国立大学法人
静岡大学
National University Corporation
Shizuoka University

The POS Library

A Highly Customisable Coordinate System Library For C++

○ **Arjonilla García, Francisco Jesús**

Kobayashi, Yuichi

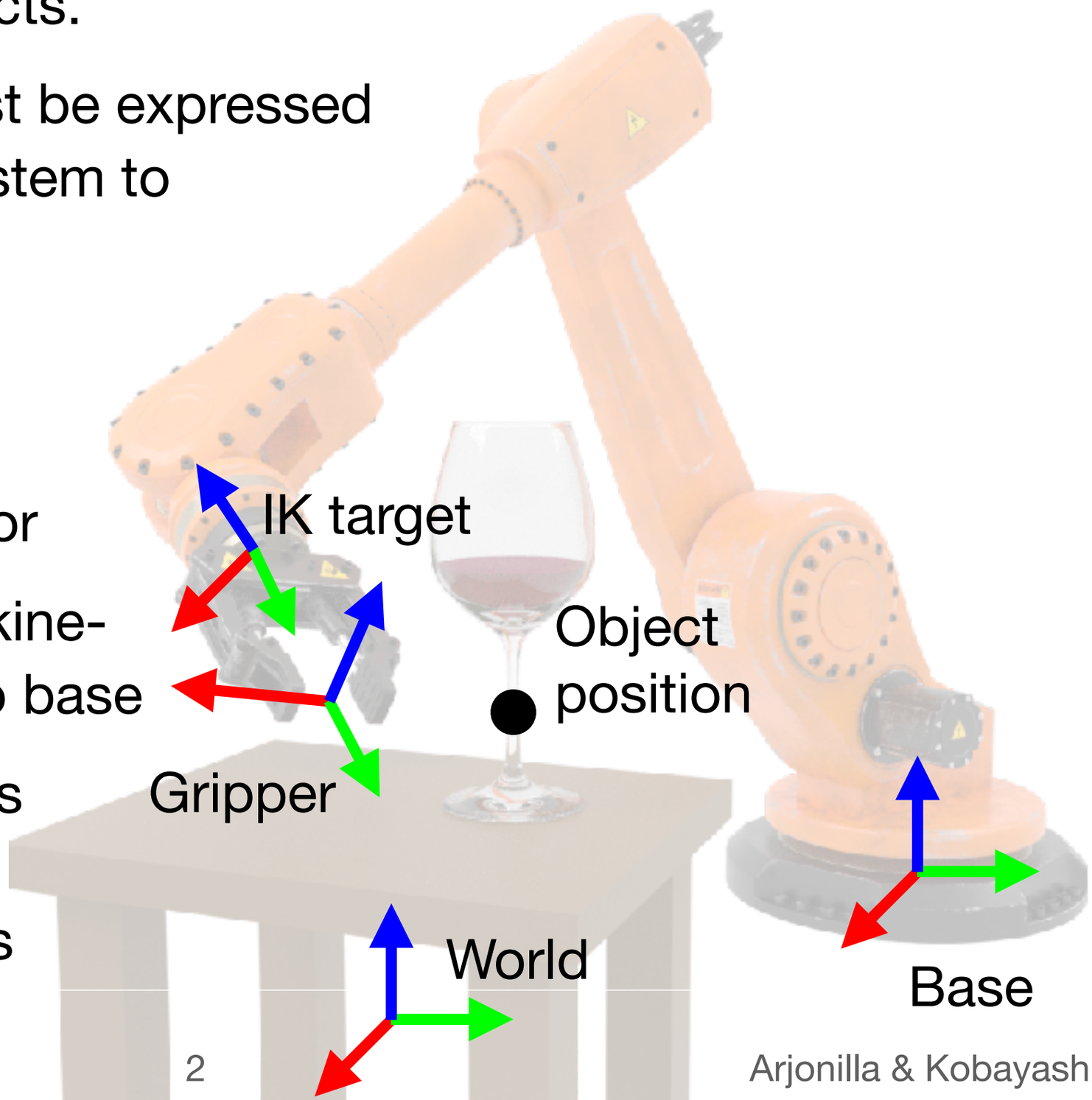
RSJ 2023
September 2023

The POS Library: Introduction

- In robotics, coordinates are often expressed in relation to different objects.
- However, coordinates must be expressed in the same coordinate system to operate with them.
- Example: A manipulator grabbing an object
  - Object relative to sensor
  - Calculation of inverse kinematics (IK) is relative to base
- Coordinate system libraries simplify calculation of coordinate transformations

The POS Library: Introduction

- In robotics, coordinates are often expressed in relation to different objects.
- However, coordinates must be expressed in the same coordinate system to operate with them.
- Example: A manipulator grabbing an object
  - Object relative to sensor
  - Calculation of inverse kinematics (IK) is relative to base
- Coordinate system libraries simplify calculation of coordinate transformations



