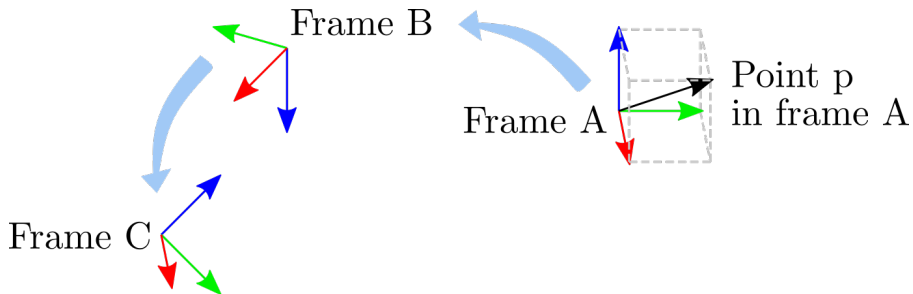


Transformations in Three Dimensions

Alexander Fabisch

DFKI GmbH, Robotics Innovation Center



pytransform3d



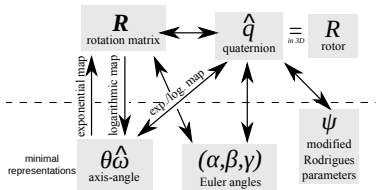
git

<https://github.com/dfki-ric/pytransform3d>

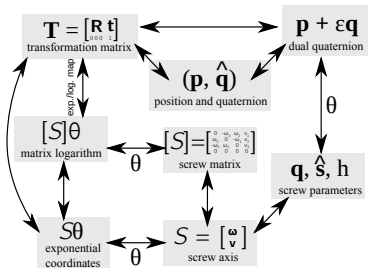
Why?

Why?

pytransform3d.rotations

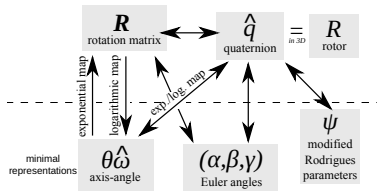


pytransform3d.transformations

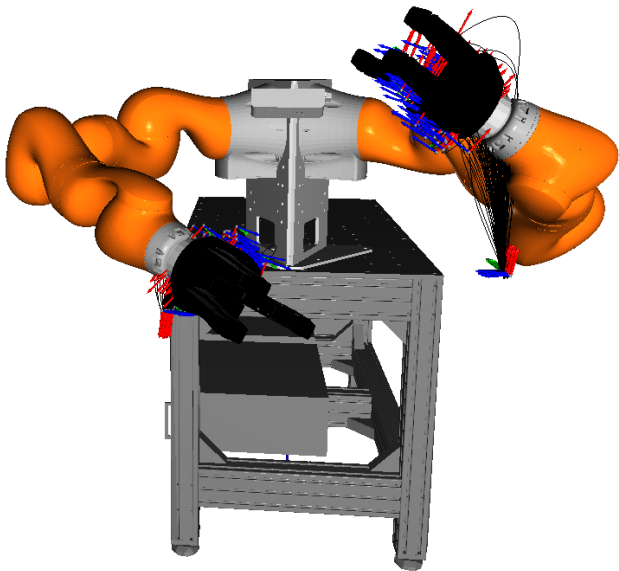
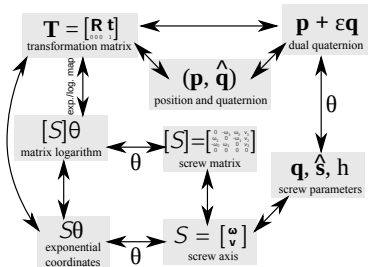


Why?


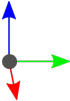
pytransform3d.rotations





pytransform3d.transformations



Introduction to 3D Rigid Transformations

Representation	Term	
\mathbb{R}^3	position	
$SO(3)$	orientation	
$SE(3)$	pose	

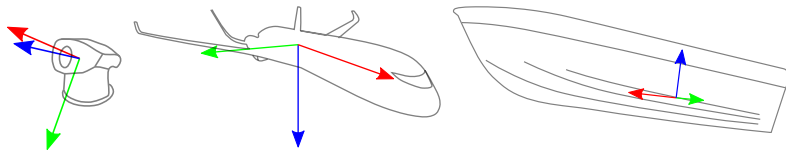
Introduction to 3D Rigid Transformations

Representation	Term	Displacement	
\mathbb{R}^3	position	translation	
$SO(3)$	orientation	rotation	
$SE(3)$	pose	rigid transformation	

Frames

A *coordinate reference frame* **frame** is...

