*Alexander Osiik (653897)*

## Applications of Medical Robotics

There are 4 types of Robots

- Robots for **Navigation**
  *Surgical drill* for Precise positioning, motion compensation..
- Robots for **Imaging**
  *Minimally-invasive surgery* by Motion downscaling, reduce tremor...
- Robots for **Motion Replication**
  *Robotic ultrasound* for Automation, speed...
- **Rehabilitation** and **Prosthetics**
  *Exoskeletons* that Replace damaged structures, autonomous rehabilitation...

Application: Epiphyseolysis

## Navigation

### Problem: We have an image.

- Where is the **Robot** in the image coordinate system?
- Where is the **Target** in the image?

### Ingridients for Radiologic Navigation:

- **C-Arm Robot:**
  A mobile x-ray system, robot containing 2 prismatic and 3 revolute Joints
- **Infrared Tracking:**
  Place Marker on Endeffector, on Target and on C-Arm. A Camera the ncalculates the positions:
  - Skin incision
  - Place marker on bone
  - Take 2 C-Arm images
  - Tavigate

$\Longrightarrow$ Navigation, Registration, Calibration!

### Spatial position and orientation

Every Transformation is a multiplication, simply write down the vectors.
Rotatory Matrices:

$$R(x,\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_\theta & -S_\theta & 0 \\ 0 & S_\theta & C_\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, R(y,\theta) = \begin{pmatrix} C_\theta & 0 & S_\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S_\theta & 0 & C_\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, R(z,\theta) = \begin{pmatrix} C_\theta & -S_\theta & 0 & 0 \\ S_\theta & C_\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translatory Matrices:

$$T(x,a) = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T(y,a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, T(z,a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

### Denavit Hartenberg Parameter

TODO
In **Forward Kinematics**, we derive the Endeffector Position by multiplication of given Joint Matrices, e.g.

$$^{B}M_G =^{B} M_1 \cdot^{1} M_2 \cdot^{2} M_3 \cdot^{3} M_G$$

In **Reverse Kinematics**, we calculate the Joint angles from a given Endeffector Position, e.g. One Joint Robot:
$P = (P_X, P_Y)$, Get Angle $\theta$ by: $\frac{P_Y}{P_X} = \frac{\sin\theta}{\cos\theta} = \tan\theta \implies \theta = atan2(P_Y, P_X)$
Use atan2-Function, as it gives back the angles of full circle depending on signs

## Navigation & Registration

### Problem: We have >1 images.

- Rotate images in such way, that images align
- Output is $\theta, \triangle x, \triangle y$

There are various types of Registration:

- point-based
- landmark-based
- contour-based
- intensity-based
- elastic

### Landmark based registration

Situation: We have an MR image of a head. We place 4 Landmarks (Nose and Mouth). The Target Point is given with respect to the four landmark points.
From this, two point clouds $A, B$ are generated.
We assume: $\#A = \#B,\ 2D,\ a_1 \to b_1\ etc.$
If we move $B$, this leads to:

$$b_1 \rightsquigarrow R \cdot b_1 + t,\ R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, t = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$
$$b_2 \rightsquigarrow R \cdot b_2 + t$$
$$\vdots$$
$$b_n \rightsquigarrow R \cdot b_n + t$$

We create a function $f$, which is the distance between the two point clouds $A$ and $B$. We want to minimize this function:

$$f = ||R \cdot b_1 + t - a_1||^2 + \cdots + ||R \cdot b_n + t - a_n||^2$$

### Gaussian Least Square

If we assume that the angle $\theta$ is small, we can linearize the rotational matrix R such that:

$$\cos\theta \sim 1,\ \sin\theta \sim \theta \implies R = \begin{pmatrix} 1 & -\theta \\ \theta & 1 \end{pmatrix}$$

This matrix $R$ is linear in $\theta$.
We can also linearize the 3D rotation matrix as follows:

$$R(x,\alpha) \cdot R(y,\beta) \cdot R(z,\gamma) = \begin{pmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & \alpha & 1 \end{pmatrix}$$

As the multiplication of two small values leads to an even smaller value. This method works fine until angles of 10 degrees.

### Iterative Closest Points (ICP)

Assume we have a point cloud $A$ and a point cloud $B$. We want know the transformation needed to overlap $B$ over $A$. We do **not** deform $B$, as all motion is rigid (rotation and translation).

Cloud A:

| | | | |
|---|---|---|---|
| 1 | 1 | 3 | 3 |
| 1 | 1 | 3 | 3 |
| 2 | 2 | 3 | 3 |
| 2 | 2 | 3 | 3 |

Cloud B

| | | | |
|---|---|---|---|
| 2 | 2 | 1 | 1 |
| 2 | 2 | 1 | 1 |
| 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 |

$\implies$ Just rotate $90^o$ to align

But sometimes, coloring differs in images depending on the imaging process. For Example, in CT the bone has a bright white color, while in MR the bone is more gray:

Cloud A: $\begin{array}{|c|c|c|c|}\hline 1&1&3&3\\\hline 1&1&3&3\\\hline 2&2&3&3\\\hline 2&2&3&3\\\hline\end{array}$ Cloud B $\begin{array}{|c|c|c|c|}\hline 3&3&1&1\\\hline 3&3&1&1\\\hline 2&2&2&2\\\hline 2&2&2&2\\\hline\end{array}$ $\Longrightarrow$ rotate $90^o$ enough?

