Formal Foundations of 3D Geometry to Model Robot Manipulators

Reynald Affeldt [1] Cyril Cohen [2]

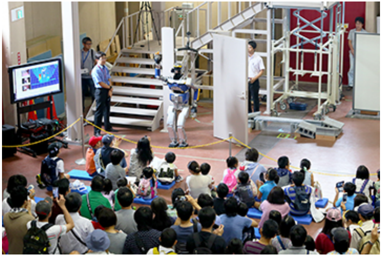
[1]AIST, Japan

[2]INRIA, France

January 16, 2017

Why Verify Robots?

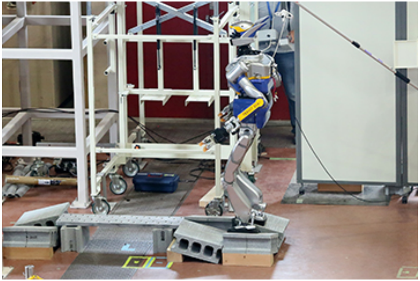
Last summer, I attended a demonstration of the **rescue** capabilities of the HRP-2 robot.



AIST open house in Tsukuba [2016-07-23]

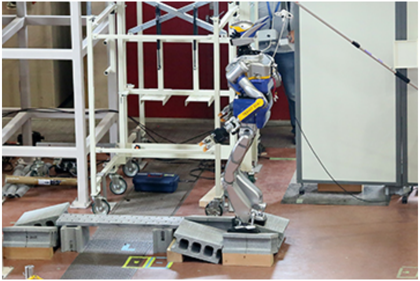
Why Verify Robots?

One of the task of the robot was to walk among debris.
In particular, it started walking a very narrow path.



Why Verify Robots?

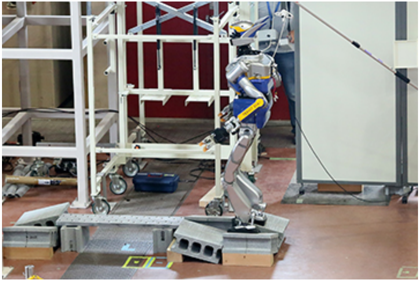
One of the task of the robot was to walk among debris.
In particular, it started walking a very narrow path.



It started walking like this...

Why Verify Robots?

One of the task of the robot was to walk among debris. In particular, it started walking a very narrow path.



It started walking like this...

. . . but fell after a few steps

Motivation and Contribution

- This is a need for safer robots
  - As of today, even a good robot can unexpectedly fail
    - HRP-2 was number 10 among 23 participants at the Finals of the 2015 DARPA Robotics Challenge
- Our work
  - (does not solve any issue with HRP-2 yet)
  - provides formal theories of
    - 3D geometry
    - *rigid body transformations*
  - for describing *robot manipulators*
  - in the Coq proof-assistant [INRIA, 1984∼]

What is a Robot Manipulator?

- E.g., SCARA (Selective Compliance Assembly Robot Arm)

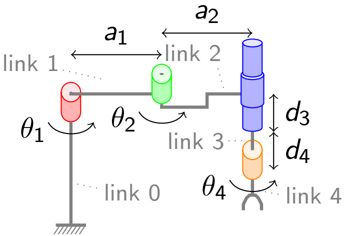
Mitsubishi RH-S series

What is a Robot Manipulator?

- E.g., SCARA (Selective Compliance Assembly Robot Arm)

Mitsubishi RH-S series

Schematic version

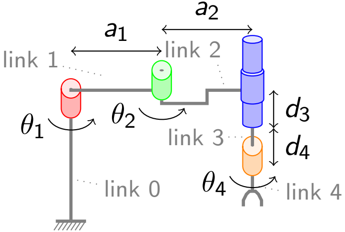


What is a Robot Manipulator?

- E.g., SCARA (Selective Compliance Assembly Robot Arm)

Mitsubishi RH-S series

Schematic version



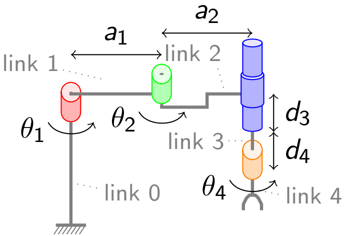
- Robot manipulator $\stackrel{def}{=}$ *Links* connected by *joints*
    - Revolute joint $\leftrightarrow$ rotation
    - Prismatic joint $\leftrightarrow$ translation

What is a Robot Manipulator?