- ▶ defined by position and orientation
- ▶ attached to a rigid body



We will use RGB arrows to display frames.

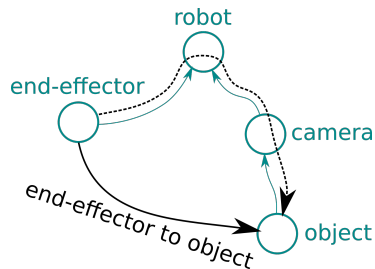
Graphs of Transformations

Examples: kinematic structures of robots (and other articulated bodies), estimated states, trajectories

```
from pytransform3d.transform_manager import
    TransformManager

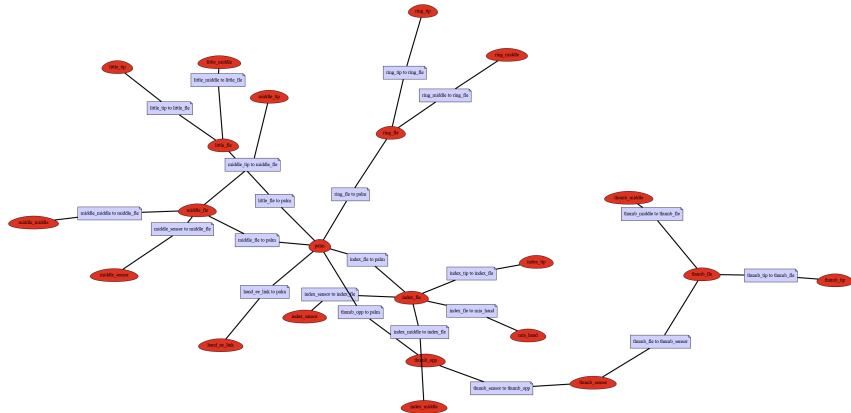
tm = TransformManager()
tm.add_transform("end-effector", "robot", ee2robot)
tm.add_transform("object", "camera", object2cam)
tm.add_transform("camera", "robot", cam2robot)

ee2object = tm.get_transform(
    "end-effector", "object")
```



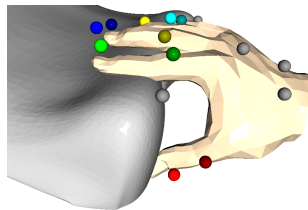
Robot Kinematics (Example: Robotic Hand)

```
from pytransform3d.urdf import UrdfTransformManager
tm = UrdfTransformManager()
with open("robot.urdf", "r") as f:
    robot_urdf = f.read()
tm.load_urdf(robot_urdf)
tm.write_png(graph_filename)
```



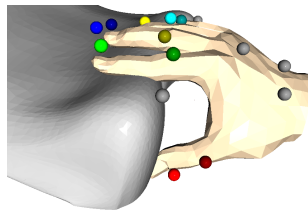
Application: Motion Transfer to Robotic Hands

Transfer of Motions to Robotic Hand



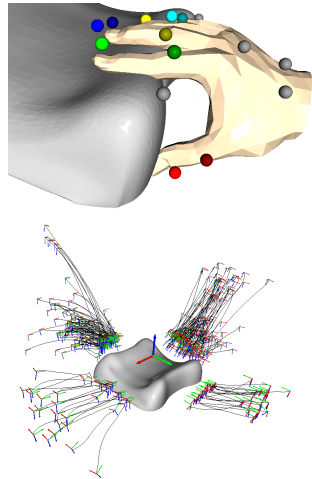
https://github.com/dfki-ric/hand_embodiment (Fabisch et al. 2022)

Transfer of Motions to Robotic Hand



https://github.com/dfki-ric/hand_embodiment (Fabisch et al. 2022)

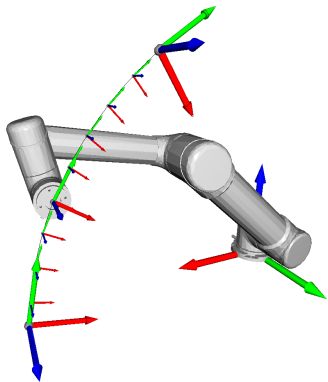
Transfer of Motions to Robotic Hand



https://github.com/dfki-ric/hand_embodiment (Fabisch et al. 2022)

Application: Imitation Learning

Imitation Learning



Given: one or more solutions to a problem (trajectories)

