x_1 \neq x_2$; $y_1 \neq y_2$



- Non-holonomic systems
  - differential equations are not integrable to the final position.
  - the measure of the traveled distance of each wheel is not sufficient to calculate the final position of the robot. One has also to know how this movement was executed as a function of time.