Navigation

of a cleaning robot

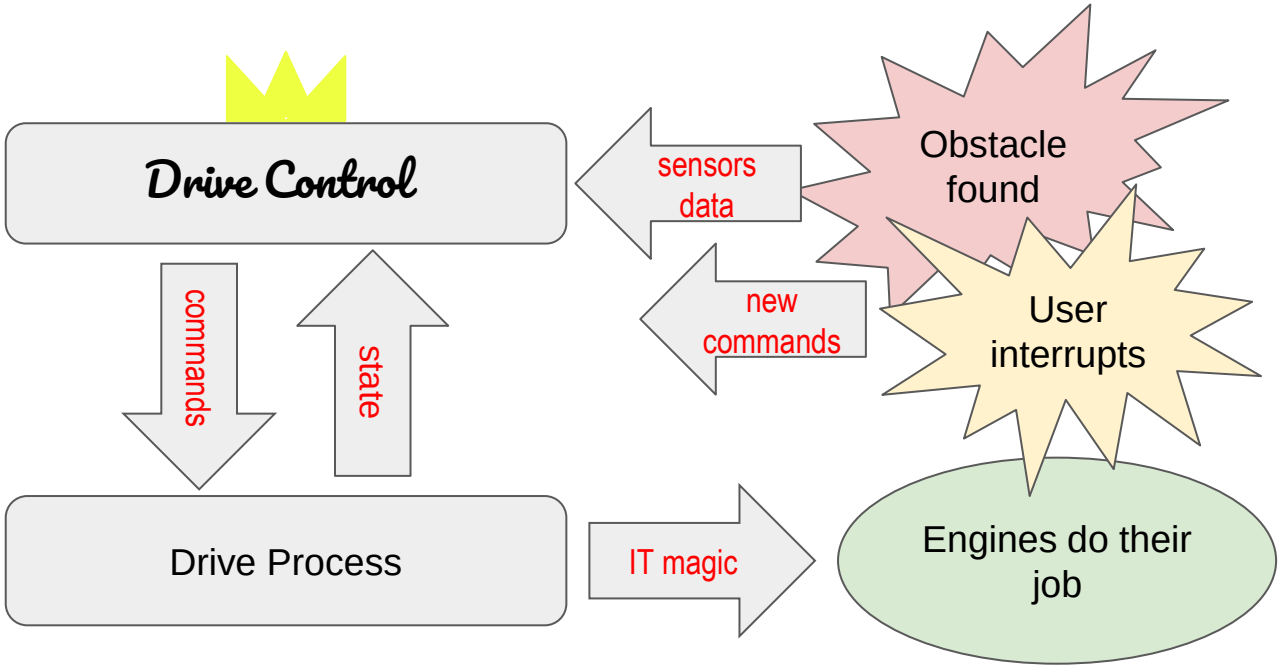
What’s the robot?

- Industrial Robot Cleaner
  - → Big and heavy
- Has “eyes”
  - → 4 cameras
  - → 2 lidars
  - → 8 IR proximity sensors
- Can’t touch it!
  - → Has no bumper or any other physical touch sensor

Main goals

- Moving along a pre-prepared route
- Avoiding obstacles
- Returning to the route after avoiding an obstacle
- Computing speed matters

Motion realization and logic



Drive Control

- Monitoring data from sensors
- Sends commands to Drive Process

If necessary:

- Stops previous command
- Creates new commands

Commands

- Simple
  - go : value
  - left/right: value
  - stop

- Complex
  - start wash program
  - park
  - goto: coordinates

Obstacle avoidance

- detecting an obstacle
- creating commands to avoid it
- finish avoiding (special state parameter)

Returning to route

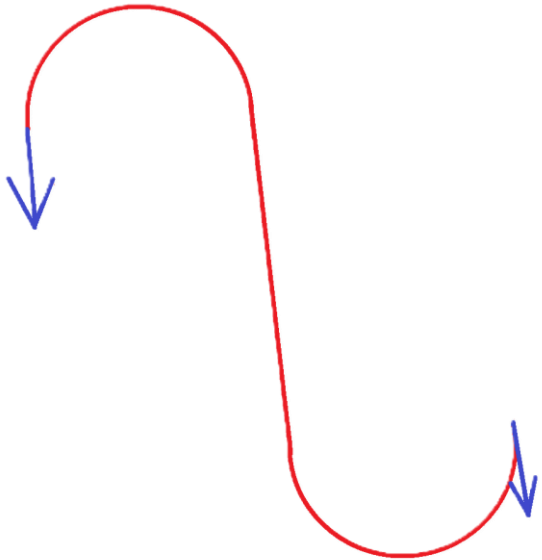
- calculating point to return to (closest one)
- “goto” command