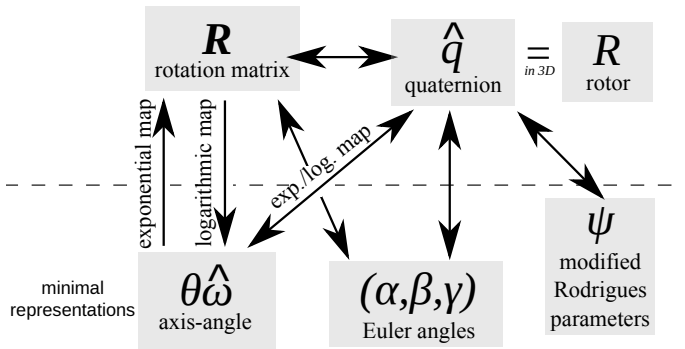
How can we represent orientation?

Library: https://github.com/dfki-ric/movement_primitives

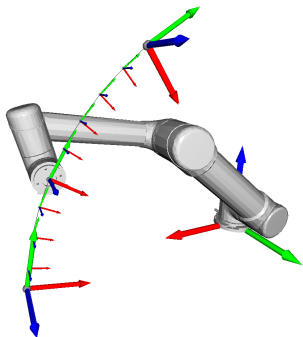
SO(3) (SO: special orthogonal group)

- group of all rotations in 3D
- represented by 3D rotation matrices

pytransform3d.rotations



Imitation Learning



Problems

- ▶ Rotation matrices have 6 constraints that are not easy to enforce
- ▶ Minimal representations $\in \mathbb{R}^3$ have discontinuities / singularities
- ▶ Quaternions $q \in \mathbb{S}^3$ have an ambiguity:
 $q \equiv -q$

Quaternions - Pitfalls

We want to use quaternions (Ude et al. 2014)

Quaternions - Pitfalls

We want to use quaternions (Ude et al. 2014)

Be careful:

- ▶ Quaternions $q \in \mathbb{S}^3$ have an ambiguity: $q \equiv -q$

```
from pytransform3d.batch_rotations import smooth_quaternion_trajectory  
quaternions = smooth_quaternion_trajectory(quaternions)
```

Quaternions - Pitfalls

We want to use quaternions (Ude et al. 2014)

Be careful:

- ▶ Quaternions $q \in \mathbb{S}^3$ have an ambiguity: $q \equiv -q$

```
from pytransform3d.batch_rotations import smooth_quaternion_trajectory
quaternions = smooth_quaternion_trajectory(quaternions)
```

- ▶ 2 conventions: **Hamilton** vs. JPL
(Sommer et al. 2018)

Quaternions - Pitfalls

We want to use quaternions (Ude et al. 2014)

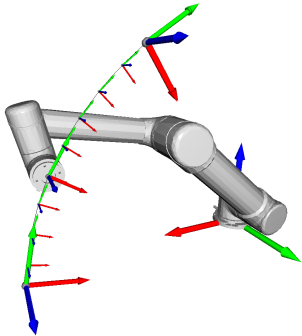
Be careful:

- ▶ Quaternions $q \in \mathbb{S}^3$ have an ambiguity: $q \equiv -q$

```
from pytransform3d.batch_rotations import smooth_quaternion_trajectory
quaternions = smooth_quaternion_trajectory(quaternions)
```

- ▶ 2 conventions: **Hamilton** vs. JPL
(Sommer et al. 2018)
- ▶ 4 numbers, e.g., $(0, 1, 0, 0)$
 - **scalar first** (w, x, y, z) or last (x, y, z, w)

Visualizer – matplotlib-like interface to Open3D



```
import pytransform3d.visualizer as pv

fig = pv.figure()

fig.plot_transform(np.eye(4), s=0.3)

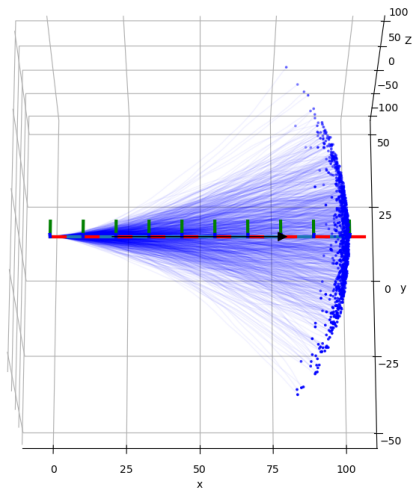
fig.plot_graph(
    urdf_transform_manager,
    "ur5_base_link",
    show_collision_objects=True,
    show_frames=True)

pv.Trajectory(trajecory).add_artist(fig)

fig.view_init()
fig.show()
```

