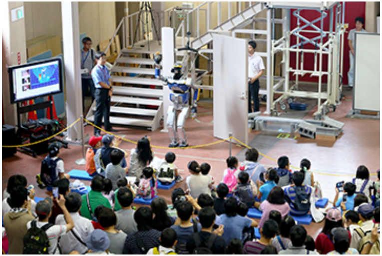
Formalized 3D Geometry for Robot Manipulators

EUTypes COST Meeting, Nijmegen, January 23, 2018

Reynald Affeldt *(AIST, Japan)* Cyril Cohen *(Inria, France)*

Why Verify Robots?

In summer 2016, Reynald attended a demonstration of the **rescue** capabilities of the HRP-2 robot.

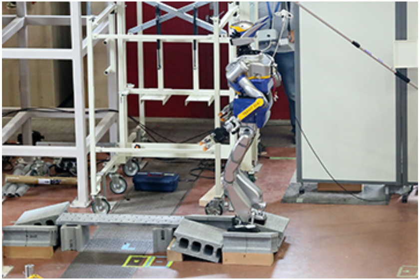


AIST open house in Tsukuba [2016-07-23]

Can you find Reynald?

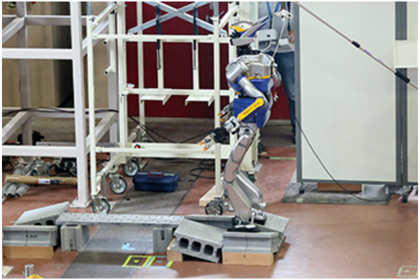
Why Verify Robots?

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



Why Verify Robots?

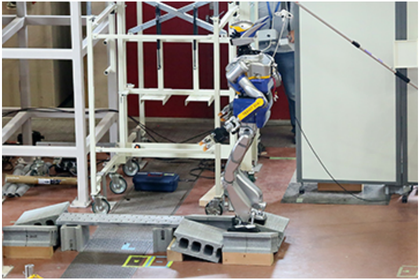
One of the task of the robot was to walk among debris.
In particular, it started walking a very narrow path.



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

- There 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~]
  - using the Mathematical Components library
  - `https://github.com/affeldt-aist/coq-robot`

What is a Robot Manipulator?

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

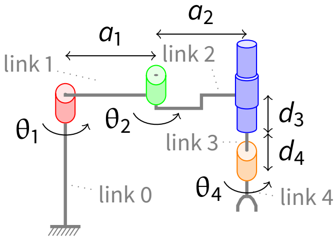
Mitsubishi RH-S series

What is a Robot Manipulator?

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

Mitsubishi RH-S series

Schematic version

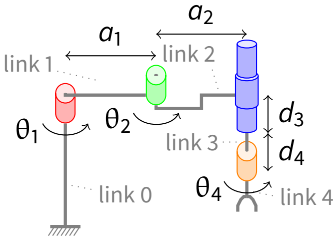


What is a Robot Manipulator?

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

Mitsubishi RH-S series

Schematic version



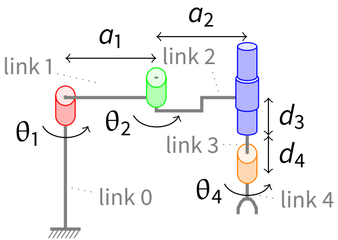
- Robot manipulator $\stackrel{def}{=}$ *Links* connected by *joints*
  - Revolute joint $\leftrightarrow$ rotation
  - Prismatic joint $\leftrightarrow$ translation

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 $\leftrightarrow$ rotation
  - Prismatic joint $\leftrightarrow$ translation

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

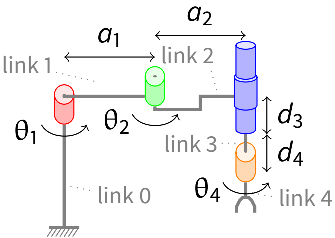
Why Rigid Body Transformations?

- 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:

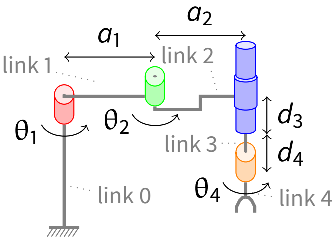
Without frames



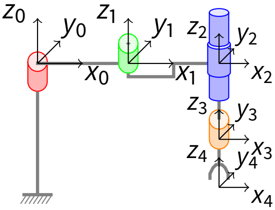
Why Rigid Body Transformations?

