Workshop on Geometric Relations between Rigid Bodies: semantics for standardization and software

Tinne De Laet, Herman Bruyninckx, Joris De Schutter

February, 2013

Overview

Overview

Introduction

Problem statement

Unclear semantics for rigid body geometric relations
(relative position, orientation, pose, translational, rotational velocity, twist, force, torque, and wrench)

→ leads to hidden assumptions, errors in calculations, ...

→ leads to system integration errors

Contribution

- **minimal yet complete** semantics for geometric relations
- **software support** for geometric relation calculations including **semantic checking**

Effect

- clear definition → standardization of definition and notation
- helps researchers to reveal hidden assumptions
- software support to prevent errors + help system integration

Geometric relations between rigid bodies

- relative position + orientation $\rightarrow$ Relative pose
- relative translational + rotational velocity $\rightarrow$ Relative twist

$\rightarrow$ no standardized definition of these relations
$\rightarrow$ often used without making all assumptions explicit

Geometric relations between rigid bodies

example

ROS: geometry_msgs/Pose

{ Position x, y, z; Orientation x, y, z, w}

- the pose of which body is expressed relative to which body?
- which points on the bodies are used to express their relative position?
- which orientation frames on the bodies are used to express their relative orientation?
- in which coordinate frame are the coordinates expressed?

Common errors

logic errors

- inverse of position $e|\mathcal{C}$ w.r.t $f|\mathcal{D}$ $(\mathrm{Position}\,(e|\mathcal{C}, f|\mathcal{D}))$ = position $f|\mathcal{D}$ w.r.t $e|\mathcal{C}$ $(\mathrm{Position}\,(f|\mathcal{D}, e|\mathcal{C})$

  $$\updownarrow$$

  inverse of linear velocity $e|\mathcal{C}$ w.r.t $\mathcal{D}$ $(\mathrm{LinearVelocity}\,(e|\mathcal{C}, \mathcal{D}))$ = linear velocity $e|\mathcal{D}$ w.r.t. $\mathcal{C}$ $(\mathrm{LinearVelocity}\,(e|\mathcal{D}, \mathcal{C}))$
- composing the relations involving three rigid bodies: geometric relation $\mathcal{C}$ w.r.t $\mathcal{D}$ = composition of geometric relation between $\mathcal{C}$ and a third body $\mathcal{E}$ with geometric relation between $\mathcal{E}$ and body $\mathcal{D}$ (and not between $\mathcal{D}$ and $\mathcal{E}$ for instance).

$\rightarrow$ **when using the semantic representation, semantics of result of inverse can be found automatically**

Common errors

composition of twists with different velocity reference points

composing twists requires a common point (i.e. the twists have to express the linear velocity of the same point on the body)
$\rightarrow$ **when including the velocity reference point in semantic representation, constraint can be checked explicitly**

Common errors

composition of geometric relations expressed in different coordinate frames

composing geometric relations requires that the coordinates are expressed in the *same coordinate frame*
$\rightarrow$ **when including the coordinate frame in semantic representation, constraint can be checked explicitly**

Common errors

composition of poses and orientation coordinate representations in wrong order

the rotation matrix and homogeneous transformation matrix coordinate representations can be composed using simple multiplication. Since matrix multiplication is however not commutative, a common error is to use a wrong multiplication order in the composition.
$\rightarrow$ **correct multiplication order can be directly derived when bodies, frames, and points are included in semantic representation**

Common errors

integration of twists when point and coordinate frame do not belong to same frame

a twist can only be integrated when it expresses the linear velocity of the origin of the coordinate frame the twist is expressed in.
$\rightarrow$ **when including point and coordinate frame in semantic representation of twist, constraint can be checked explicitly**

Introduction

contribution

- **minimal yet complete** semantics for geometric relations
- **software support** for geometric relation calculations including **semantic checking**

effect

- clear definition $\rightarrow$ standardization of definition and notation
- helps researchers to reveal hidden assumptions
- software support to prevent errors + help system integration