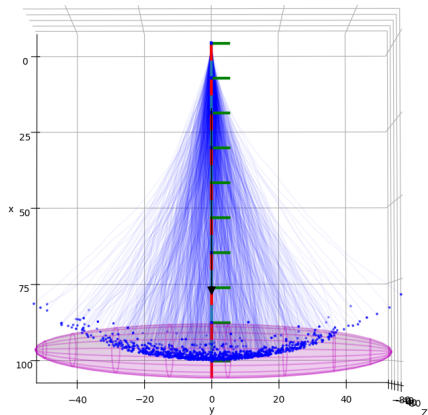
Application: Uncertain Transformations

Concatenation of Uncertain Transformations



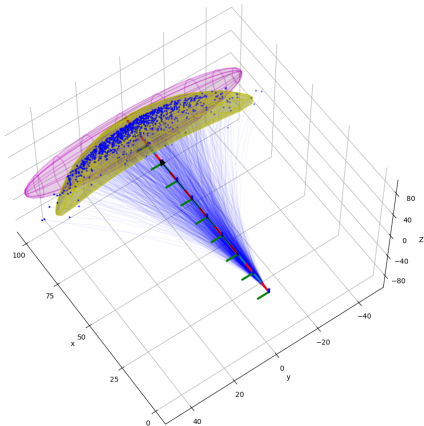
- Uncertainty of angular velocity about z-axis
- Blue: Monte Carlo (MC) sampling of trajectories

Concatenation of Uncertain Transformations



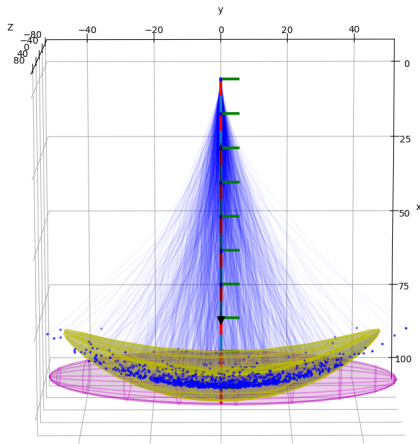
- Uncertainty of angular velocity about z-axis
- Blue: Monte Carlo (MC) sampling of trajectories
- Magenta: Gaussian of MC-sampled final positions

Concatenation of Uncertain Transformations



- Uncertainty of angular velocity about z-axis
- Blue: Monte Carlo (MC) sampling of trajectories
- Magenta: Gaussian of MC-sampled final positions
- Yellow: Propagated uncertainty in *exponential coordinates* (**banana distribution**)

Concatenation of Uncertain Transformations

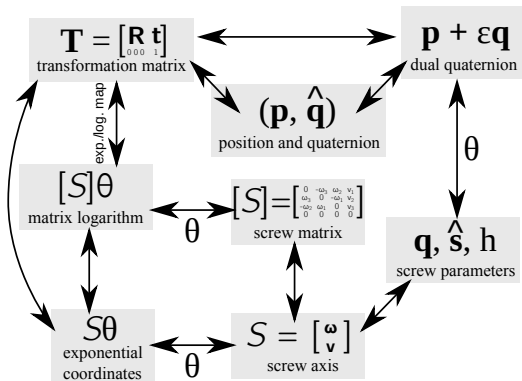


- Uncertainty of angular velocity about z-axis
- Blue: Monte Carlo (MC) sampling of trajectories
- Magenta: Gaussian of MC-sampled final positions
- Yellow: Propagated uncertainty in *exponential coordinates* (**banana distribution**)

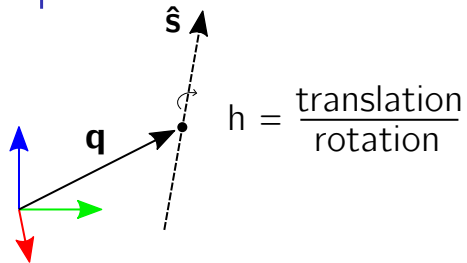
SE(3) (SE: special Euclidean group)

- group of all proper rigid transformations in 3D
- represented by transformation matrices

