KÄRCHER

# 🤖 Robotics Interview Coding Challenge

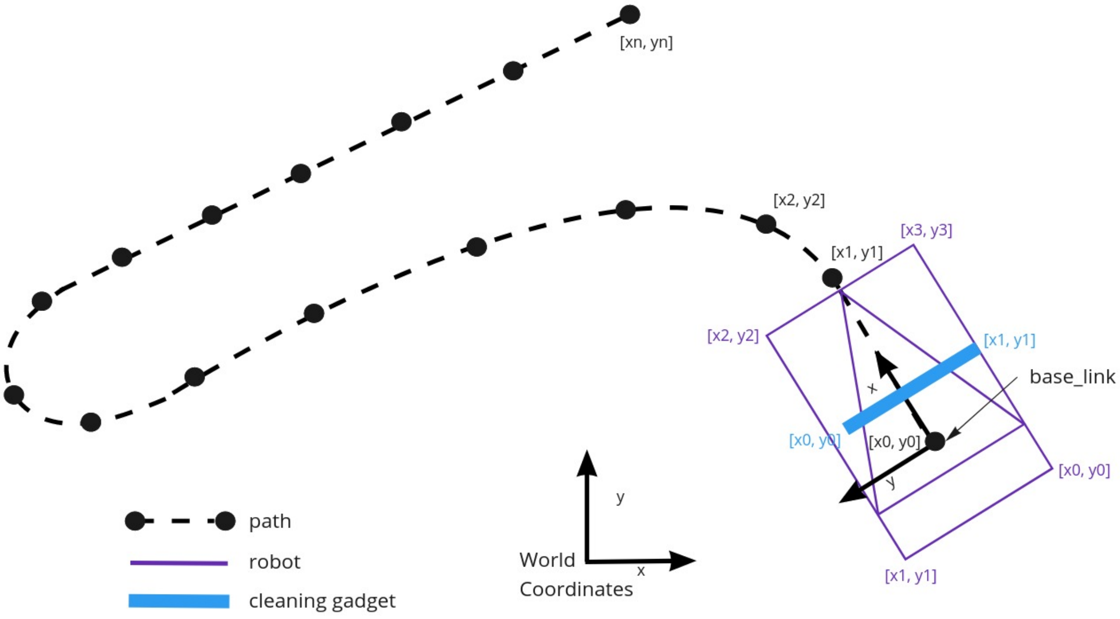
Given:

- An input JSON file (`short.json`), containing a path, robot polygon and a cleaning gadget line
- Line representing a cleaning gadget, and it's offset to the robot base
- Path velocity is proportional to the absolute curvature, so $v = v(\kappa)$ which is defined as

$$v(\kappa) = \begin{cases} v_{max} & , \kappa < \kappa_{crit} \\ v_{max} - \frac{v_{max} - v_{min}}{\kappa_{max} - \kappa_{crit}}(\kappa - \kappa_{crit}) & , \kappa_{crit} \leq \kappa < \kappa_{max} . \\ v_{min} & , \kappa \geq \kappa_{max} \end{cases}$$

whereby $\kappa_{crit}$ is the absolute critical curvature, starting from which, the robot has to reduce its speed to be able to make the curve safely. Here please assume, $\kappa_{crit} = 0.5[\frac{1}{m}]$. The curvature of the path will only have an effect up to $\kappa_{max}$, which is set to $10[\frac{1}{m}]$ in this example. When the path exceeds the maximum curvature, the robot speed is not longer affected and the minimal speed of $v_{min} = 0.15[\frac{m}{s}]$ can be assumed. On straights and flat curves the robot will drive with it's maximum speed, $v_{max} = 1.1[\frac{m}{s}]$.

```
{
// waypoints in 2D coordinates
"path": [
    [x0, y0],
    [x1, y1],
    ...,
    [xN, yN]
],
// polygon, origin is considered base link
"robot": [
    [x0, y0], [x1, y1], [x2, y2], [x3, y3]
],
// line in same coordinates as the robot
"cleaning_gadget": [
    [x0, y0], [x1, y1]
]
}
```



## Task1: C++ Computation

Write a modern, tested C++ program, which accepts the JSON file above as input and computes the following:

- The length of the path in $[m]$
- Cleaned area covered by the cleaning gadget in $[m^2]$, whereby overlapping areas count only once
- How long does the robot need to traverse the path given $v(\kappa)$

For the third part, consider that the test path was recorded using an actual robot, so to compute the curvature along the path, reasonable simplifications/approximations must be made.

## Task2: Data Visualization

Using a suitable known graphical framework, generate meaningful visualization of the results computed in **Task1**.

## General Notes and Requirements:

- Project has to compile using `CMake` and a reasonably modern Clang/GCC/MSVC (has to build on Linux, Windows optional)
- Share the project with us, such that we can easily obtain the sources
- Executable meaningful tests have to be provided along with the project, integrated into `ctest`
- As available dependencies `boost` can be assumed, all other dependencies must be provided along with the project or handled by the build system
- Document the project inside the `README.md` file