The POS Library: Desired library features

The POS Library: Desired library features

Installation

Early error detection

Easy to use

Productivity

Beginners

Dependencies

Advanced

The POS Library: Desired library features

Easy to use

Depth sensors ~10^6 points

Low-power **Performant**

Embedded systems

The POS Library: Desired library features

Easy to use

Performant

License

Guarantees

Industry friendly

Standards

The POS Library: Desired library features

Easy to use

Portability

Standalone

BSD

Linux

OSX

Windows

VxWorks

Performant

Industry friendly

The POS Library: Desired library features

Easy to use

Portability

Performant

Industry friendly

Inter-operability

ROS2
OpenGL
OpenRTM
PyProj
ROS
Unity
Blender
MuJoCo

The POS Library: Desired library features

Easy to use

Portability

Generalised coordinates

Customisable

Custom inter-operability

Performant

Inter-operability

Industry friendly

The POS Library: Review

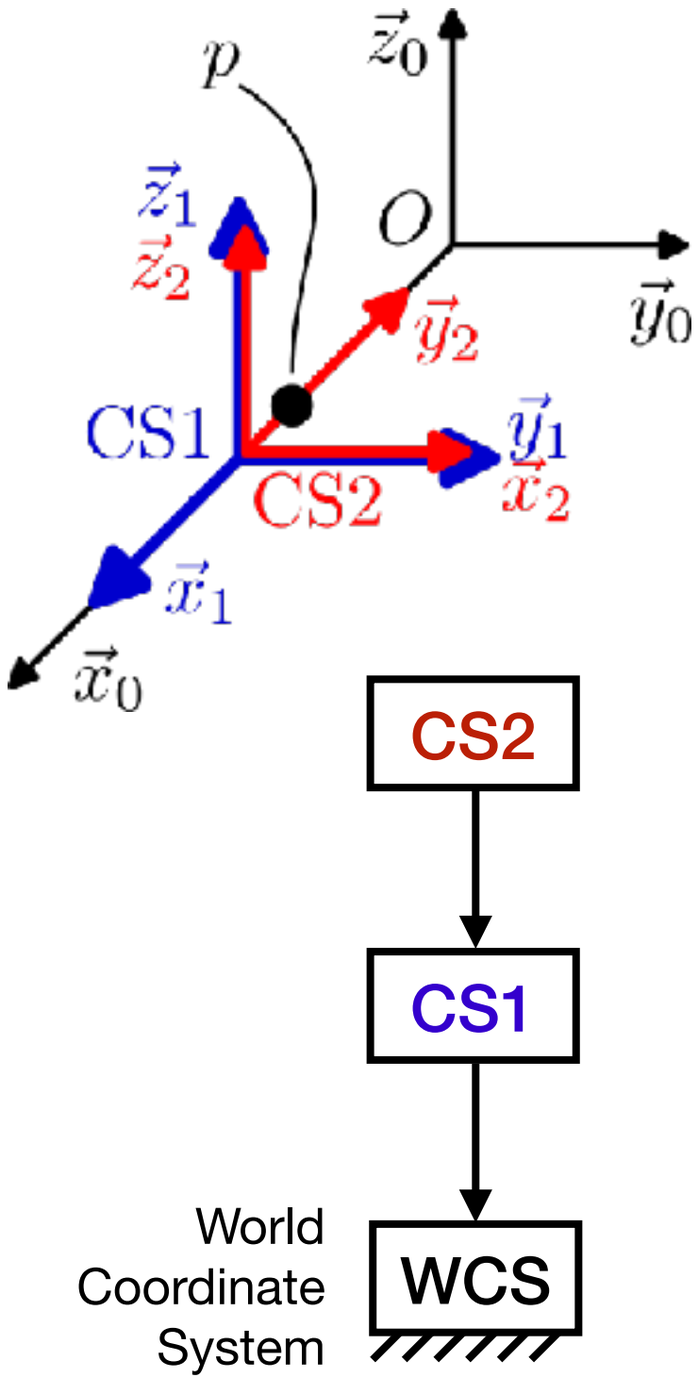
- There is no ideal coordinate system (CS) library.
- Many tools have specialised CS libraries.

	ROS	ROS2	MATLAB	PCL	Robotics Library	OpenGL	Ideal
Easy to use	△	△	△	○	△	△	○
Performant	△	△	△	○	○	○	○
Industry friendly	△	△	○	△	△		○
Portability		○		○	○		○
Inter-operability							○
Customisable							○

- The POS library pursues the ideal features and is based on:
  IEEE 1872-2015 *Standard Ontologies for Robotics and Automation*