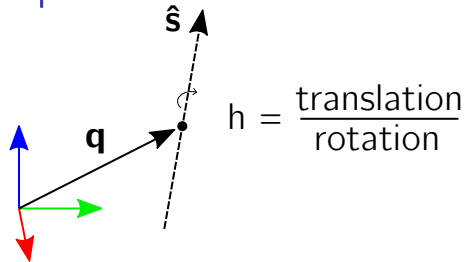
pytransform3d.transformations



Exponential Coordinates (Screw Theory)



Exponential Coordinates (Screw Theory)

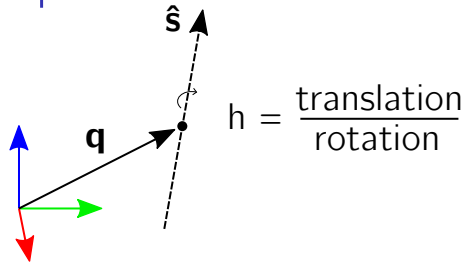


Angle: $\theta \in \mathbb{R}$

Screw axis:
$$\begin{bmatrix} \hat{s} \\ \mathbf{q} \times \hat{s} + h\hat{s} \end{bmatrix} = \mathcal{S} \in \mathbb{R}^6$$

Exp. coord.: $\mathcal{S}\theta \in \mathbb{R}^6$

Exponential Coordinates (Screw Theory)



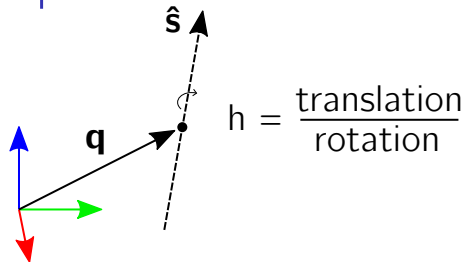
Exponential map:
 $\text{Exp}(\mathcal{S}\theta) = \mathbf{T} \in SE(3)$

Angle: $\theta \in \mathbb{R}$

Screw axis: $\begin{bmatrix} \hat{\mathbf{s}} \\ \mathbf{q} \times \hat{\mathbf{s}} + h\hat{\mathbf{s}} \end{bmatrix} = \mathcal{S} \in \mathbb{R}^6$

Exp. coord.: $\mathcal{S}\theta \in \mathbb{R}^6$

Exponential Coordinates (Screw Theory)



Exponential map:
 $Exp(\textcolor{red}{S}\textcolor{blue}{\theta}) = \textcolor{black}{T} \in SE(3)$

Angle: $\textcolor{blue}{\theta} \in \mathbb{R}$

Screw axis: $\begin{bmatrix} \hat{s} \\ \mathbf{q} \times \hat{s} + h\hat{s} \end{bmatrix} = \textcolor{red}{S} \in \mathbb{R}^6$ (Lynch and Park 2017)

Exp. coord.: $\textcolor{red}{S}\textcolor{blue}{\theta} \in \mathbb{R}^6$

```
import pytransform3d.transformations as pt
pt.transform_from_exponential_coordinates(
    Stheta)
```

Concatenation of Uncertain Transformations

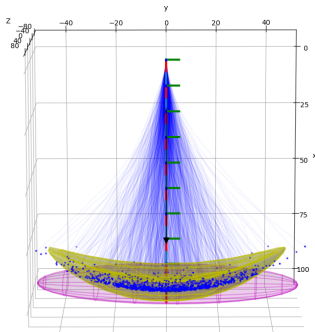
```
import pytransform3d.uncertainty as pu

# estimated mean pose (transformation matrix, 4x4)
T_estimated = ...
# covariance in exponential coordinates (6x6)
cov_estimated = ...

# mean velocity (transformation matrix, 4x4)
T_velocity = ...
# covariance in exponential coordinates (6x6)
cov_velocity = ...

T_estimated, cov_estimated = pu.concat_globally_uncertain_transforms(
    T_estimated, cov_estimated,
    T_velocity, cov_velocity)
```

Concatenation of Uncertain Transformations



The **banana distribution** is Gaussian in exponential coordinates:

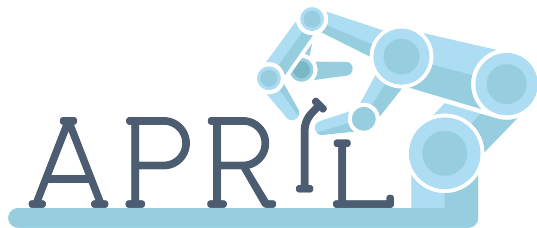
$$\mathbf{T} = \text{Exp}(\mathcal{S}\theta) \overline{\mathbf{T}}, \quad \text{with} \quad \mathcal{S}\theta \sim \mathcal{N}(0, \mathbf{\Sigma}_{6 \times 6})$$