- E.g., SCARA (Selective Compliance Assembly Robot Arm)

Mitsubishi RH-S series

Schematic version



- Robot manipulator $\stackrel{def}{=}$ *Links* connected by *joints*
  - Revolute joint ↔ rotation
  - Prismatic joint ↔ translation

NB: A humanoid robot can be seen as made of robot manipulators

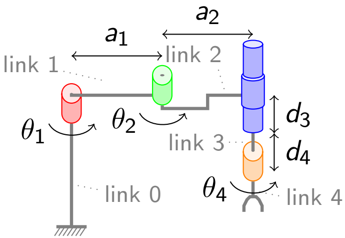
Why Rigid Body Transformations?

- To describe the relative position of links

Why Rigid Body Transformations?

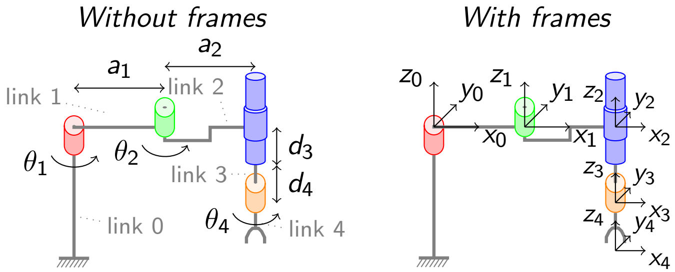
- To describe the relative position of links
- For this purpose, *frames* are attached to links:

Without frames



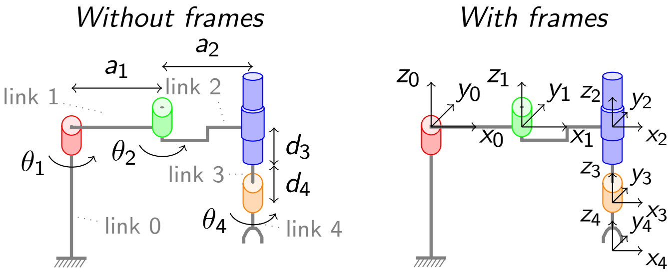
Why Rigid Body Transformations?

- To describe the relative position of links
- For this purpose, *frames* are attached to links:



Why Rigid Body Transformations?

- To describe the relative position of links
- For this purpose, *frames* are attached to links:



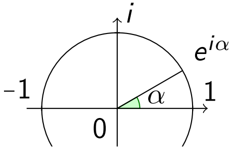
$\Rightarrow$ Approach: use the Mathematical Components library [INRIA/MSR, 2007∼]

- it contains the most extensive formalized theory on matrices and linear algebra

Outline

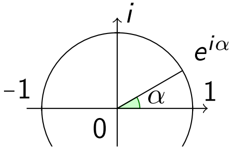
Formalization of Angles

Basic idea:
angle $\alpha \leftrightarrow$
unit complex number $e^{i\alpha}$



Formalization of Angles

Basic idea:
angle $\alpha \leftrightarrow$
unit complex number $e^{i\alpha}$

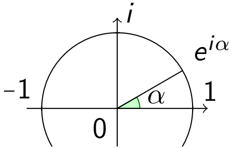


- Dependent record:

```
Record angle := Angle {
  expi : R[i] (* type of complex numbers *) ;
  _ : `| expi | == 1 }.
```

Formalization of Angles

Basic idea:
angle $\alpha \leftrightarrow$
unit complex number $e^{i\alpha}$



- Dependent record:

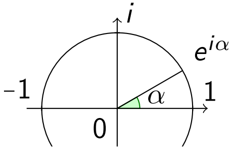
```
Record angle := Angle {
  expi : R[i] (* type of complex numbers *) ;
  _ : `| expi | == 1 }.
```

- The *argument* of a complex number defines an angle:

```
Definition arg (x : R[i]) : angle :=
  insubd angle0 (x / `| x |).
```

Formalization of Angles

Basic idea:
angle $\alpha \leftrightarrow$
unit complex number $e^{i\alpha}$



- Dependent record:

```
Record angle := Angle {
  expi : R[i] (* type of complex numbers *) ;
  _ : `| expi | == 1 }.
```

- The *argument* of a complex number defines an angle:

```
Definition arg (x : R[i]) : angle :=
  insubd angle0 (x / `| x |).
```