The POS Library: Basic concepts

Example: Transform point in kinematic tree to World Coordinate System

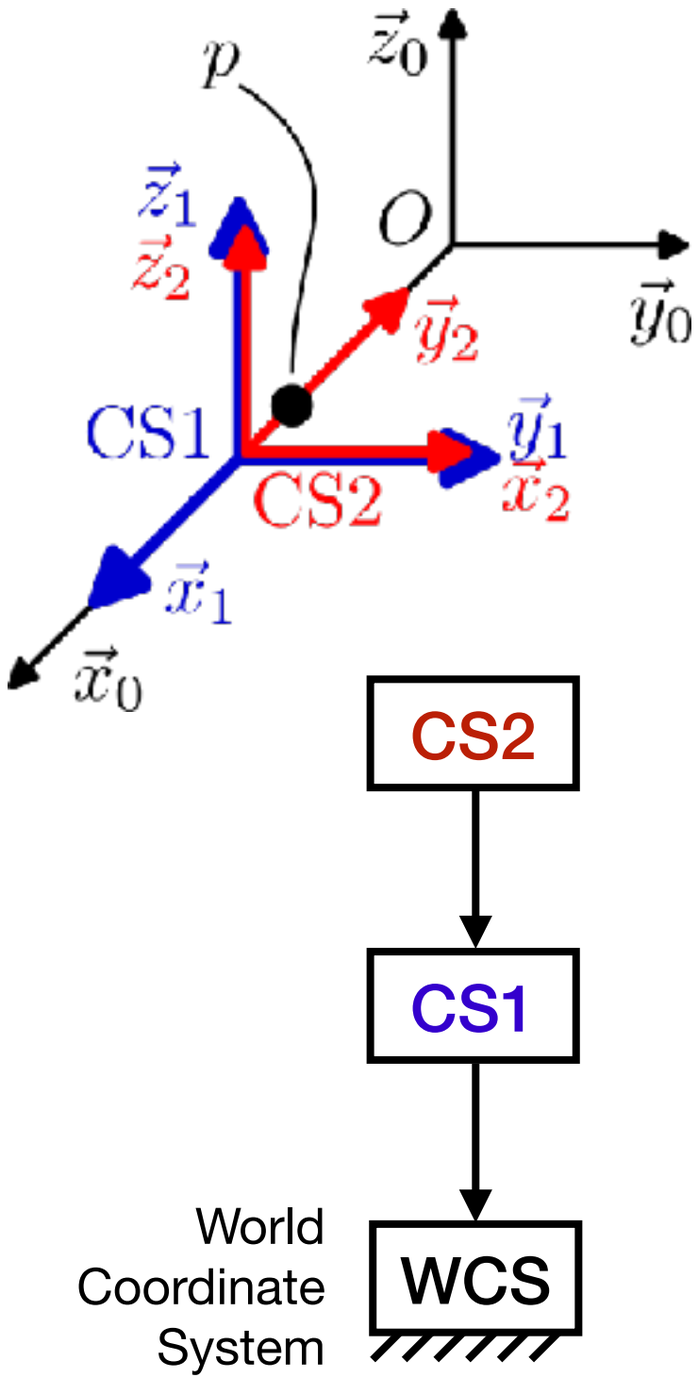



```
using namespace nin;

pos_coordsys_child CS1 (WCS, pose({1_m, 0_m, 0_m}));
pos_coordsys_child CS2 (CS1, pose{ {},
          {0_rad, 0_rad, 90_deg, euler_order::XYZ});

point              p_CS2 = {0_m, 10_cm, 0_m};


position_value v_CS2 = {CS2, p_CS2};
pos_coord_tf   tf    = mapCS(CS2, WCS);
position_value v_WCS = tf(v_CS2);
point          p_WCS = v_WCS.qty; // Quantity



std::cout << "x = " << p_WCS.X().SI() << " metres\n";
```

Output: `x = 0.9 metres`

The POS Library: Basic concepts

Example: Transform point in kinematic tree to World Coordinate System

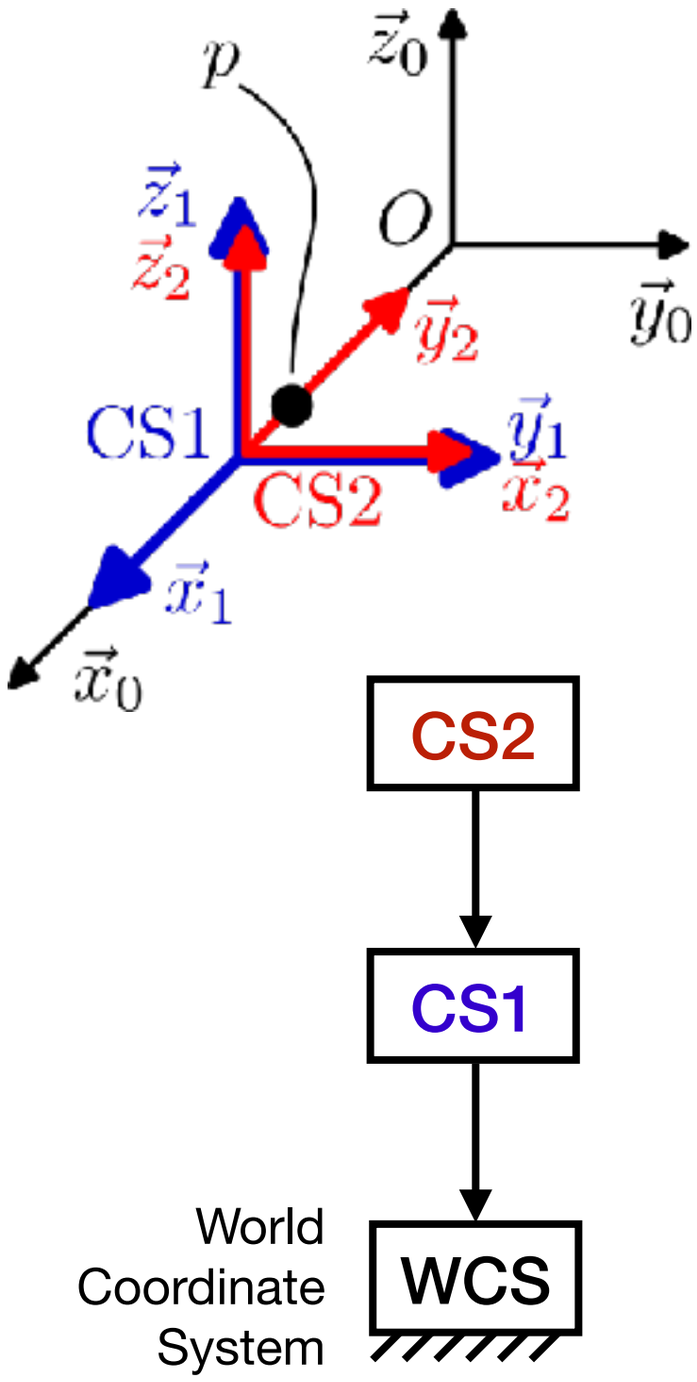



```
using namespace nin;

pos_coordsys_child CS1 (WCS, pose({1_m, 0_m, 0_m}));
pos_coordsys_child CS2 (CS1, pose{ {},
          {0_rad, 0_rad, 90_deg, euler_order::XYZ});

point          p_CS2 = {0_m, 10_cm, 0_m};


position_value v_CS2 = {CS2, p_CS2};
pos_coord_tf   tf    = mapCS(CS2, WCS);
position_value v_WCS = tf(v_CS2);
point          p_WCS = v_WCS.qty; // Quantity

point p_WCS = mapCS( position_value{CS2,p_CS2}, WCS );

std::cout << "x = " << p_WCS.X().SI() << " metres\n";
```