Overview

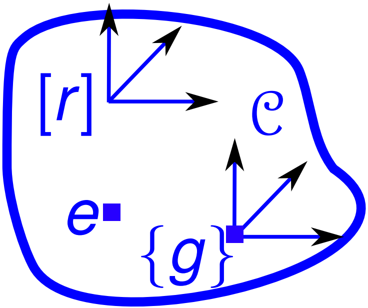
Geometric relations requirements

- express geometric relations between two rigid bodies:

  position, orientation, pose, translational velocity, rotational velocity, twist, force, torque, and wrench
- all expressions are **relative**, so they need a reference body to which they are expressed
- on both rigid bodies **points and orientation frames** have to be defined
- coordinates can only be interpreted correctly if it is clear in which **coordinate frame** they are expressed

Geometric primitives

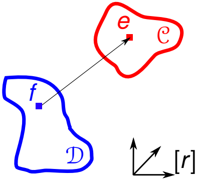
- body $\mathcal{C}$
- point e
- orientation frame $[r]$
- frame $\{g\}$
- point e fixed to body $\mathcal{C}$: $e \mid \mathcal{C}$



Overview

Relative position

- **Semantics:** $\mathrm{PositionCoord}\,(e|\mathcal{C}, f|\mathcal{D}, [r])$
  - expresses the position of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - i.e. between point $e \mid \mathcal{C}$ and point $f \mid \mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$



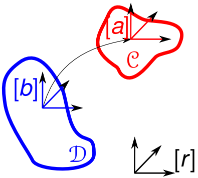
$\rightarrow$ any coordinates representing a position can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$$_{[r]}\boldsymbol{p}^{f|\mathcal{D},e|\mathcal{C}} \quad \sim \quad \mathrm{PositionCoord}\,(e|\mathcal{C}, f|\mathcal{D}, [r])$$

Relative orientation

- **Semantics:** $\mathrm{OrientationCoord}\left([a]|\mathcal{C}, [b]|\mathcal{D}, [r]\right)$
  - Expresses the orientation of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - i.e. between orientation $[a] \mid \mathcal{C}$ and orientation $[b] \mid \mathcal{D}$
  - Coordinates expressed in coordinate frame $[r]$



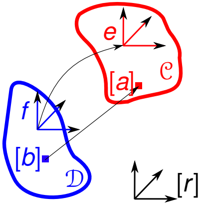
$\rightarrow$ any coordinates representing an orientation can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation**:

$$^{[a]|\mathcal{C}}_{[b]|\mathcal{D}}\boldsymbol{R} \quad \sim \quad \mathrm{OrientationCoord}\left([a]|\mathcal{C}, [b]|\mathcal{D}, [b]\right)$$

Relative pose

- **Semantics:** $\mathrm{PoseCoord}\left((a,[e])|\mathcal{C},(b,[f])|\mathcal{D},[r]\right)$
  - expresses the pose of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - i.e. the relative position between $e \mid \mathcal{C}$ and $f \mid \mathcal{D}$ and the relative orientation $[a] \mid \mathcal{C}$ and $[b] \mid \mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$



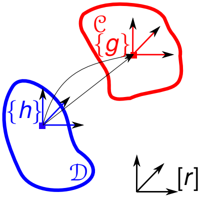
$\rightarrow$ any coordinates representing a pose can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$$\mathrm{PoseCoord}\left((e,[a])|\mathcal{C},(f,[b])|\mathcal{D},[r]\right)$$

Relative pose (2)

- **Semantics:** $\mathrm{PoseCoord}\left(\{g\}|\mathcal{C}, \{h\}|\mathcal{D}, [r]\right)$
  - expresses the pose of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - i.e. between $\{g\} \mid \mathcal{C}$ and $\{h\} \mid \mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$



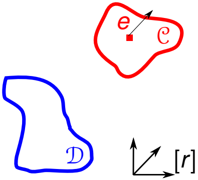
$\rightarrow$ any coordinates representing a pose can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$$ {}^{\{g\}|\mathcal{C}}_{\{h\}|\mathcal{D}}\boldsymbol{T} \quad \sim \quad \mathrm{PoseCoord}\left(\{g\}|\mathcal{C}, \{h\}|\mathcal{D}, [h]\right) $$

Relative linear velocity

- **Semantics:** $\mathrm{LinearVelocityCoord}\,(e|\mathcal{C}, \mathcal{D}, [r])$
  - expresses the linear velocity of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - i.e. between point $e \mid \mathcal{C}$ and body $\mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$
  - does not depend on point $f \mid \mathcal{D}$!