- Example: definition of π

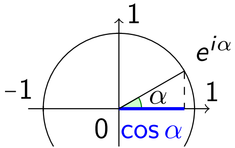
```
Definition pi := arg (-1).
```

Trigonometric Functions/Relations

Trigonometric functions
defined using complex numbers

Trigonometric Functions/Relations

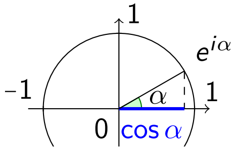
Trigonometric functions defined using complex numbers



- E.g., $\cos(\alpha) \overset{\textit{in Coq}}{\longrightarrow}$ `Re (expi` α`)`

Trigonometric Functions/Relations

Trigonometric functions
defined using complex numbers

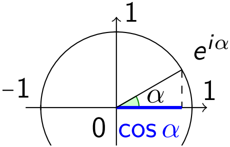


- E.g., $\cos(\alpha) \overset{in\ Coq}{\longrightarrow}$ `Re (expi` α`)`
- E.g., $\arcsin(x) \overset{def}{=} \arg\left(\sqrt{1-x^2}+xi\right)$

$\overset{in\ Coq}{\longrightarrow}$ `arg (Num.sqrt (1 - x^2) +i* x)`

Trigonometric Functions/Relations

Trigonometric functions
defined using complex numbers



- E.g., $\cos(\alpha) \overset{\textit{in Coq}}{\longrightarrow}$ `Re (expi` α`)`
- E.g., $\arcsin(x) \overset{\textit{def}}{=} \arg\left(\sqrt{1-x^2}+xi\right)$

  $\overset{\textit{in Coq}}{\longrightarrow}$ `arg (Num.sqrt (1 - x^2) +i* x)`

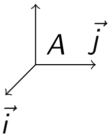
Standard trigonometric relations recovered easily:

- `Lemma acosK x : -1 <= x <= 1 -> cos (acos x) = x.`
- `Lemma sinD a b : sin (a+b) = sin a * cos b + cos a * sin b.`
- . . .

Formalization of the Cross-product

The cross-product is used to define oriented frames

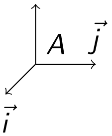
$$\vec{k} = \vec{i} \times \vec{j}$$



Formalization of the Cross-product

$$\vec{k} = \vec{i} \times \vec{j}$$

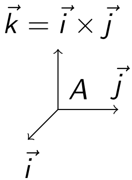
The cross-product is used to define oriented frames



- Let 'e_0, 'e_1, 'e_2 be the canonical vectors

Formalization of the Cross-product

The cross-product is used to define oriented frames



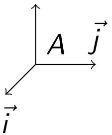
- Let 'e_0, 'e_1, 'e_2 be the canonical vectors
- Pencil-and-paper definition of the cross-product:

$$\vec{u} \times \vec{v} \stackrel{def}{=} \begin{vmatrix} 1 & 0 & 0 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_0} + \begin{vmatrix} 0 & 1 & 0 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_1} + \begin{vmatrix} 0 & 0 & 1 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_2}$$

Formalization of the Cross-product

The cross-product is used to define oriented frames

$$\vec{k} = \vec{i} \times \vec{j}$$



- Let 'e_0, 'e_1, 'e_2 be the canonical vectors
- Pencil-and-paper definition of the cross-product:

$$\vec{u} \times \vec{v} \stackrel{def}{=} \begin{vmatrix} 1 & 0 & 0 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_0} + \begin{vmatrix} 0 & 1 & 0 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_1} + \begin{vmatrix} 0 & 0 & 1 \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix} \text{'e\_2}$$

- Formal definition using Mathematical Components:

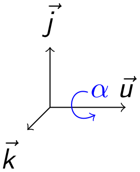
```
Definition crossmul u v :=
  \row_(k < 3) \det (col_mx3 'e_k u v).
```

Outline

Formal Definition of a Rotation

Rotation of angle α around $\vec{u} \stackrel{def}{=}$

A linear function f and a frame $\langle \frac{\vec{u}}{||\vec{u}||}, \vec{j}, \vec{k} \rangle$ such that:

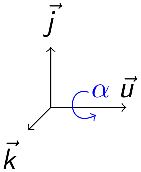


$$
\begin{aligned}
f(\vec{u}) &= \vec{u} \\
f(\vec{j}) &= \cos(\alpha)\vec{j} + \sin(\alpha)\vec{k} \\
f(\vec{k}) &= -\sin(\alpha)\vec{j} + \cos(\alpha)\vec{k}
\end{aligned}
$$