Output: `x = 0.9 metres`

The POS Library: Basic concepts

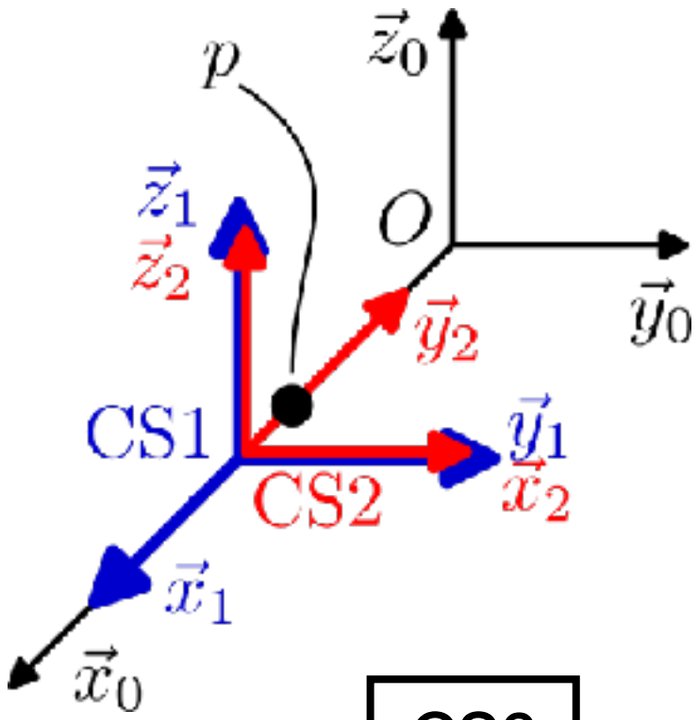
Example: Transform point in kinematic tree to World Coordinate System

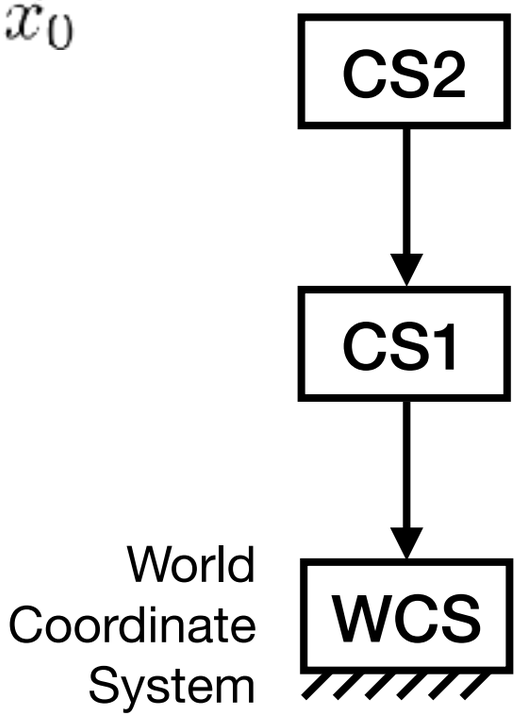



```
using namespace nin;

pos_coordsys_child CS1 (WCS, pose({1_m, 0_m, 0_m}));
pos_coordsys_child CS2 (CS1, pose{ {},
          {0_rad, 0_rad, 90_deg, euler_order::XYZ});

point              p_CS2 = {0_m, 10_cm, 0_m};


position_value v_CS2 = {CS2, p_CS2};
pos_coord_tf   tf    = mapCS(CS2, WCS);
position_value v_WCS = tf(v_CS2);
point          p_WCS = v_WCS.qty; // Quantity

point p_WCS = mapCS( position_value{CS2,p_CS2}, WCS );

std::cout << "x = " << p_WCS.X().SI() << " metres\n";
```

Output: `x = 0.9 metres`

The POS Library: mapCS( )






```
pos_coord_tf tf = mapCS(CS_from, CS_to);
```

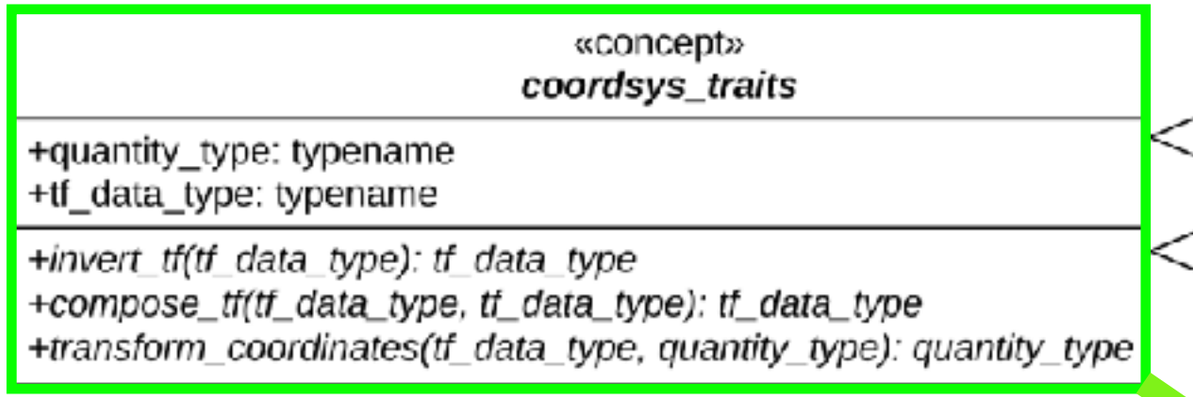
`mapCS()`:

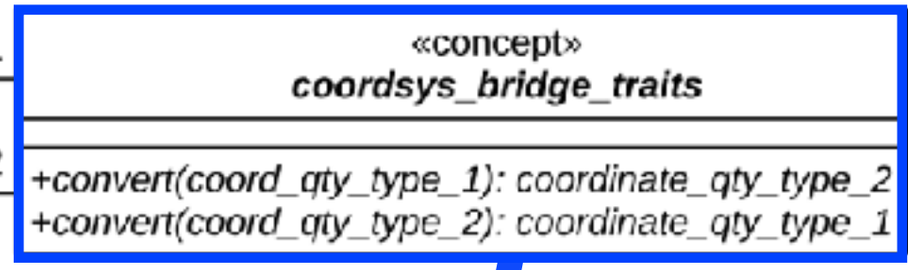
- **Generates** transform functions
- Queries both coordinate systems
  - Which in turn query their parents
  - Or retrieves data from ROS, etc.
- **Composes** offsets appropriately
- Carries the performance burden (network, etc)

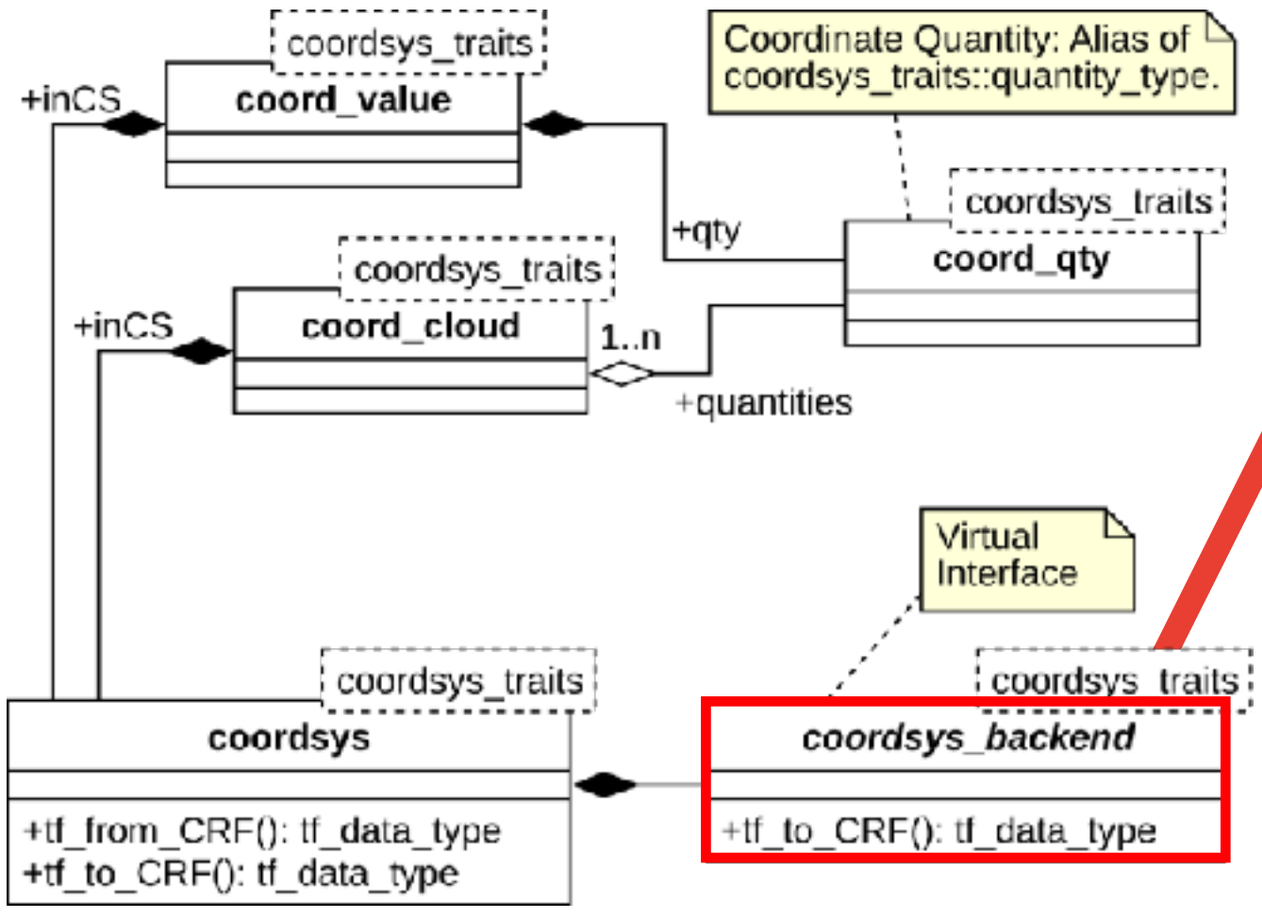
A transform function `coord_tf<>`:

- Is **highly optimized**
- Execution is constant-time
- Accepts point values and **point clouds**
- Needs to be re-generated when the coordinate system tree changes

The POS Library: Advanced Customisation







- Instantiate concept coordsys_traits to add custom generalised coordinate types, *e.g.* joint space coordinates.

- Derive class coordsys_backend to add custom connections to other frameworks, *e.g.* ROS, OpenRTM.