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

Without frames

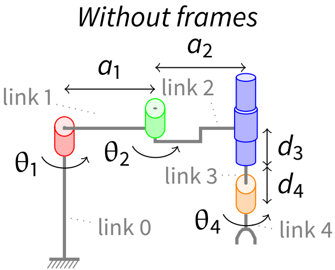


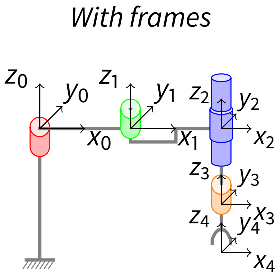
With frames



Why Rigid Body Transformations?

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





⇒ Approach: use the MATHEMATICAL COMPONENTS library [INRIA/MSR, 2007~]

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

Outline

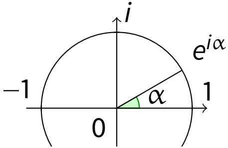
1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
   - 3D Rotations
   - Rigid Body Transformations
   - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
   - Exponential of skew-symmetric matrices
   - Screw Motions
   - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

1

Basic Elements of 3D Geometry

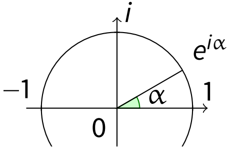
Formalization of Angles (CPP2017)

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



Formalization of Angles (CPP2017)

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

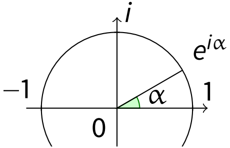


- Dependent record:

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

Formalization of Angles (CPP2017)

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



- Dependent record:

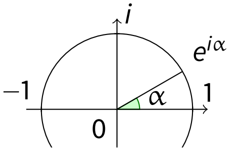
```
Record angle := Angle {
  expi : R[i] (* think of it as the type of complex numbers *);
  _ : `| expi | == 1 }.
```

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

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

Formalization of Angles (CPP2017)