Formal Definition of a Rotation

Rotation of angle α around $\vec{u} \stackrel{def}{=}$

A linear function f and a frame $\langle \frac{\vec{u}}{||\vec{u}||}, \vec{j}, \vec{k} \rangle$ such that:



$$\begin{aligned} f(\vec{u}) &= \vec{u} \\ f(\vec{j}) &= \cos(\alpha)\vec{j} + \sin(\alpha)\vec{k} \\ f(\vec{k}) &= -\sin(\alpha)\vec{j} + \cos(\alpha)\vec{k} \end{aligned}$$

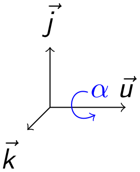
In practice, rotations are represented by *rotation matrices*

- Matrices M such that $M\,M^T = 1$ and $\det(M) = 1$
- Special orthogonal group 'SO[R]_3

Formal Definition of a Rotation

Rotation of angle α around $\vec{u} \stackrel{def}{=}$

A linear function f and a frame $\langle \frac{\vec{u}}{||\vec{u}||}, \vec{j}, \vec{k} \rangle$ such that:



$$\begin{array}{rcl} f(\vec{u}) & = & \vec{u} \\ f(\vec{j}) & = & \cos(\alpha)\vec{j} + \sin(\alpha)\vec{k} \\ f(\vec{k}) & = & -\sin(\alpha)\vec{j} + \cos(\alpha)\vec{k} \end{array}$$

In practice, rotations are represented by *rotation matrices*

- Matrices M such that $M\,M^T = 1$ and $\det(M) = 1$
- Special orthogonal group 'SO[R]_3

$\Rightarrow$ Equivalent to rotations defined above

- See the paper for formal proofs

Outline

Definition of a Rigid Body Transformation

A RBT preserves lengths and orientation

Definition of a Rigid Body Transformation

A RBT preserves lengths and orientation

1. f preserves lengths when
   - $||p - q|| = ||f(p) - f(q)||$ for all points p and q

Definition of a Rigid Body Transformation

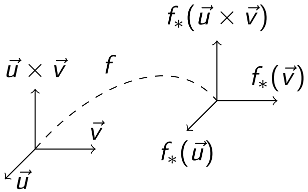
A RBT preserves lengths and orientation

1. f preserves lengths when
   - $||p - q|| = ||f(p) - f(q)||$ for all points p and q
2. f preserves orientation when it preserves the cross-product

Definition of a Rigid Body Transformation

A RBT preserves lengths and orientation

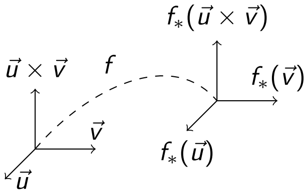
1. f preserves lengths when
   - $||p - q|| = ||f(p) - f(q)||$ for all points p and q
2. f preserves orientation when it preserves the cross-product
   - $f_*(\vec{u} \times \vec{v}) = f_*(\vec{u}) \times f_*(\vec{v})$ for all vectors $\vec{u}$ and $\vec{v}$



Definition of a Rigid Body Transformation

A RBT preserves lengths and orientation

1. f preserves lengths when
   - $||p - q|| = ||f(p) - f(q)||$ for all points p and q
2. f preserves orientation when it preserves the cross-product
   - $f_*(\vec{u} \times \vec{v}) = f_*(\vec{u}) \times f_*(\vec{v})$ for all vectors $\vec{u}$ and $\vec{v}$

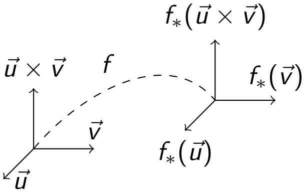


   - $f_*(\vec{w}) \stackrel{def}{=} f(q) - f(p)$ with $\vec{w} = q - p$

Definition of a Rigid Body Transformation

A RBT preserves lengths and orientation

1. f preserves lengths when
   - $||p - q|| = ||f(p) - f(q)||$ for all points p and q
2. f preserves orientation when it preserves the cross-product
   - $f_*(\vec{u} \times \vec{v}) = f_*(\vec{u}) \times f_*(\vec{v})$ for all vectors $\vec{u}$ and $\vec{v}$



   - $f_*(\vec{w}) \stackrel{def}{=} f(q) - f(p)$ with $\vec{w} = q - p$

$\Rightarrow$ Equivalent to *direct isometries*

- See the paper for formal proofs [O'Neill, 1966]