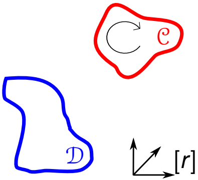
$\rightarrow$ any coordinates representing a linear velocity can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$$
\begin{aligned}
{}_{[r]}\dot{\boldsymbol{p}}^{\forall b|\mathcal{D}, a|\mathcal{C}} &\sim \mathrm{LinearVelocityCoord}\,(a|\mathcal{C}, \forall b|\mathcal{D}, [r]) \\
{}_{[r]}\dot{\boldsymbol{p}}^{\mathcal{D}, a|\mathcal{C}} &\sim \mathrm{LinearVelocityCoord}\,(a|\mathcal{C}, \mathcal{D}, [r])
\end{aligned}
$$

Relative angular velocity

- **Semantics:** $\text{AngularVelocityCoord}\left(\mathcal{C}, \mathcal{D}, [r]\right)$
  - expresses the angular velocity of body $\mathcal{C}$ *relative* to body $\mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$
  - does not depend on orientation frames chosen on $\mathcal{C}$ and $\mathcal{D}$!



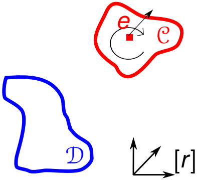
$\rightarrow$ any coordinates representing an angular velocity can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$$
\begin{aligned}
{}^{\forall[a]|\mathcal{C}}_{[r]}\boldsymbol{\omega}_{\forall[b]|\mathcal{D}} \quad &\sim \quad \text{AngularVelocityCoord}\left(\forall\,[a]|\mathcal{C}, \forall\,[b]|\mathcal{D}, [r]\right) \\
{}^{\mathcal{C}}_{[r]}\boldsymbol{\omega}_{\mathcal{D}} \quad &\sim \quad \text{AngularVelocityCoord}\left(\mathcal{C}, \mathcal{D}, [r]\right)
\end{aligned}
$$

Relative twist

- **Semantics:** $\mathrm{TwistCoord}\,(e|\mathcal{C}, \mathcal{D}, [r])$
  - expresses angular velocity of body $\mathcal{C}$ *relative* to body $\mathcal{D}$ *and*
  - expresses linear velocity of point $e \mid \mathcal{C}$, relative to body $\mathcal{D}$
  - coordinates expressed in coordinate frame $[r]$
  - does not depend on the orientation frames chosen on $\mathcal{C}$ and $\mathcal{D}$ or the point chosen on $\mathcal{D}$!



$\rightarrow$ any coordinates representing a twist can only be interpreted correctly if *all* the above information is present!

- **Symbolic notation:**

$${}^{e|\mathcal{C}}_{[r]}\mathbf{t}_{\mathcal{D}} \quad \sim \quad \mathrm{TwistCoord}\,(e|\mathcal{C}, \mathcal{D}, [r])$$

Forces, torques, and wrenches

- **parallel** between wrenches (torques and forces), and twists (linear and angular velocity) $\leftarrow$ screw theory
- $\hookrightarrow$ directly reflected in semantics

$$\begin{aligned} \text{torque} &\Leftrightarrow \text{linear velocity} \\ \text{force} &\Leftrightarrow \text{angular velocity} \end{aligned}$$

Coordinate representations

1. when doing actual calculations one has to use the coordinate representation of the geometric relations
2. particular coordinate representations can **impose constraints** on the semantics

examples

- rotation matrix

$$ {}^{[a]|\mathcal{C}}_{[b]|\mathcal{D}}\boldsymbol{R} \quad \sim \quad \mathrm{OrientationCoord}\left([a]|\mathcal{C}, [b]|\mathcal{D}, [b]\right) $$