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



- Dependent record:

```
Record angle := Angle {
  expi : R[i] (* think of it as the 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

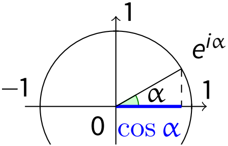
```
Definition pi := arg (−1).
```

Trigonometric Functions/Relations (CPP2017)

Trigonometric functions
defined using complex numbers

Trigonometric Functions/Relations (CPP2017)

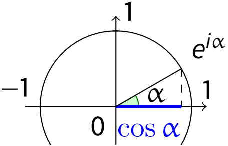
Trigonometric functions
defined using complex numbers



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

Trigonometric Functions/Relations (CPP2017)

Trigonometric functions
defined using complex numbers

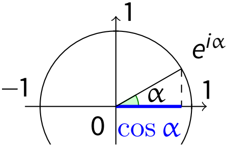


- E.g., $\cos(\alpha) \stackrel{\textit{in Coq}}{\longrightarrow}$ `Re (expi` α`)`
- E.g., $\arcsin(x) \stackrel{\textit{def}}{=} \arg\left(\sqrt{1-x^2}+xi\right)$

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

Trigonometric Functions/Relations (CPP2017)

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)`

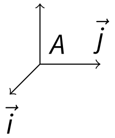
Standard trigonometric relations recovered easily:

- `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 (CPP2017)

The cross-product is used
to define oriented frames

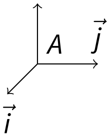
$$\vec{k} = \vec{i} \times \vec{j}$$



Formalization of the Cross-product (CPP2017)

The cross-product is used
to define oriented frames

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

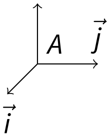


- Let 'e_0, 'e_1, 'e_2 be the canonical vectors

Formalization of the Cross-product (CPP2017)

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

The cross-product is used
to define oriented frames



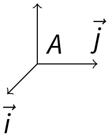
- Let ’e_0, ’e_1, ’e_2 be the canonical vectors
- Pencil-and-paper definition of the cross-product:

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

Formalization of the Cross-product (CPP2017)

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:

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

- Formal definition using Mathematical Components:

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

Formalization of the Cross-product (WIP)

First formalize Exterior Algebra
`https://github.com/CohenCyril/Clifford`
(joint work with Maxime Bombar)

- If $(e_0, \ldots, e_{n-1})$ is a basis, a basis of the exterior algebra is

$$\left(e_{i_0} \wedge \ldots \wedge e_{i_k}\right)_{i_0 < \ldots < i_k}.$$

- We embed it in $\mathbb{R}^{2^n}$

Formalization of the Cross-product (WIP)

First formalize Exterior Algebra
`https://github.com/CohenCyril/Clifford`
(joint work with Maxime Bombar)

- If $(e_0, \ldots, e_{n-1})$ is a basis, a basis of the exterior algebra is

$$\left(e_{i_0} \wedge \ldots \wedge e_{i_k}\right)_{i_0 < \ldots < i_k}.$$

- We embed it in $\mathbb{R}^{2^n}$
- Cross product : $u \times v = \varphi(u \wedge v)$ where $\varphi : e_i \wedge e_j \mapsto \pm e_{2-(i+j)}$

Formalization of the Cross-product (WIP)

First formalize Exterior Algebra
`https://github.com/CohenCyril/Clifford`
(joint work with Maxime Bombar)

- If $(e_0, \ldots, e_{n-1})$ is a basis, a basis of the exterior algebra is

$$\left(e_{i_0} \wedge \ldots \wedge e_{i_k}\right)_{i_0 < \ldots < i_k}.$$

- We embed it in $\mathbb{R}^{2^n}$
- Cross product : $u \times v = \varphi(u \wedge v)$ where $\varphi : e_i \wedge e_j \mapsto \pm e_{2-(i+j)}$
- Looks overkill but . . .
  - Factors bi-linearity theorems and triple product.
  - Generalizes cross product to higher dimensions.
  - Generalizations of exterior algebras are Clifford algebras, also very useful in robotics [Ma et al., 2016]
  - May help the study of differentials.

2

Robot Manipulators with Matri-ces

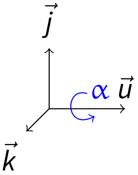
Outline

1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
    - 3D Rotations
    - Rigid Body Transformations
    - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
    - Exponential of skew-symmetric matrices
    - Screw Motions
    - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

Formal Definition of a Rotation (CPP2017)

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:

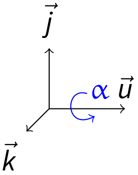


$$
\begin{aligned}
f(\vec{u}) &= \vec{u} \\
f(\vec{j}) &= \cos(\alpha)\vec{j} + \sin(\alpha)\vec{k} \\
f(\vec{k}) &= -\sin(\alpha)\vec{j} + \cos(\alpha)\vec{k}
\end{aligned}
$$

Formal Definition of a Rotation (CPP2017)

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:



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

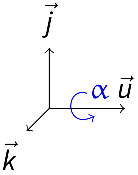
In practice, rotations are represented by *rotation matrices*

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

Formal Definition of a Rotation (CPP2017)

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

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



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

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

Spectral Theorem (WIP)

https://github.com/CohenCyril/spectral

A shorter option to establish the equivalence.

- Orthogonal matrices ($MM^T = 1$) are normal ($MM^T = M^T M$)
- Normal matrices are diagonalizable in an orthonormal basis.

```
Theorem normal_spectralP {n} {A : 'M[C]_n}
  (P := spectralmx A) (sp := spectral_diag A) :
  reflect (A = invmx P *m diag_mx sp *m P) (A \is normalmx).

Lemma spectral_unitary n (A : 'M[C]_n) :
  spectralmx A \is unitary.
```

Spectral Theorem (WIP)

`https://github.com/CohenCyril/spectral`

A shorter option to establish the equivalence.

- Orthogonal matrices ($MM^T = 1$) are normal ($MM^T = M^TM$)
- Normal matrices are diagonalizable in an orthonormal basis.

```
Theorem normal_spectralP {n} {A : 'M[C]_n}
  (P := spectralmx A) (sp := spectral_diag A) :
  reflect (A = invmx P *m diag_mx sp *m P) (A \is normalmx).

Lemma spectral_unitary n (A : 'M[C]_n) :
  spectralmx A \is unitary.
```

- 3-dimensional rotations eigenvalues are 1, $e^{i\alpha}$, $e^{-i\alpha}$ *(WIP)*

Spectral Theorem (WIP)

`https://github.com/CohenCyril/spectral`

A shorter option to establish the equivalence.

- Orthogonal matrices ($MM^T = 1$) are normal ($MM^T = M^T M$)
- Normal matrices are diagonalizable in an orthonormal basis.