### Mutual Information (MI)

For **Mutual Information** Registration, the color mapping does not have to be exactly 1x1. It can be different, as shown in the example above.
**Basic Definition:**
We generate Images $A, B$ in a random process. For each pixel of the image, we throw a dice (random generation), so we map 6 colors $\{1, \dots, 6\}$ to each pixel. $A, B$ are random variables with distributions $P_A, P_B$, namely:

- $P_A(a)$ = Probability of grey level value $a$ in image $A$
- $P_B(b)$ = Probability of grey level value $b$ in image $B$

$\Longrightarrow$ $P_{A,B}(a,b)$ = Probability of grey level value $a$ in image $A$ occuring at the same position in image $B$

Examples:

- $A \neq B$ : $A$ tells nothing about $B$, 2 different images
- $A = B$ : $A$ tells everything about $B$
- $A$ tells something about $B$

Keep in mind the law of Independence of two random variables:

$$P_{A,B}(a,b) = P_A(a) \cdot P_B(b)$$

We want to know the maximum Mutual Information, calculated as follows:

$$I(A,B) = \sum_{a,b} P_{A,B}(a,b) \cdot \log_2 \left( \frac{P_{A,B}(a,b)}{P_A(a) \cdot P_B(b)} \right)$$

The Sum ranges over all greyscale pairs of diefferent colors. $I(A,B)$ measures the information dependencies. We maximize the Mutual Information to find exact position.
Example 1:

$$A = \begin{array}{|c|c|}\hline 0 & 1\\\hline 1 & 0\\\hline\end{array}\,, \quad B = \begin{array}{|c|c|}\hline 0 & 1\\\hline 1 & 0\\\hline\end{array}$$

$$P_A(0) = 0.5,\ \ P_B(0) = 0.5$$
$$P_A(1) = 0.5,\ \ P_B(1) = 0.5$$
$$P_{A,B}(0,0) = 0.5,\ \ P_{A,B}(0,1) = 0$$
$$P_{A,B}(1,1) = 0.5,\ \ P_{A,B}(1,0) = 0$$
$$\sum = 1$$
$$I(A,B) = \sum_{a,b} P_{A,B}(a,b) \cdot \log_2\left(\frac{P_{A,B}(a,b)}{P_A(a)\cdot P_B(b)}\right)$$
$$= 0.5{\cdot}\log_2\left(\frac{P_{A,B}(0,0)}{P_A(0)\cdot P_B(0)}\right) + 0.5{\cdot}\log_2\left(\frac{P_{A,B}(1,1)}{P_A(1)\cdot P_B(1)}\right) + 0{\cdot}\log_2\left(\frac{P_{A,B}(0,1)}{P_A(0)\cdot P_B(1)}\right) + 0{\cdot}\log_2\left(\frac{P_{A,B}(1,0)}{P_A(1)\cdot P_B(0)}\right)$$
$$= 0.5 \cdot \log_2(2) + 0.5 \cdot \log_2(2)$$
$$= 1$$

Example 2:

$$A = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline\end{array}\ ,\quad B = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline\end{array}$$

$MI(A,B) = 1$, as we care for the structure of the image. The color values do not really matter. Example 3:

$$A = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline\end{array}\ ,\quad B = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline\end{array}$$

$MI(A,B) = 0$, as the structures are very different. Example 4:

$$A = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline\end{array}\ ,\quad B = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline\end{array}$$

$MI(A,B) = 0.32$
Keep in mind that $MI \geq 0$, but can go up to 100 for large images!

#### Calculating $P_A$, $P_B$, $P_{A,B}$

Just create a table, then divide the right hand side by the amount of pixels/voxels:

| | |
|---|---|
| 0 | number of Pixels in image $A$ with greylevel 0 |
| $\vdots$ | $\vdots$ |
| 255 | number of Pixels in image $A$ with greylevel 255 |

Calculating $P_{A,B}$ is a little difficult:

| | $0 \dots$ | $i$ | $\dots$ 255 |
|---|---|---|---|
| 0 | | | |
| $\vdots$ | | $\vdots$ | |
| $i$ | $\dots$ | # pixel pins with colors $i, j$ | |
| $\vdots$ | | | |
| 255 | | | |

There are 3 algorithms/methods for calculation:

1. **Nearest Neighbour (NN)**
2. **Trilinear Interpolation (TI)**
3. **Trilinear Partial Volume (TPV)**

## Imaging

#### Image Deformation

There are multiple applications:

- Elastic Registration
- Distortion Correction

#### Bilinear Onterpolation

Assume we have 2 adjacent grid points $u, v$. To get from $G'$ to $G$ we have to deform $u$ in x-direction by $du$. We obtain information for $w$ by **linear interpolation**. Same for the y-direction.
**Advantage:**

- Simplicity
- We will move all points to its goal $\implies$ Accuracy!

**Disadvantage:**

- Not smooth

### Cubic Spline Interpolation

We define a correction polynomial, two variables $x, y$

$$a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 + a_6 xy^2 + a_7 x^2 y + a_8 x^3 + a_9 y^3$$

$$b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2 + b_6 xy^2 + b_7 x^2 y + b_8 x^3 + b_9 y^3$$

We have our model: $(x_m, y_m)$ and our distorted points $(x_d, y_d)$.
We further assume, that $a$'s and $b$'s are variables, while $x, y$ are constant.
We set up $f$ as:

$$f = [(x_m, y_m) - (x_d, y_d)]^2$$

For $N$ grid points, we adjust $f$ to:

$$f = [(x_m^{(1)}, y_m^{(1)}) - (x_d^{(1)}, y_d^{(1)})]^2 + \cdots + [(x_m^{(N)}, y_m^{(N)}) - (x_d^{(N)}, y_d^{(N)})]^2$$

With $x_d$ being the $a$-Polynomial and $x_m$ being the $b$-Polynomial, function $f$ is quadratic in the $a$'s and $b$'s.

## Motion Replication