Matrix Representation for $\mathrm{R{\small BT}}$

In practice, $\mathrm{R{\small BT}}$ are given in *homogeneous representation*

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

- 4×4-matrices

Matrix Representation for RBT

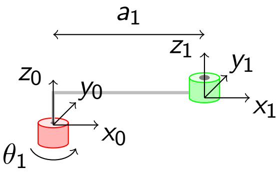
In practice, RBT are given in *homogeneous representation*

- 4×4-matrices
- $\begin{bmatrix} \mathtt{r} & \mathtt{0} \\ \mathtt{t} & \mathtt{1} \end{bmatrix}$ with $\mathtt{r}$ a rotation matrix and $\mathtt{t}$ a translation

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

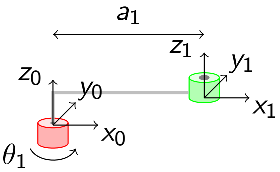
- 4×4-matrices
- $\begin{bmatrix} \mathtt{r} & 0 \\ \mathtt{t} & 1 \end{bmatrix}$ with $\mathtt{r}$ a rotation matrix and $\mathtt{t}$ a translation
- Example:



Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

- 4×4-matrices
- $\begin{bmatrix} \mathtt{r} & 0 \\ \mathtt{t} & 1 \end{bmatrix}$ with r a rotation matrix and t a translation
- Example:

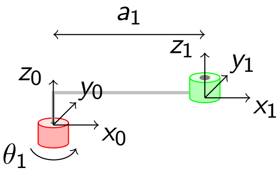


1) Rotation of θ_1 around z-axis
2) Translation $[a_1 \cos \theta_1 ; a_1 \sin \theta_1 ; 0]$

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

- 4×4-matrices
- $\begin{bmatrix} \mathtt{r} & \mathtt{0} \\ \mathtt{t} & \mathtt{1} \end{bmatrix}$ with `r` a rotation matrix and `t` a translation
- Example:



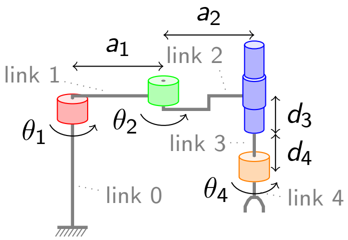
1) Rotation of θ_1 around z-axis
2) Translation $[a_1 \cos\theta_1; a_1 \sin\theta_1; 0]$

```
Definition A10 :=
  hom (Rz θ1) (row3 (a1 * cos θ1) (a1 * sin θ1) 0).
```

Outline

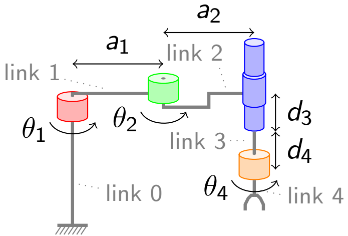
Forward Kinematics for the SCARA Robot Manipulator

Fwd Kin. = Position and orientation of the end-effector given the link and joint parameters



Forward Kinematics for the SCARA Robot Manipulator

Fwd Kin. = Position and orientation of the end-effector given the link and joint parameters

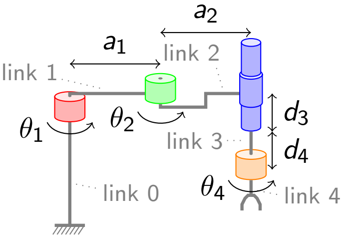


- Just perform the product of the successive RBT's:

```
Lemma hom_SCARA_forward :
  A43 * A32 * A21 * A10 = hom scara_rot scara_trans.
```

Forward Kinematics for the SCARA Robot Manipulator

Fwd Kin. = Position and orientation of the end-effector given the link and joint parameters



- Just perform the product of the successive RBT's:

```
Lemma hom_SCARA_forward :
  A43 * A32 * A21 * A10 = hom scara_rot scara_trans.
```

with

```
Definition scara_rot := Rz (θ1 + θ2+ θ4).
Definition scara_trans := row3
  (a2 * cos (θ2 + θ1) + a1 * cos θ1)
  (a2 * sin (θ2 + θ1) + a1 * sin θ1)
  (d4 + d3).
```

Outline

Denavit-Hartenberg Convention

- Convention for the relative positioning of frames

Denavit-Hartenberg Convention

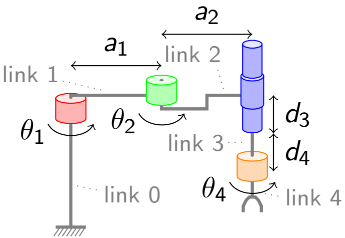
- Convention for the relative positioning of frames
  - Consecutive frames i and j are such that
    1) $(o_j, \vec{x_j})$ and $(o_i, \vec{z_i})$ are perpendicular
    2) and intersect