GoTo

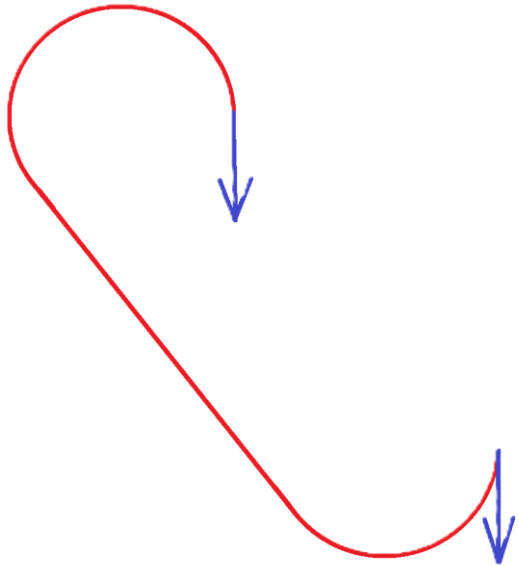
- Destination coords equals a vector (x; y; angle_z)
- Current coords is getting from Drive Service process
- Complex
  - Is to be processed into simple movement commands
    - turn
    - go
    - turn again

Trajectory

- Two arcs and a strait line



Different direction turns



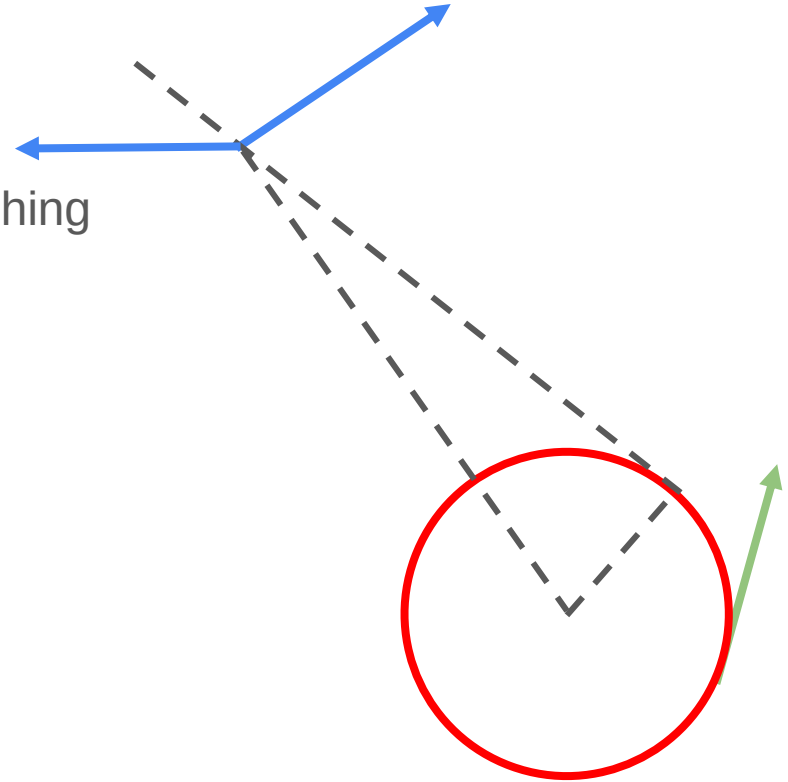
Same direction turns

Choosing first turn side

- building a strait line from start to finish
- calculating declination angle
  - angle_03 = atan2 (x3 - x0; y3 - y0)
- first turn = angle_03 - angle_z
- turn < 0 → left else right

Choosing the trajectory type

- build 1st turn circle
- build a tangent from finishing point
- compare angles



a function

```
def get_rot_center(coords, turn):
    rad_z = rad(coords[2]) # conversion to radians
    if turn > 0:     # right turn
        x = coords[0] + r * cos(rad_z)
        y = coords[1] - r * sin(rad_z)
    else:  # left turn
        x = coords[0] - r * cos(rad_z)
        y = coords[1] + r * sin(rad_z)
    return [x, y]
```

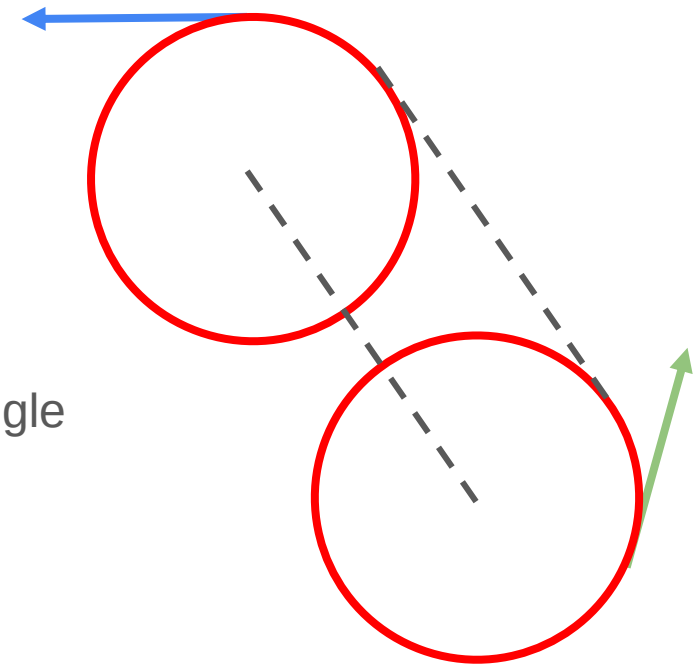
is it?

```
def is_inner_tangent(c0: list):

    a_cp3 = angle(c0,next_coords)

    B = deg(asin(r / length(c0, next_coords))) if t_00 > 0 \

        else -deg(asin(r / length(c0, next_coords)))

    angle_kp3 = B + a_cp3

    check = (round_a(next_coords[2] - angle_kp3) * t_00)


    return check < 0
```

