

Controlling Robot Manipulators

Alexis Maldonado, Andreas Fedrizzi

Lehrstuhl für Informatik IX
Technische Universität München

Wintersemester 2008

Outline of the Course

- Introduction
 - Robot-Components
 - Manipulators
 - Configuration Space
- Software
 - Player
 - Gazebo
- Position Analysis
 - Vectors in Space
 - Representation of Rigid Bodies
 - Representation of Frame Transformations
 - Roboop
- Forward and Inverse Kinematics
- Modeling of Manipulators
 - DH Parameters
 - GLrooop
- Further topics
 - Trajectory Planning
 - Modern Control Methods

Outline

- 1 Representation of Frame Transformations
- 2 Introduction to Roboop

Outline

- 1 Representation of Frame Transformations
- 2 Introduction to Roboop

Representation of Frame Transformations

- A transformation is defined as making a **movement in space**.
- A transformation can be
 - a pure translation
 - a pure rotation about an axis
 - a combination of translations and rotations
- We use **homogenous transformation matrices** for representing transformations, which are 4×4 matrices that are like the matrices that we used for frame representation.
- Square matrices have useful properties like
 - multiplication in any order is possible
 - calculation of the inverse matrix is easier (an example will follow)

Naming conventions

- For general homogenous transformation matrices
 - Homogenous Transformation (HT) of frame R relative to the reference frame O (world origin): ${}^O T_R$
 - Special cases of a HT
 - Pure translation: $Trans(d_x, d_y, d_z)$
 - Pure rotation around one axis: $Rot(x, \theta)$, $Rot(y, \theta)$, $Rot(z, \theta)$
 - Rotation matrix defined through the angles roll, pitch, yaw: $RPY(\phi_a, \phi_o, \phi_n)$
- For coordinate frames in our practical course
 - World origin: O
 - Base of B21: $B21$
 - Right shoulder of B21: RS
 - Right gripper of B21: RG
 - Right laser range finder of B21: RL
 - Frames on the left side are named LS , LG , LL accordingly

Representation of a pure translation relative to a reference frame

- Only the position of the origin of the body frame changes, while the directional unit vectors point into the same direction.
- The translation matrix that translates the body frame's origin is given relative to the reference frame.
- In matrix form, the new body frame representation can be found by **premultiplying** the body frame with the translation matrix.
- Computation: Current body frame: F_{cur}

Translation matrix: $Trans(d_x, d_y, d_z)$

$$F_{new} = Trans(d_x, d_y, d_z) \times F_{cur}$$

$$F_{new} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x + d_x \\ n_y & o_y & a_y & p_y + d_y \\ n_z & o_z & a_z & p_z + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Representation of a pure rotation about an axis of a reference frame

- Matrix for rotating a point by θ around the x-axis.

$$Rot(x, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrix for rotating a point by θ around the y-axis.

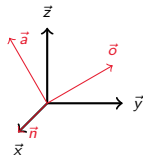
$$Rot(y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrix for rotating a point by θ around the z-axis.

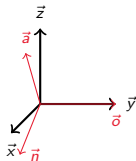
$$Rot(z, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Visualization of rotations

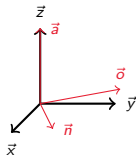
$$Rot(x, 30^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & -0.5 & 0 \\ 0 & 0.5 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rot(y, 30^\circ) = \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rot(z, 30^\circ) = \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Representation of a combined transformation relative to a reference frame

- Consists of a number of **successive translations and rotations** about the fixed reference frame or the moving frame's axes.
- Any transformation can be resolved into a set of translations and rotations in a particular order.
- The **multiplication order** of the transformation matrices is important.
- It is a difference to first rotate a frame and then translating it, or first translating a frame and then rotating it.

Visualization of transformation order

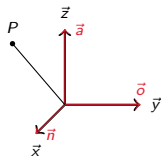
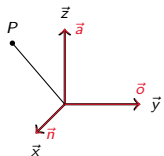
The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

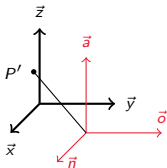
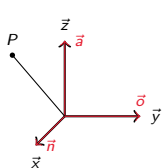


Visualization of transformation order

The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

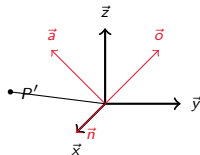
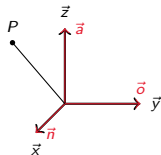
Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$



Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$



Visualization of transformation order

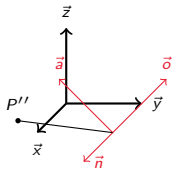
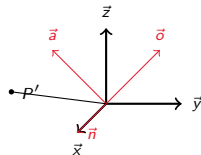
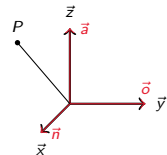
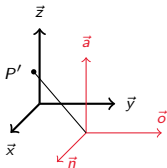
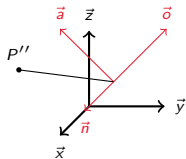
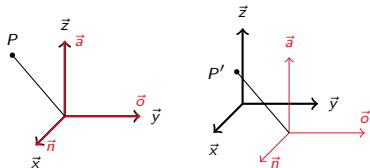
The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

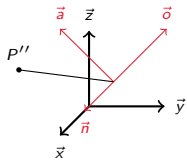
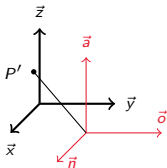
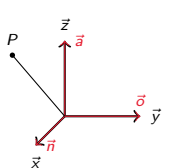


Visualization of transformation order

The position $\langle 0.5, -0.5, 1 \rangle$ of a point P is given relative to the reference frame and transformed by a translation and rotation.

Translation followed by rotation

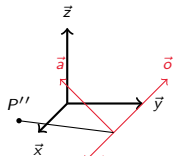
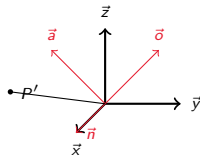
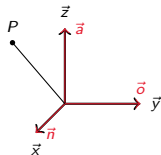
$$P'' = \text{rot}(x, 45^\circ) \times \text{trans}(1, 1, 0) \times P$$



$$P'' = \begin{bmatrix} 1.50 \\ -0.35 \\ 1.06 \end{bmatrix}$$

Rotation followed by translation

$$P'' = \text{trans}(1, 1, 0) \times \text{rot}(x, 45^\circ) \times P$$

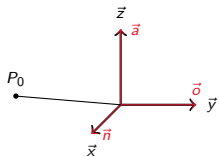


$$P'' = \begin{bmatrix} 1.50 \\ -0.06 \\ 0.35 \end{bmatrix}$$

Example

- The goal is to transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the z-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the x-axis.
 - Rotation of 20° about the y-axis.

$$P_0 = \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

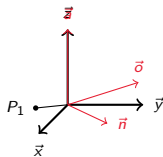
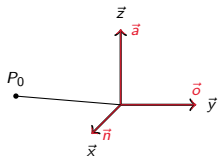


Example

- The goal is to transform a point P_0 at position $\langle 1, -1, 0.5 \rangle$ by the following operations.
 - Rotation of 50° about the z-axis.
 - Translation of $\langle 0.3, 1.2, -0.3 \rangle$.
 - Rotation of 150° about the x-axis.
 - Rotation of 20° about the y-axis.

$$P_0 = \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_1 = \text{rot}(z, 50^\circ) \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 0 \\ 0.77 & 0.64 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 1.41 \\ 0.77 & 0.64 & 0 & 0.12 \\ 0 & 0 & 1 & 0.50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



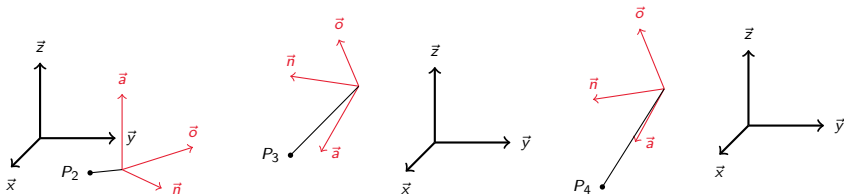
Example (cont.)