```
Theorem normal_spectralP {n} {A : 'M[C]_n}
  (P := spectralmx A) (sp := spectral_diag A) :
  reflect (A = invmx P *m diag_mx sp *m P) (A \is normalmx).

Lemma spectral_unitary n (A : 'M[C]_n) :
  spectralmx A \is unitary.
```

- 3-dimensional rotations eigenvalues are 1, $e^{i\alpha}$, $e^{-i\alpha}$ *(WIP)*
- The axis is a non zero vector from the eigenspace $\ker(M - 1)$.
- The angle is α or its negation. *(WIP)*

Spectral Theorem (WIP)

`https://github.com/CohenCyril/spectral`

A shorter option to establish the equivalence.

- Orthogonal matrices ($MM^T = 1$) are normal ($MM^T = M^TM$)
- Normal matrices are diagonalizable in an orthonormal basis.

```
Theorem normal_spectralP {n} {A : 'M[C]_n}
  (P := spectralmx A) (sp := spectral_diag A) :
  reflect (A = invmx P *m diag_mx sp *m P) (A \is normalmx).

Lemma spectral_unitary n (A : 'M[C]_n) :
  spectralmx A \is unitary.
```

- 3-dimensional rotations eigenvalues are $1, e^{i\alpha}, e^{-i\alpha}$ *(WIP)*
- The axis is a non zero vector from the eigenspace $\ker(M - 1)$.
- The angle is α or its negation. *(WIP)*

NB: Spectral theorem useful for singular value decomposition.

Outline

1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
    - 3D Rotations
    - Rigid Body Transformations
    - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
    - Exponential of skew-symmetric matrices
    - Screw Motions
    - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

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

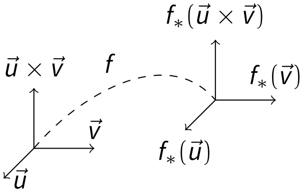
A Rbt preserves lengths and orientation

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

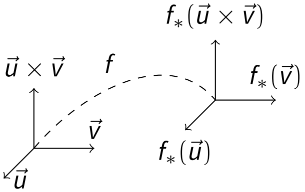
1. f preserves lengths when
   - $\|p - q\| = \|f(p) - f(q)\|$ for all points p and q
2. f preserves orientation when it preserves the cross-product
   - $f_*(\vec{u} \times \vec{v}) = f_*(\vec{u}) \times f_*(\vec{v})$ for all vectors $\vec{u}$ and $\vec{v}$



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 $\vec{v}$

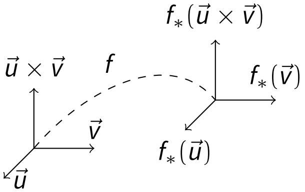


   - $f_*(\vec{w}) \stackrel{def}{=} f(q) - f(p)$ with $\vec{w} = q - p$

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 $\vec{w} = q - p$

⇒ Equivalent to *direct isometries* [O'Neill, 1966]

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

- 4×4-matrices

Matrix Representation for RBT

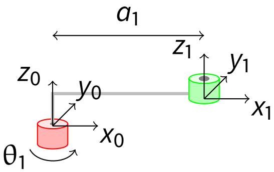
In practice, RBT are given in *homogeneous representation*

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

Matrix Representation for RBT

In practice, RBT are given in *homogeneous representation*

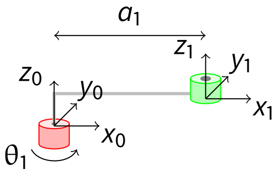
- 4×4-matrices
- $\begin{bmatrix} r & 0 \\ t & 1 \end{bmatrix}$ with r a rotation matrix and t a translation
- Example:



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 $\mathtt{r}$ a rotation matrix and $\mathtt{t}$ a translation
- Example:



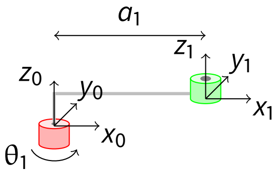
1) Rotation of θ_1 around z-axis

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



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

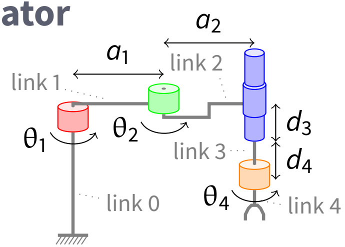
```
Definition A10 :=
  hom (Rz θ1) (row3 (a1 * cos θ1) (a1 * sin θ1) 0).
```

Outline

1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
    - 3D Rotations
    - Rigid Body Transformations
    - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
    - Exponential of skew-symmetric matrices
    - Screw Motions
    - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

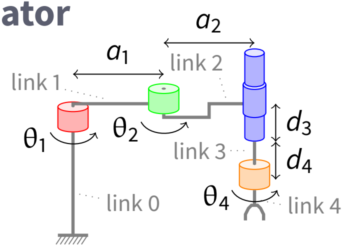
Forward Kinematics for the SCARA Robot Manipulator

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

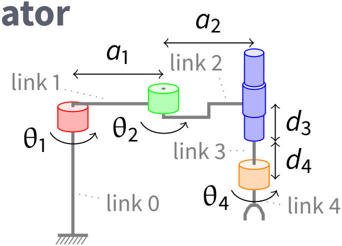


Just perform the product of the successive RBT's:

```
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).
```

3

Robot Manipulators with Exponential Coordinates

Outline

1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
    - 3D Rotations
    - Rigid Body Transformations
    - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
    - Exponential of skew-symmetric matrices
    - Screw Motions
    - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

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

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

(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)

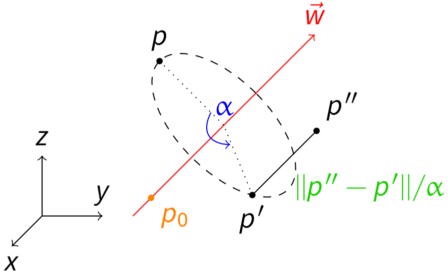
$\Rightarrow$ Equivalent to a rotation of angle α around $\vec{w}$

Outline

1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
   - 3D Rotations
   - Rigid Body Transformations
   - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
   - Exponential of skew-symmetric matrices
   - Screw Motions
   - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

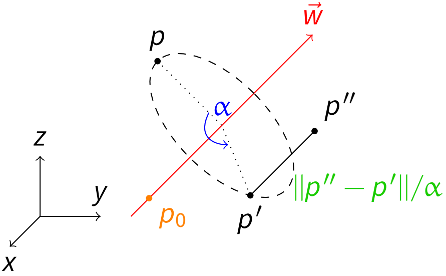
What is a Screw Motion?

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



What is a Screw Motion?

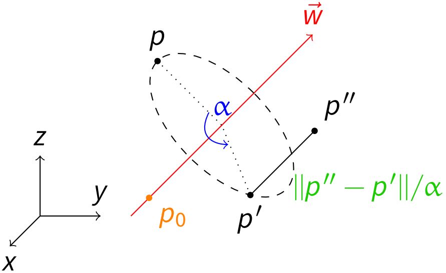
- An axis (a point and a vector), an angle, a pitch



- 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

⇒ Are screw motions RBT?
  - Yes: Chasles' theorem ("the first theorem of robotics")

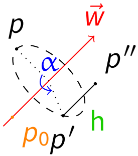
Represent Screw Motions with Exponentials of Twists

- 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 $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

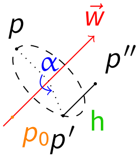
With $v = -w \times p_0 + hw$ we recover the screw motion of the previous slide



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

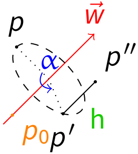


- The pair of vectors (v, w) is called a **twist**

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



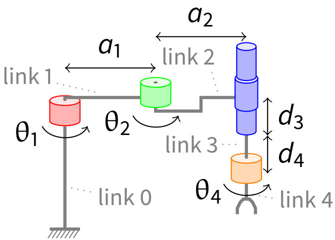
- The pair of vectors (v, w) is called a **twist**
- Luckily, there is a closed formula for $e^{\alpha \begin{bmatrix} S(w) & 0 \\ v & 0 \end{bmatrix}}$

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

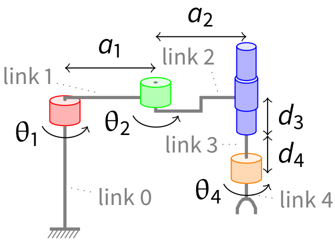
Outline

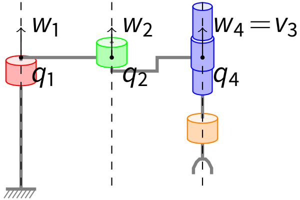
1. Basic Elements of 3D Geometry
2. Robot Manipulators with Matrices
   - 3D Rotations
   - Rigid Body Transformations
   - Example: SCARA
3. Robot Manipulators with Exponential Coordinates
   - Exponential of skew-symmetric matrices
   - Screw Motions
   - Example: SCARA
4. Velocity in Robot Manipulators (WIP)
5. Conclusion

Fwd Kinematics for SCARA with Screw Motions

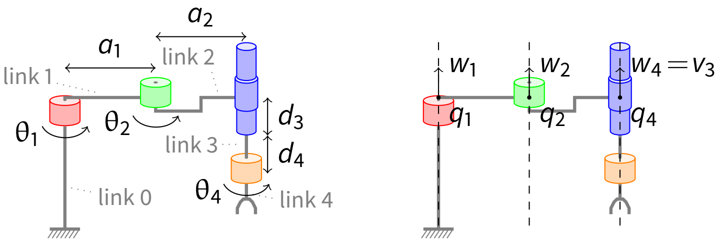


Fwd Kinematics for SCARA with Screw Motions



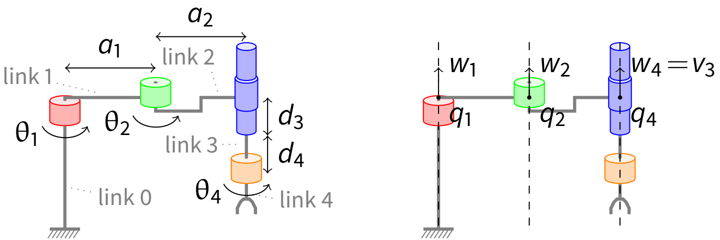


Fwd Kinematics for SCARA with Screw Motions



Position and orientation of the end-effector:

Fwd Kinematics for SCARA with Screw Motions

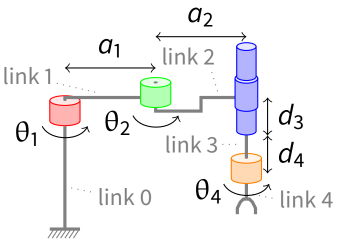


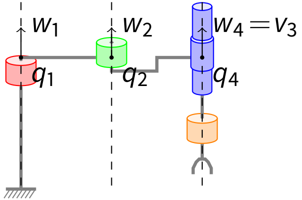
Position and orientation of the end-effector:

- When the joint parameters are fixed at 0:

  `Definition g0 := hom1 (row3 (a1 + a2) 0 d4).`

Fwd Kinematics for SCARA with Screw Motions





Position and orientation of the end-effector:

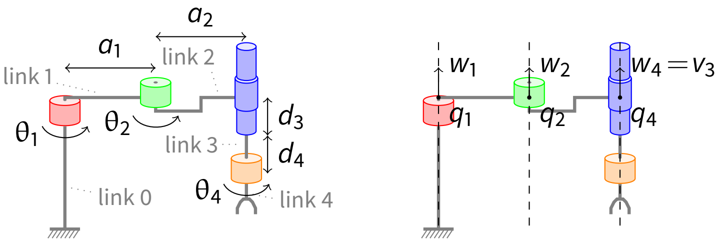
- When the joint parameters are fixed at 0:

```
Definition g0 := hom1 (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).
```

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$($\theta_4$, t4) ∗
  ‘e$(Rad.angle_of d3, t3) ∗ ‘e$($\theta_2$, t2) ∗ ‘e$(θ_1, t1).

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

4

Velocity in Robot Manipulators (WIP)

Velocity and Jacobian of a Robot

- If $q = (q_0, \ldots, q_k)$ are the joint parameters, and $P_n = (X_n, Y_n, Z_n, \omega_{X_n}, \omega_{Y_n}, \omega_{Z_n})$ the position and orientation of the final frame. We should establish a relation $\dot{P_n} = J(q) \cdot \dot{q}$.
- J is called the *Jacobian* of the robot.
- We want to certify a closed algebraic expression for J for our chains. *E.g.*

$$J = S(\theta_0) + A_0 S(\theta_1) A_0^{-1} + A_0 A_1 S(\theta_1) A_1^{-1} A_0^{-1} + \ldots$$

- We must rely on an analysis library, compatible with mathematical components.

Analysis with Mathematical Components (WIP)

- The Coquelicot Library [Boldo et al.] is a good start.
- It does not contain matrices and thus no Jacobian.
- “Halfway” between constructive and classical

We are in the process of re-implementing a *classical* analysis based on Coquelicot’s ideas, but compatible with Mathematical Components.

- `https://github.com/math-comp/analysis`

  (Reynald Affeldt, C.C., Damien Rouhling)