- Instantiate coordsys_bridge_traits to use different coordinate system types in the same kinematic tree

```
IK = mapCS(sensor_CS, bridge, joint_CS);

joints_value p = IK(pose);
```

The POS Library: Features

Easy to use

Portability

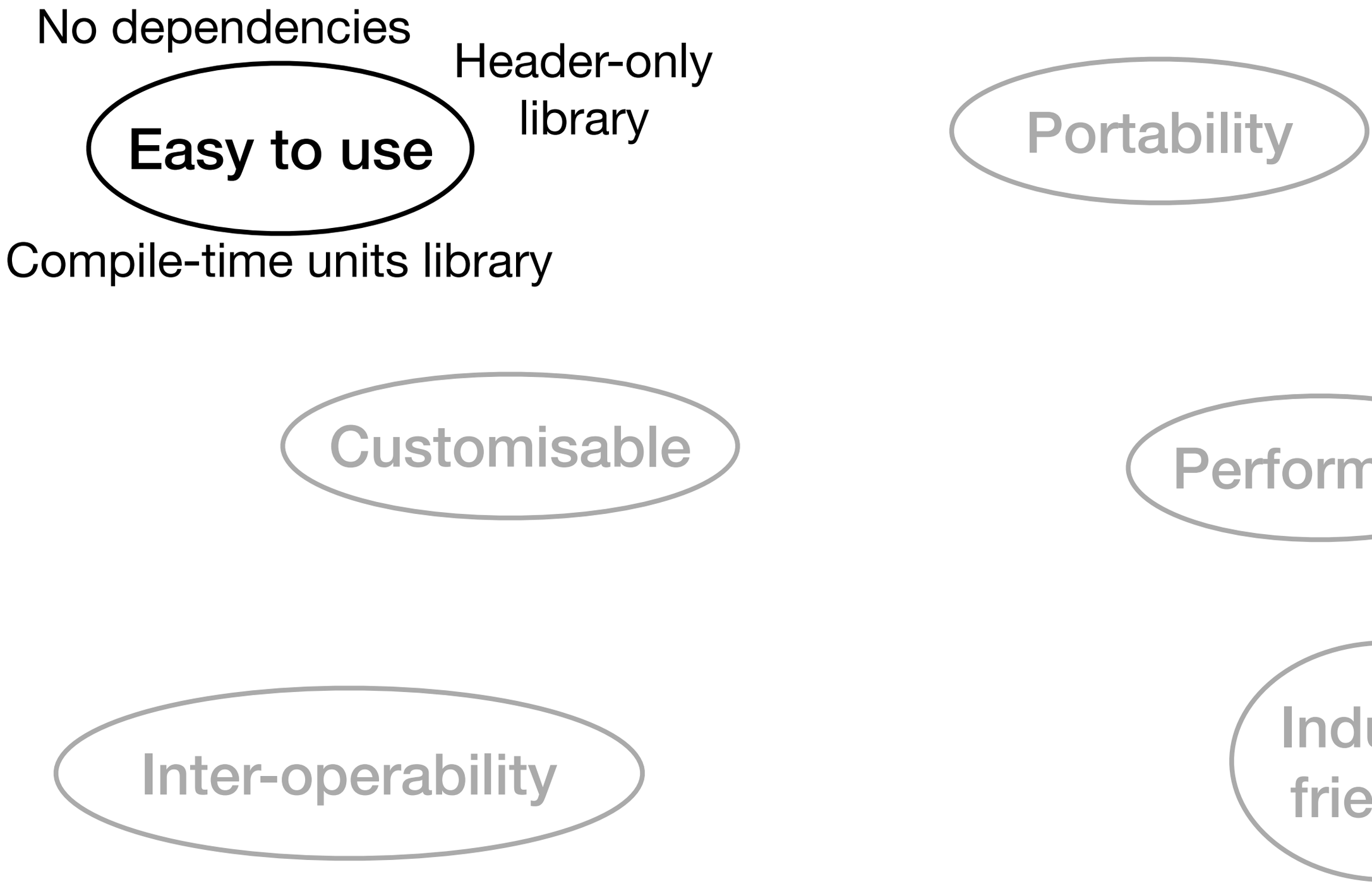
Customisable

Performant

Inter-operability

Industry friendly

The POS Library: Features



The POS Library: Features

No dependencies

Header-only library

Easy to use

Compile-time units library

Portability

Customisable

Transform functions

Performant

Inter-operability

Industry friendly

The POS Library: Features