Making final trajectory

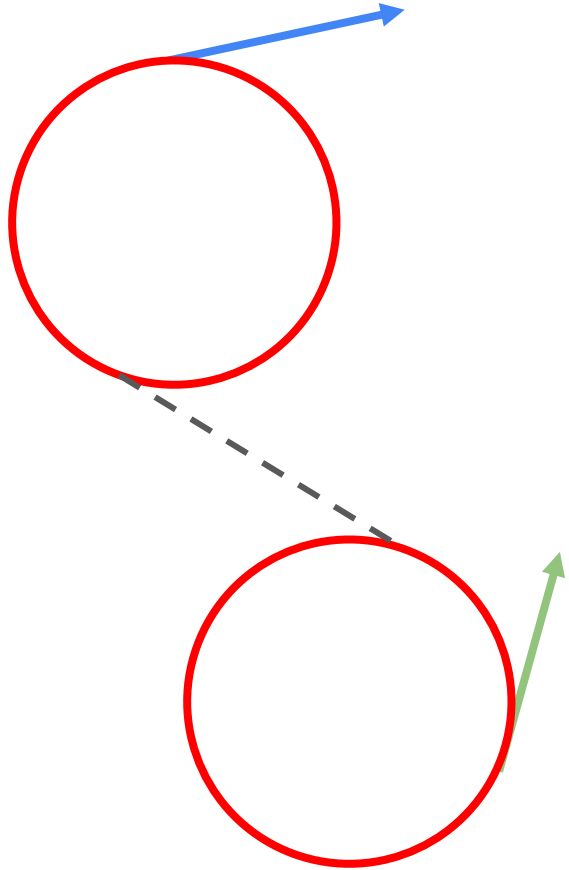
Outer tangent movement

- building up turn circles
- line between their centers
- first turn = centers line angle - start angle
- second turn = finish angle - centers line angle
- distance is length of center line



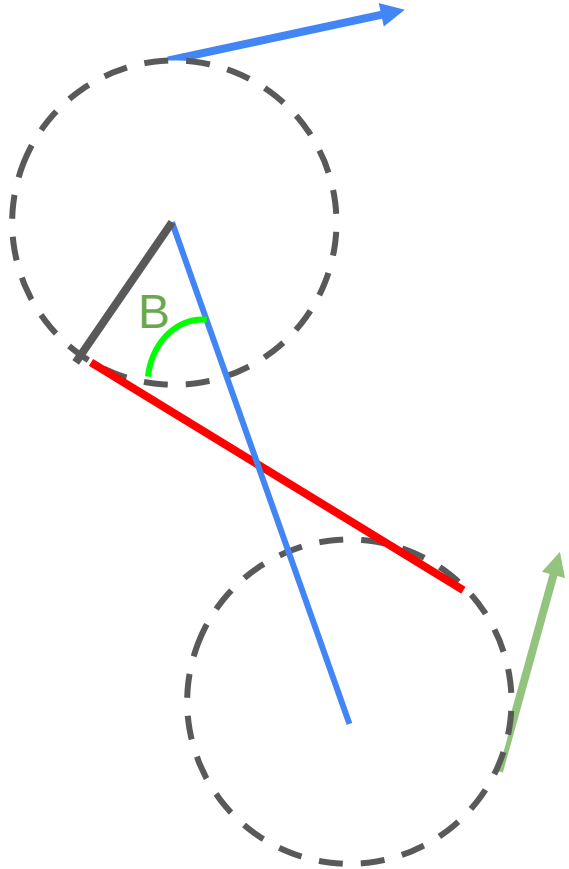
Inner tangent trajectory

- building up turn circles
- find their inner tangent
- first turn = tangent angle - start angle
- second turn = finish angle - tangent
- distance is tangent length



Tangent calculations

- find angle and length of circles centers line
- angle B is found through arcsin
- tangent angle = B + centers line angle
  - now we can find our turns
- need to find p1 & p2 coords for tangent length
  - use get_rot_center() with negative turn



THE END ?