Denavit-Hartenberg Convention

- Convention for the relative positioning of frames
  - Consecutive frames i and j are such that
    1) $(o_j, \vec{x}_j)$ and $(o_i, \vec{z}_i)$ are perpendicular
    2) and intersect
  - The corresponding RBT can then be written $^hR_x(\alpha)^hT_x(a)^hT_z(d)^hR_z(\theta)$

Denavit-Hartenberg Convention

- Convention for the relative positioning of frames
  - Consecutive frames i and j are such that
    1) $(o_j, \vec{x_j})$ and $(o_i, \vec{z_i})$ are perpendicular
    2) and intersect
  - The corresponding RBT can then be written $^hR_x(\alpha)^hT_x(a)^hT_z(d)^hR_z(\theta)$
- Example: parameters for the SCARA robot manipulator



link	α_i	a_i	d_i	θ_i
	twist	length	offset	angle
1	0	a_1	0	θ_1
2	0	a_2	0	θ_2
3	0	0	d_3	0
4	0	0	d_4	θ_4

Outline

Exponential Coordinates of Rotations

- Alternative representation with less parameters
- $e^{\alpha S(w)}$ where $S(w) = \begin{bmatrix} 0 & w_z & -w_y \\ -w_z & 0 & w_x \\ w_y & -w_x & 0 \end{bmatrix}$

Exponential Coordinates of Rotations

- Alternative representation with less parameters
- $e^{\alpha S(w)}$ where $S(w) = \begin{bmatrix} 0 & w_z & -w_y \\ -w_z & 0 & w_x \\ w_y & -w_x & 0 \end{bmatrix}$
  - We could use a generic matrix exponential $e^M = 1 + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \cdots$

Exponential Coordinates of Rotations

- Alternative representation with less parameters
- $e^{\alpha S(w)}$ where $S(w) = \begin{bmatrix} 0 & w_z & -w_y \\ -w_z & 0 & w_x \\ w_y & -w_x & 0 \end{bmatrix}$
  - We could use a generic matrix exponential $e^M = 1 + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \cdots$
  - But when M is skew-symmetric, there is closed formula

$$e^{\alpha S(w)} \stackrel{def}{=} 1 + \sin(\alpha)S(w) + (1 - \cos(\alpha))S(w)^2$$

(Rodrigues' formula)

Exponential Coordinates of Rotations

- Alternative representation with less parameters
- $e^{\alpha S(w)}$ where $S(w) = \begin{bmatrix} 0 & w_z & -w_y \\ -w_z & 0 & w_x \\ w_y & -w_x & 0 \end{bmatrix}$
  - We could use a generic matrix exponential $e^M = 1 + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \cdots$
  - But when M is skew-symmetric, there is closed formula

$$e^{\alpha S(w)} \stackrel{def}{=} 1 + \sin(\alpha) S(w) + (1 - \cos(\alpha)) S(w)^2$$

(Rodrigues' formula)

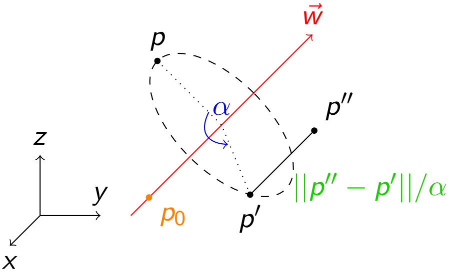
$\Rightarrow$ Equivalent to a rotation of angle α around $\vec{w}$

- See the paper for formal proofs

Outline

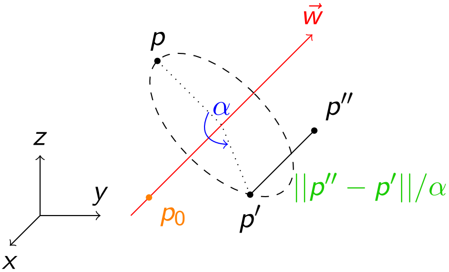
What is a Screw Motion?

- An axis (a point and a vector), an angle, a pitch



What is a Screw Motion?