- ```
  Lemma diff_locally (x : V) (f : V → W) : differentiable x f →
    f \o shift x = cst (f x) + 'd_x f +o_(0 : V) id.
  ```

  ```
  Definition jacobian n m (f : 'rV_n → 'rV_m) p := lin1_mx ('d_p f).
  ```
  ```

5

Conclusion

Libraries Overview (Merge in progress)

- `https://github.com/affeldt-aist/coq-robot`: homogeneous coordinates, cross products, angles, rotations, RBT, Denavit-Hartenberg, screw motions, quaternions, exponential of skew-matrices, *future work: velocity kinematics*
- `https://github.com/CohenCyril/spectral`: generalized eigenspaces, Gram-Schmidt, cotrigonalization, codiagonalization, spectral theorem for normal, unitary and hermitian matrices, *future work: systematic study of quadratic forms*
- `https://github.com/CohenCyril/Clifford`: exterior algebra (WIP), *future work: Clifford algebras*
- `https://github.com/math-comp/analysis`: filter based **classical** topology, continuity, Landau notations, differential, Jacobian, *future work: integration*

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]
- Planar manipulators in HOL-Light [Farooq et al., ICFEM 2013]
- Event-based programming framework in Coq [Anand et al., ITP 2015]
- (in 3D) Conformal geometric algebra in HOL-Light [Ma et al., Advances in Applied Clifford Algebras 2016]

Applications?

- 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]

6

Additional Slides

Denavit-Hartenberg Convention

- Convention for the relative positioning of frames

Denavit-Hartenberg Convention

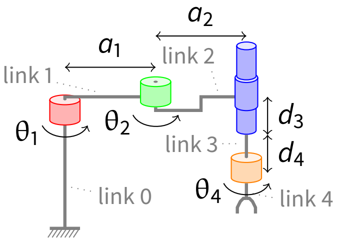
- Convention for the relative positioning of frames
  - Consecutive frames *i* and *j* are such that
    1) $(o_j, \vec{x_j})$ and $(o_i, \vec{z_i})$ are perpendicular
    2) and intersect

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

Angle-axis Representation of a Rotation

- A even more direct computation method for the exponential coordinates:
  - $a \stackrel{def}{=} \arccos\left(\frac{\mathrm{tr}(M)-1}{2}\right) \stackrel{in\ Coq}{\longrightarrow}$ `Aa.angle`
  - $\vec{w} \stackrel{def}{=} \mathrm{unskew} \frac{1}{2\sin(a)}(M - M^T) \stackrel{in\ Coq}{\longrightarrow}$ `Aa.vaxis`

  (with special cases when the angle is 0 or π)

- Correctness:

```
Lemma angle_axis_eskew M : M \is 'SO[R]_3 →
  M = `e^(Aa.angle M, normalize (Aa.vaxis M)).
```

Exponential of Twists are RBT

Notation: $e^{a\,t} \overset{in\ Coq}{\longrightarrow}$ `` `e$(a, t) ``

$\longrightarrow$ `'e$(a, t)` represents some RBT

- the shape of the matrix corresponds to some homogeneous representation

$\longleftarrow$ Any RBT can be represented by some `'e$(a, t)`:

```
Lemma etwist_is_onto_SE f : f \is 'SE3[R] →
  exists t a, f = `e$(a, t).
```

- constructive proof:
  a) from f, extract rotation r and translation p
  b) a and w are the exponential coordinates of r
  c) $v = ||w||^2 p \left(\frac{1}{a} - \frac{1}{2} S(w) + \left(\frac{1}{a} - \frac{1}{2}\cot\left(\frac{a}{2}\right)\right) S(w)^2\right)$