(Long et al. 2012; Barfoot and Furgale 2014)

Probabilistic Robot Kinematics

Examples > 3D Visualizations > Probabilistic Product of Exponentials

Funding



This work was supported by the European Commission under the Horizon 2020 framework program for Research and Innovation (project acronym: APRIL, project number: 870142).

pytransform3d



- ▶ a library for rigid transformations in 3D
- ▶ organizes complex graphs of transformations
- ▶ coupling with matplotlib for quick visualization
- ▶ a matplotlib-like interface to Open3D's visualizer
- ▶ operations for representations of rotation and translation
- ▶ conversions between representations
- ▶ clear documentation of conventions

Backup

Application: Collision Detection

Collision Detection

- ▶ Broad phase collision detection with AABBs
- ▶ GJK algorithm (Gilbert et al. 1988)

Collision Detection

Library: <https://github.com/AlexanderFabisch/distance3d>

```
from pytransform3d.urdf import UrdfTransformManager
from distance3d import broad_phase

# Load graph of transformations
robot_tree = UrdfTransformManager()
with open(filename, "r") as f:
    robot_urdf = f.read()
robot_tree.load_urdf(robot_urdf)

# Define configuration of robot
for joint_name in ["joint%d" % i for i in range(1, 7)]:
    robot_tree.set_joint(joint_name, 0.7)

# Construct bounding volume hierarchy for broad phase
robot_bvh = broad_phase.BoundingVolumeHierarchy(
    robot_tree, "robot_arm")
robot_bvh.fill_tree_with_colliders(robot_tree)
```

Transformation Matrices

$$SE(3) = \left\{ \mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{4 \times 4} \mid \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}$$

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

How should we define a probability distribution in $SE(3)$?

Modeling Transformations

Mathematical notation: T_{BA} for transformation *from frame A to frame B*. In concatenation, read from right to left:

$$T_{CB} T_{BA} = T_{C\cancel{B}} T_{\cancel{B}A} = T_{CA}.$$

In code we should prefer the notation A2B for a transformation *from frame A to frame B*:

```
from pytransform3d.transformations import concat
A2B = ... # transformation from frame A to frame B
B2C = ... # transformation from frame B to frame C
A2C = concat(A2B, B2C)
```

Imitation Learning

Literature I

- 
- Barfoot, Timothy D. and Paul T. Furgale (2014). "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems". In: *IEEE Transactions on Robotics* 30.3, pp. 679–693. DOI: [10.1109/TR0.2014.2298059](https://doi.org/10.1109/TR0.2014.2298059).
- 
- Fabisch, Alexander, Manuela Uliano, Dennis Marschner, Melvin Laux, Johannes Brust, and Marco Controzzi (2022). "A Modular Approach to the Embodiment of Hand Motions from Human Demonstrations". In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pp. 801–808. DOI: [10.1109/Humanoids53995.2022.10000165](https://doi.org/10.1109/Humanoids53995.2022.10000165).
- 
- Gilbert, E.G., D.W. Johnson, and S.S. Keerthi (1988). "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE Journal on Robotics and Automation* 4.2, pp. 193–203. DOI: [10.1109/56.2083](https://doi.org/10.1109/56.2083).
- 
- Long, Andrew W., Kevin C. Wolfe, Michael Mashner, and Gregory S. Chirikjian (2012). "The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates". In: *Robotics: Science and Systems*.
- 
- Lynch, Kevin M. and Frank C. Park (2017). *Modern Robotics: Mechanics, Planning, and Control*. 1st. USA: Cambridge University Press. ISBN: 1107156300.
- 
- Sommer, Hannes, Igor Gilitschenski, Michael Bloesch, Stephan Weiss, Roland Siegwart, and Juan Nieto (2018). "Why and How to Avoid the Flipped Quaternion Multiplication". In: *Aerospace* 5.3. ISSN: 2226-4310. DOI: [10.3390/aerospace5030072](https://doi.org/10.3390/aerospace5030072).
- 
- Ude, Aleš, Bojan Nemec, Tadej Petrič, and Jun Morimoto (2014). "Orientation in Cartesian space dynamic movement primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Ed. by Ning Xi and William R. Hamel, pp. 2997–3004.