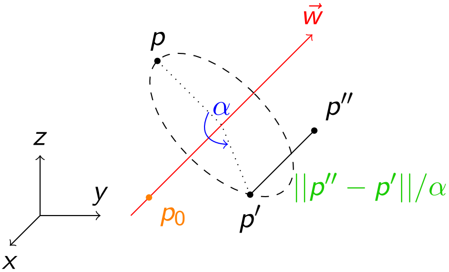
- An axis (a point and a vector), an angle, a pitch



- Translation and rotation axes are **parallel**
  - This was not required for homogeneous representations

What is a Screw Motion?

- An axis (a point and a vector), an angle, a pitch



- Translation and rotation axes are **parallel**
  - This was not required for homogeneous representations

$\Rightarrow$ Are screw motions RBT?

  - See the paper for Chasles' theorem ("the first theorem of robotics")

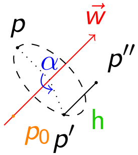
Represent Screw Motions with Exponentials of Twists

- To represent screw motions, we can use $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

Represent Screw Motions with Exponentials of Twists

- To represent screw motions, we can use $e^{\alpha\begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

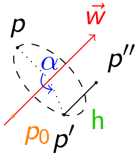
  With $v = -w \times p_0 + hw$ we recover the screw motion of the previous slide



Represent Screw Motions with Exponentials of Twists

- To represent screw motions, we can use $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

  With $v = -w \times p_0 + hw$ we recover the screw motion of the previous slide

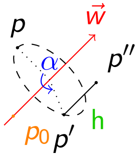


- The pair of vectors (v, w) is called a **twist**

Represent Screw Motions with Exponentials of Twists

- To represent screw motions, we can use $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

  With $v = -w \times p_0 + hw$ we recover the screw motion of the previous slide

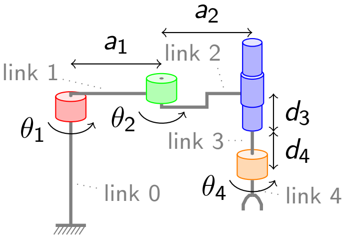


- The pair of vectors (v, w) is called a **twist**
- Luckily, there is a closed formula for $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

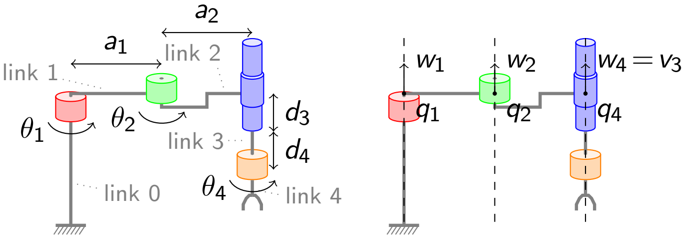
$$\begin{cases} \begin{bmatrix} I & 0 \\ \alpha\, v & 1 \end{bmatrix} & \text{if } w = 0 \\ \begin{bmatrix} e^{\alpha\, S(w)} & 0 \\ \frac{(w \times v)(1 - e^{\alpha\, S(w)}) + (\alpha\, v)(w^T w)}{||w||^2} & 1 \end{bmatrix} & \text{if } w \neq 0 \end{cases}$$

Outline

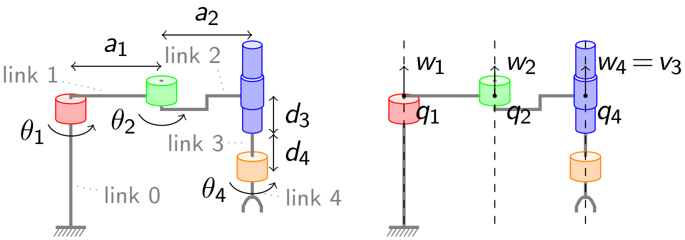
Fwd Kinematics for SCARA with Screw Motions



Fwd Kinematics for SCARA with Screw Motions

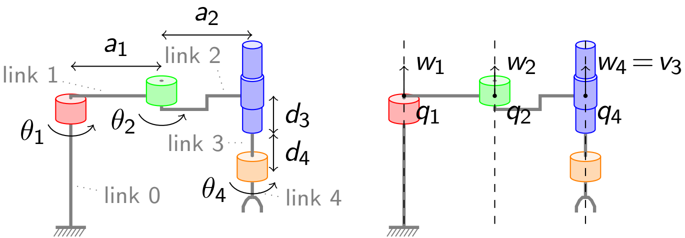


Fwd Kinematics for SCARA with Screw Motions



Position and orientation of the end-effector:

Fwd Kinematics for SCARA with Screw Motions

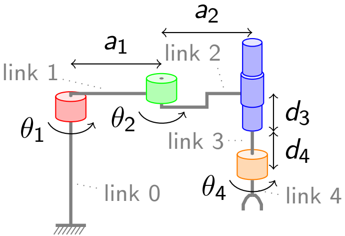


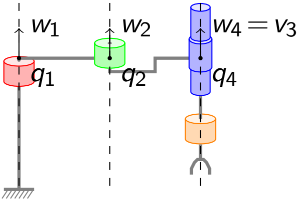
Position and orientation of the end-effector:

- When the joint parameters are fixed at 0:

```
Definition g0 := hom 1 (row3 (a1 + a2) 0 d4).
```

Fwd Kinematics for SCARA with Screw Motions





Position and orientation of the end-effector:

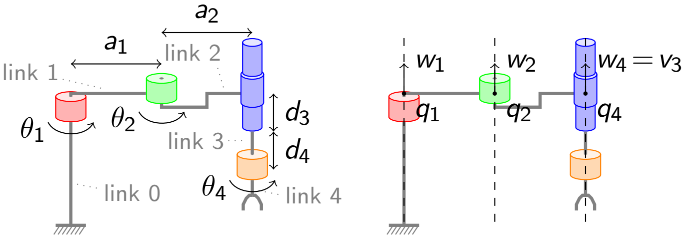
- When the joint parameters are fixed at 0:

  Definition g0 := hom 1 (row3 (a1 + a2) 0 d4).

- With joints with twists t_i and parameters d_i or θ_i

  Definition g := g0 ∗ \`e\$(θ_4, t4) ∗
  \`e\$(Rad.angle_of d3, t3) ∗ \`e\$(θ_2, t2) ∗ \`e\$(θ_1, t1).

Fwd Kinematics for SCARA with Screw Motions



Position and orientation of the end-effector:

- When the joint parameters are fixed at 0:

  `Definition g0 := hom 1 (row3 (a1 + a2) 0 d4).`

- With joints with twists t_i and parameters d_i or θ_i

  ```
  Definition g := g0 * `e$(θ4, t4) *
    `e$(Rad.angle_of d3, t3) * `e$(θ2, t2) * `e$(θ1, t1).
  ```

- Revolute: $t_i = (-w_i \times q_i, w_i)$; prismatic: $t_3 = (v_3, 0)$

Outline

Library Overview

Basic elements of 3D Geometry	
angles	slides 8
trigonometric functions	slide 9
cross-product	slide 10
lines	see the paper
3D Rotations	
using matrices ('SO[R]_3)	slide 12
using exponential coordinates ('so[R]_3)	slide 21
using quaternions	see the paper
using Euler angles	work in progress
Rigid Body Transformations	
using matrices ('SE3[R])	slide 15
using exp. coor. (screw motions) ('se3[R])	slide 24
using dual quaternions	work in progress
using the Denavit-Hartenberg convention	slide 19

$\Rightarrow$ Covers the introductory material of textbooks on robotics

$\Rightarrow$ Enough for forward kinematics of robot manipulators

Related Work

Mostly in 2D

- Collision avoidance algorithm for a vehicle moving in a plane in Isabelle [Walter et al., SAFECOMP 2010]
- Gathering algorithms for autonomous robots and impossibility results
  [Auger et al., SSS 2013] [Courtieu et al., IPL 2015, DISC 2016]
- Event-based programming framework in Coq
  [Anand et al., ITP 2015]
- Planar manipulators in HOL-Light
  [Farooq et al., ICFEM 2013]
- (in 3D) Conformal geometric algebra in HOL-Light
  [Ma et al., Advances in Applied Clifford Algebras 2016]

Future Work

- Various technical improvements
  - Better theory of lines, dependent types to link coordinates with frames
- Instantiate real closed field using classical reals
  - we have been using discrete real closed fields
    - because in MATHEMATICAL COMPONENTS every algebraic structures must have a decidable Leibniz equality
  - yet, equality for classical reals can be assumed decidable
- Application to concrete software
  - by showing preservation of invariants
  - we could use CoRN ideas to bridge with a computable alternative [Kaliszyk and O'Connor, CoRR 2008] [Krebbers and Spitters, LMCS 2011]
  - using CoqEAL for program refinements [Dénès et al., ITP 2012] [Cohen et al., CPP 2013]
- Extension with velocity