- homogeneous transformation matrix

$$ {}^{\{g\}|\mathcal{C}}_{\{h\}|\mathcal{D}}\boldsymbol{T} \quad \sim \quad \mathrm{PoseCoord}\left(\{g\}|\mathcal{C}, \{h\}|\mathcal{D}, [h]\right) $$

Semantic operations

- semantic operations can compose geometric relations, change point, orientation frame, reference point, reference orientation frame, coordinate frame, ...
- operations **impose constraints** on operation arguments and on operand

Example

- **goal**= change point $\mathrm{PositionCoord}\left(e_1|\mathcal{C}, f|\mathcal{D}, [r]\right)$ from e_1 to point e_2

$$\mathrm{PositionCoord}\left(e_2|\mathcal{C}, f|\mathcal{D}, [r]\right) =$$
$$\mathrm{PositionCoord}\left(e_1|\mathcal{C}, f|\mathcal{D}, [r]\right).\mathrm{changePoint}\left(\mathrm{PositionCoord}\left(e_2|\mathcal{C}, e_1|\mathcal{C}, [r]\right)\right)$$

- **constraints**:
  1. argument of $.\mathrm{changePoint}()$ is PositionCoord geometric relation
  2. reference point argument = point of position operand
  3. body of argument = body of position operand
  4. reference body of argument = body of position operand
  5. coordinate frame of argument = point of the position operand

Overview

Software requirements

Software

In order for the standard to be usable, software supports is an absolute requirement.

- Need for semantic checking (during development and execution)
- If no need for checking: no or little overhead
- Semantic checking independent of the chosen coordinate representation (e.g. rotation matrix, quaternion, roll-pitch-yaw, . . .)
- Clear error messaging for semantic incorrect manipulation

Software output examples

Trying to compose two relative positions in the wrong way:

compose (PositionCoord ($a|\mathcal{C}$, $b|\mathcal{D}$, $[r]$) , PositionCoord ($a\_compose|\mathcal{E}$, $a|\mathcal{C}$, $[r\_wrong]$))

Composition of PositionCoord ($a|\mathcal{C}$, $b|\mathcal{D}$, $[r]$) and PositionCoord ($a\_compose|\mathcal{E}$, $a|\mathcal{C}$, $[r\_wrong]$) is NOK since:
*** Coordinate frame of PositionCoord ($a|\mathcal{C}$, $b|\mathcal{D}$, $[r]$) != coordinate frame of PositionCoord ($a\_compose|\mathcal{E}$, $a|\mathcal{C}$, $[r\_wrong]$)**

compose (PositionCoord ($a|\mathcal{C}$, $b|\mathcal{D}$, $[r]$) , PositionCoord ($a\_compose|\mathcal{E}$, $a|\mathcal{C}\_\mathcal{WRONG}$, $[r]$))

Composition of Position ($a|\mathcal{C}$, $b|\mathcal{D}$) and Position ($a\_compose|\mathcal{E}$, $a|\mathcal{C}\_\mathcal{WRONG}$) is NOK since:
*** Either the reference point and reference body of Position ($a|\mathcal{C}$, $b|\mathcal{D}$) have to be equal to the point and body of Position ($a\_compose|\mathcal{E}$, $a|\mathcal{C}\_\mathcal{WRONG}$) respectively**
OR
the point and body of Position ($a|\mathcal{C}$, $b|\mathcal{D}$) have to be equal to the reference point and reference body of Position ($a\_compose|\mathcal{E}$, $a|\mathcal{C}\_\mathcal{WRONG}$) respectively.

Software design - C++

Idea

Semantic checking for calculations with geometric relations **on top of existing geometric libraries**

Design

Each geometric relation $\Rightarrow$ four classes.