No dependencies

Header-only library

Easy to use

Compile-time units library

Portability

Customisable

Transform functions

Performant

SI

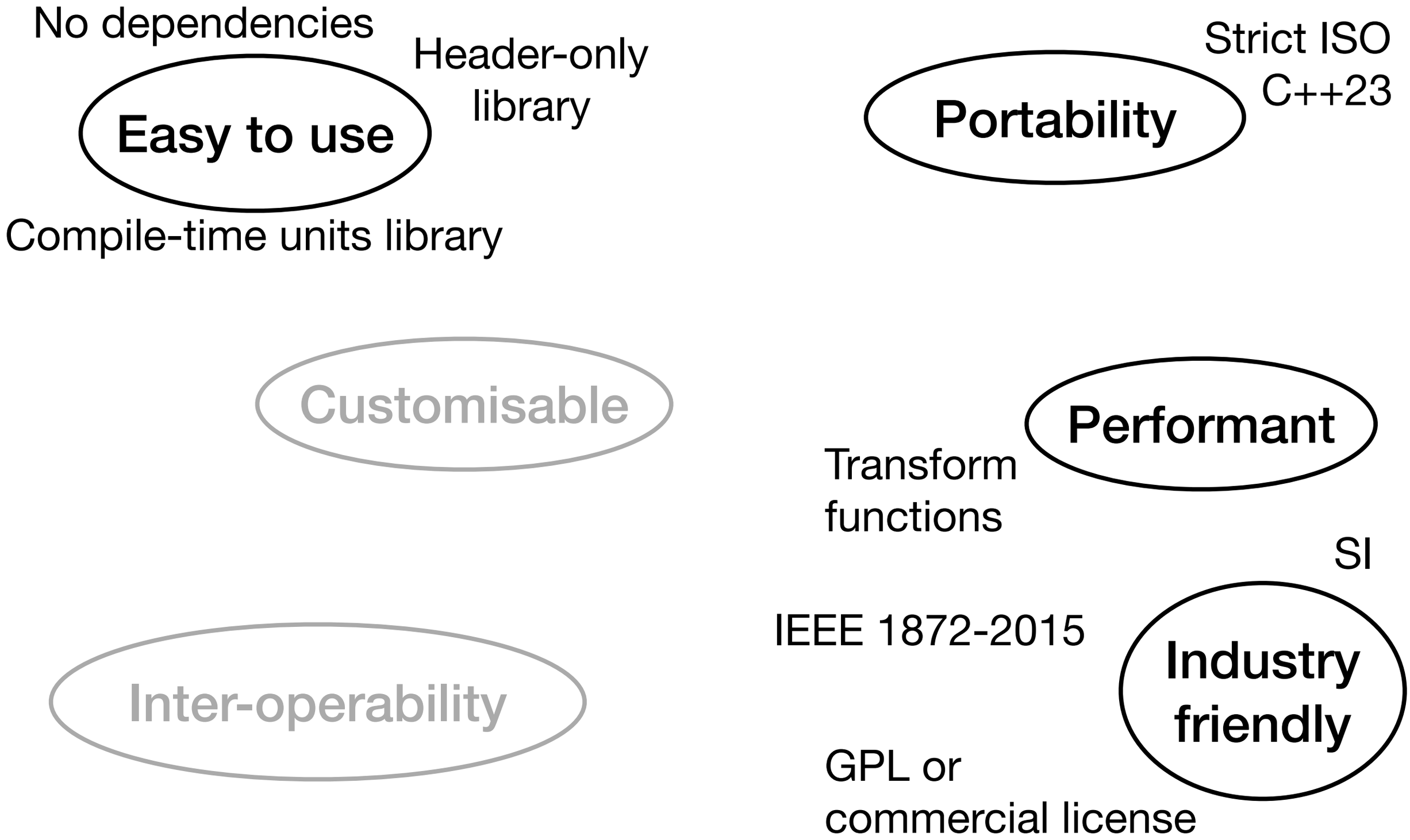
IEEE 1872-2015

Industry friendly

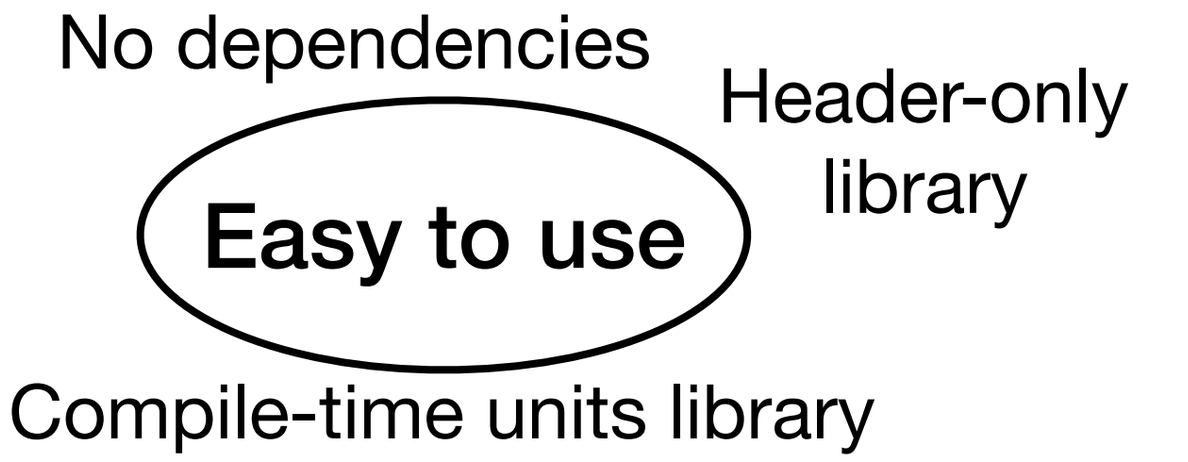
Inter-operability

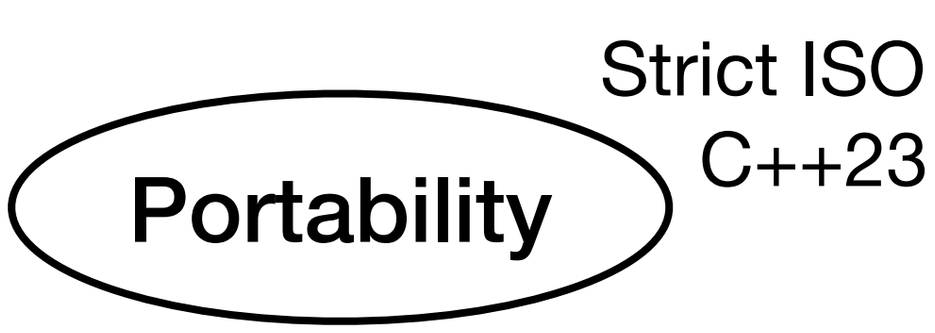
GPL or commercial license

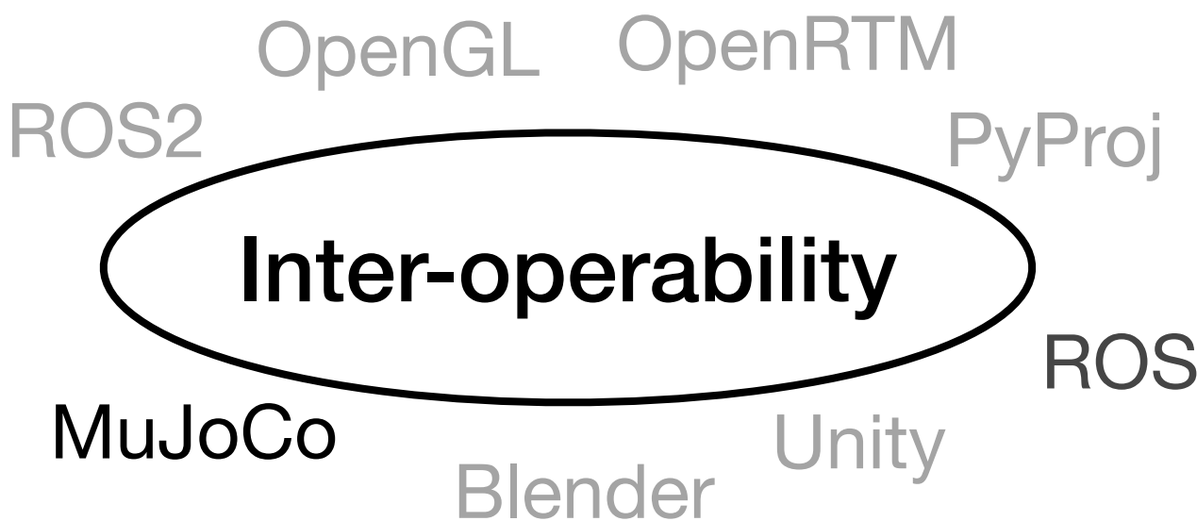
The POS Library: Features

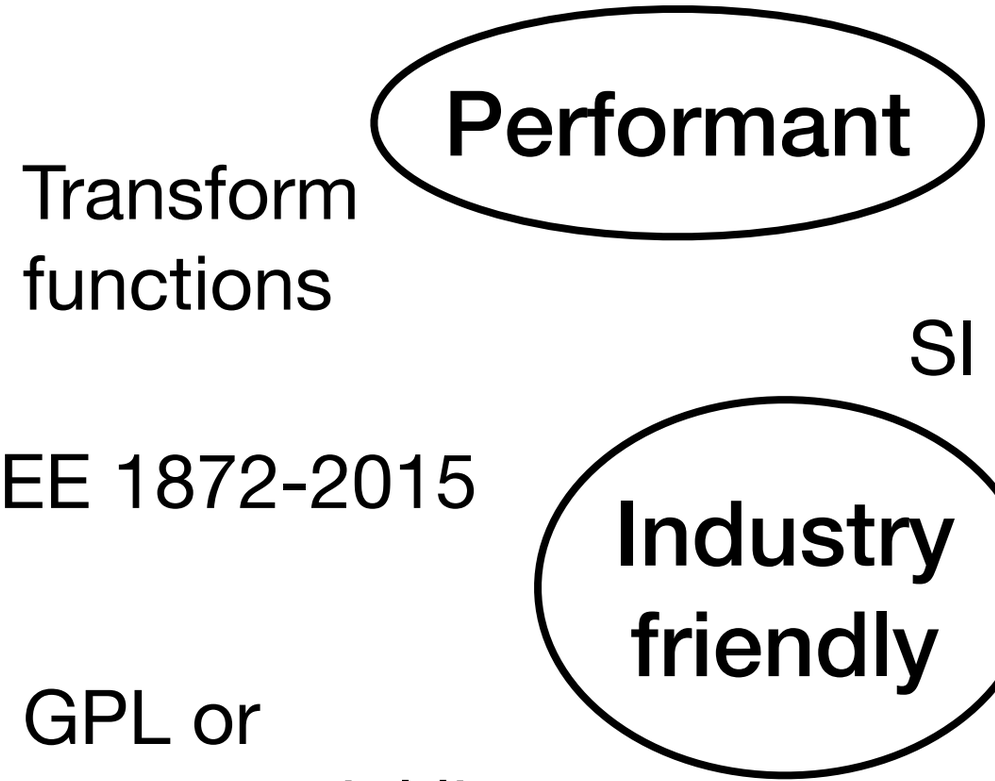


The POS Library: Features









The POS Library: Features

https://www.gitlab.org/ninbot/pos

Easy to use

No dependencies

Header-only library

Compile-time units library

Portability

Strict ISO C++23

User-defined coordinate traits

Customisable

User-defined virtual backends

Performant

Transform functions

Inter-operability

OpenGL

OpenRTM

ROS2

PyProj

ROS

MuJoCo

Unity

Blender

SI

IEEE 1872-2015

Industry friendly

GPL or commercial license

国立大学法人 静岡大学 National University Corporation Shizuoka University

The POS Library

A Highly Customisable Coordinate System Library For C++

https://www.gitlab.org/ninbot/pos

Thank you for your attention!

RSJ 2023
September 2023

○ **Arjonilla García, Francisco Jesús**
Kobayashi, Yuichi