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0.3 \\ 0 & 1 & 0 & 1.2 \\ 0 & 0 & 1 & -0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.64 & -0.77 & 0 & 1.41 \\ 0.77 & 0.64 & 0 & 0.12 \\ 0 & 0 & 1 & 0.50 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 1.71 \\ 0.77 & 0.64 & 0 & 1.32 \\ 0 & 0 & 1 & 0.20 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.87 & -0.5 & 0 \\ 0 & 0.5 & -0.87 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.64 & -0.77 & 0 & 1.71 \\ 0.77 & 0.64 & 0 & 1.32 \\ 0 & 0 & 1 & 0.20 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.64 & -0.77 & 0 & 1.71 \\ -0.66 & -0.56 & -0.5 & -1.25 \\ 0.38 & 0.32 & -0.87 & 0.49 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_4 = \text{rot}(y, 20^\circ) \times \text{rot}(x, 150^\circ) \times \text{trans} \begin{pmatrix} 0.3 \\ 1.2 \\ -0.3 \end{pmatrix} \times \text{rot}(z, 50^\circ) \times \text{trans} \begin{pmatrix} 1 \\ -1 \\ 0.5 \end{pmatrix} =$$

$$\begin{bmatrix} 0.94 & 0 & 0.34 & 0 \\ 0 & 1 & 0 & 0 \\ -0.34 & 0 & 0.94 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.64 & -0.77 & 0 & 1.71 \\ -0.66 & -0.56 & -0.5 & -1.25 \\ 0.38 & 0.32 & -0.87 & 0.49 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.74 & -0.61 & -0.30 & 1.77 \\ -0.66 & -0.56 & -0.5 & -1.25 \\ 0.14 & 0.56 & -0.81 & -0.13 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Outline

- 1 Representation of Frame Transformations
- 2 Introduction to Roboop

Roboop

- Object oriented toolbox in C++ for **manipulator simulation**.
- Wrapper for the **newmat** library that implements numerous efficient matrix operations like
 - multiplication, addition, subtraction
 - computing the dot product of two vectors
 - finding the inverse or transposed of a matrix
 - and many more
- Allows the specification of robotic manipulators in **standard DH** and **modified DH** notation.
- Performs **forward kinematic** and **inverse kinematic** computations.
- Web resources concerning Roboop
 - <http://www.cours.polymtl.ca/roboop/>
(Roboop and GLRoboop)
 - http://www.robertnz.net/nm_intro.htm (newmat)

Roboop member functions of class Matrix

- ReturnMatrix `trans(const ColumnVector& a);`
// translates along a specified vector
- ReturnMatrix `rotx(const Real alpha);`
// rotates around the x-axis by α radians
- ReturnMatrix `roty(const Real beta);`
// rotates around the y-axis by β radians
- ReturnMatrix `rotz(const Real gamma);`
// rotates around the z-axis by γ radians
- ReturnMatrix `rotk(const Real theta, const ColumnVector& k);`
// rotates around a chosen axis that is specified in vector k
- ReturnMatrix `rpy(const ColumnVector& a);`
// returns the Matrix given by $rotz(\gamma) \times roty(\beta) \times rotx(\alpha)$,
where α , β and γ are stored in a
- ReturnMatrix `irpy(const Matrix& R);`
// returns the RPY angles that lead to matrix R

Some newmat functionality

- Creating a Matrix object.

```
Matrix A(3,2);  
Real a[] = {11,12,21,22,31,33};  
A << a;
```

- Unary operators.

```
X = -A;      // change sign of elements  
X = A.t();  // transpose  
X = A.i();  // inverse of square matrix A
```

- Binary operators

```
X = A + B;      // matrix addition  
X = A - B;      // matrix subtraction  
X = A * B;      // matrix multiplication
```

This is an overview of Newmats functionality. For a complete reference, please see http://www.robertnz.net/nm_intro.htm.

Creating a translation matrix in Roboop

```
#include <iostream>
#include <utils.h>
#include <newmat.h>
using namespace std;

int main(int argc, const char **argv)
{
    ColumnVector translation(3);
    Matrix oTb21(4,4);

    translation << 1.0 << 1.0 << 1.0;
    oTb21 = trans(translation);

    cout << setprecision(3); // show only 3 decimal digits
    cout << "oTb21" << endl << oTb21 << endl;
    cout << "oTb21*oTb21.i()" <<endl<< oTb21*oTb21.i() <<endl;
}
```

Compile with: `g++ test.cpp -lrooop -lnewmat -o test`

Set the `-I` and `-L` flags correct, if headers or libraries are not found.

Creating a homogenous transformation matrix

```
#include <roboop/ utils .h>
#include <roboop/newmat.h>
#define DTOR(d) ((d) * M_PI / 180)
using namespace std;

int main(int argc, const char **argv)
{
    ColumnVector translation(3), rot_axis(3);
    Matrix oTb21(4,4);

    translation << 3.0 << 5.0 << 7.0;
    oTb21 = trans(translation) * rotx(DTOR(45.0));
}
```

This piece of code creates the matrix from slide 16.