E.g. for Position:

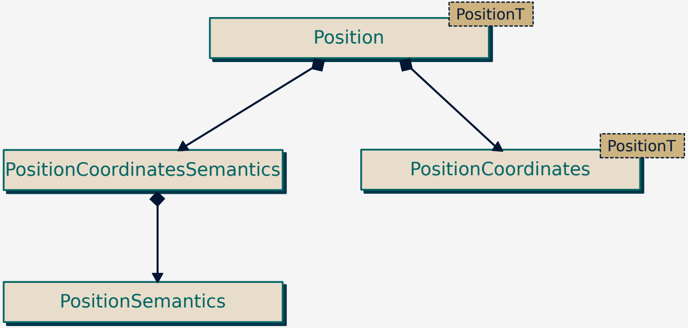
- **PositionSemantics**: semantics of the (coordinate-free) Position
- **PositionCoordinatesSemantics**: PositionSemantics object + coordinate frame semantics
- **PositionCoordinates**: templated class with actual coordinate representation
- **Position**: templated class = PositionCoordinatesSemantics + PositionCoordinates object

Software design - C++

Idea

Semantic checking for calculations with geometric relations **on top of existing geometric libraries**

Design



Enhancing your geometry library with semantics

- geometric semantics can be built on top of your geometry library
- only requires implementation of a limited number of template functions (composing, . . .)
- already supported: orocos_kdl and ros geometry

Application example

```
// Creating the geometric relations
// a Position with a KDL::Vector
Vector coordinatesPosition(1,2,3);
Position<Vector> position("a","C","b","D","r",
    coordinatesPosition);
// inverting
Position<Vector> positionInv = position.inverse();
// print the inverse
std::cout << "        " << positionInv << " is the inverse of
    " << position << std::endl;
//Composing
Position<Vector> positionComp = compose(position,
    positionInv);
 // print the composed object
std::cout << "        " << positionComp << " is the
    composition of " << position << " and " << positionInv
    << std::endl;
```

Application example

Output:

```
Position(b|D,a|C,[r])= [-1, -2,-3] is the inverse of Position(a
    |C,b|D,[r])= [1,2,3]

Composition of Position(a|C,b|D,[r]) and Position(b|D,a|C,[r])
    is OK.
Composition of Position(a|C,b|D) and Position(b|D,a|C) is OK.

Position(a|C,a|C,[r])= [0,0,0] is the composition of Position(a
    |C,b|D,[r])= [1,2,3] and Position(b|D,a|C,[r])= [-1,-2,-3]
```

Overview

Conclusion

Software

- proposal for semantics underlying geometric relationships between rigid bodies
- semantics make all underlying choices explicit
- c++ software support for semantic checking
- helps to avoid common errors
- helps during system integration

Overview

Introduction

goal

give you hands-on experience with geometric semantics: theory and software

- avoiding common errors
- helping system integration

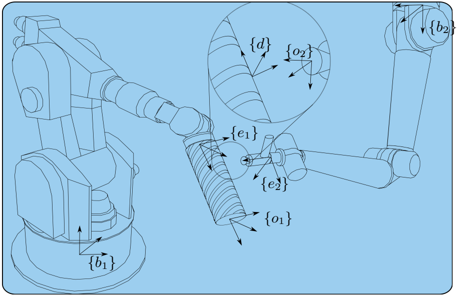
Overview

Tutorials

To get to know the software we will first follow the tutorials from the geometric semantics website.

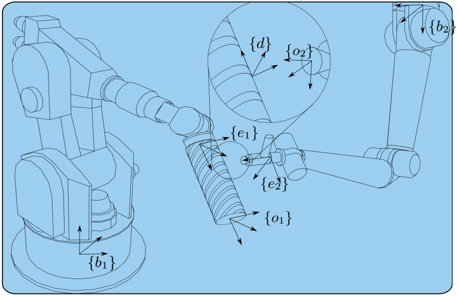
Overview

System integration tutorial: spray painting with two robots



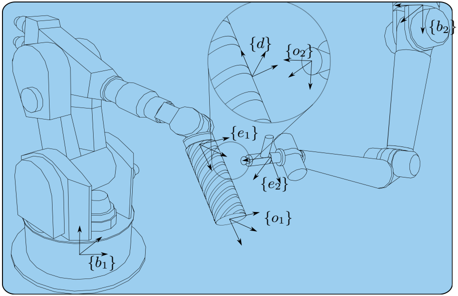
Goal = determine the joint angles of the second robot holding the spray gun such that a predefined pose between the spray gun and cylindrical object is obtained

System integration tutorial: spray painting with two robots



- $\{b_1\}$ attached to the base $\mathcal{B}_1$ of the first robot,
- $\{e_1\}$ attached to the end-effector $\mathcal{E}_1$ of the first robot,
- $\{o_1\}$ attached to cylindrical object $\mathcal{O}_1$,
- $\{b_2\}$ attached to the base $\mathcal{B}_2$ of the second robot,
- $\{e_2\}$ attached to the end-effector $\mathcal{E}_2$ of the second robot, and
- $\{o_2\}$ attached to the spray gun $\mathcal{O}_2$.

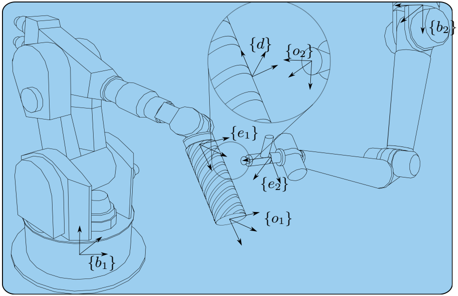
System integration tutorial: spray painting with two robots



In our example the following poses are available:

- $\text{PoseCoord}\left(\{e_1\}|\mathcal{E}_1, \{b_1\}|\mathcal{B}_1, [b_1]\right)$
- $\text{PoseCoord}\left(\{b_2\}|\mathcal{B}_2, \{b_1\}|\mathcal{B}_1, [b_1]\right)$
- $\text{PoseCoord}\left(\{o_1\}|\mathcal{O}_1, \{e_1\}|\mathcal{E}_1, [e_1]\right)$
- $\text{PoseCoord}\left(\{o_2\}|\mathcal{O}_2, \{e_2\}|\mathcal{E}_2, [e_2]\right)$
- $\text{PoseCoord}\left(\{o_2\}|\mathcal{O}_2, \{o_1\}|\mathcal{O}_1, [o_1]\right)$

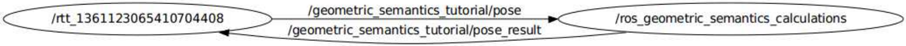
System integration tutorial: spray painting with two robots



In order to find the joint angles of the second robot the robot programmer has to find **PoseCoord** $(\{e_2\}|\mathcal{E}_2, \{b_2\}|\mathcal{B}_2, [b_2])$**,** and subsequently use the inverse kinematics of the second robot.

Software

- **two orocos components**
  1. **publisher:**
     1. publishes $\mathrm{PoseCoord}\left(\{e_1\}|\mathcal{E}_1, \{b_1\}|\mathcal{B}_1, [b_1]\right)$ on port
     2. runs periodically
     3. published pose available on topic geometric_semantics_tutorialpose
  2. **subscriber:**
     1. listens to topic geometric_semantics_tutorialpose_result
     2. runs aperiodically (wakes up when receiving a pose)
     3. checks if received pose has desired semantics ($\mathrm{PoseCoord}\left(\{e_2\}|\mathcal{E}_2, \{b_2\}|\mathcal{B}_2, [b_2]\right)$)
- **one ros node**
  1. **node:**
     1. listens to topic geometric_semantics_tutorialpose
     2. has to perform calculations to obtain $\mathrm{PoseCoord}\left(\{e_2\}|\mathcal{E}_2, \{b_2\}|\mathcal{B}_2, [b_2]\right)$
     3. publishes result to geometric_semantics_tutorialpose_result



Software

- **two orocos components**
  1. available in geometric_relations_semantics_tutorial_orocos_components package
  2. can be run using: rosrun ocl rttlua-gnulinux -i deploy_tutorial.lua
  3. look at output for checking if the semantics of your result is correct
- **one ros node**
  1. available in geometric_relations_semantics_tutorial_ros_nodes package
  2. can be run using: rosrun geometric_relations_semantics_tutorial_ros_nodes node
  3. fill in the necessary geometric calculations in function doGeometricSemanticsCalculations in node.cpp

Good luck!