
Path planning in the robotics: towards intelligent path planning based on AI

Robotix-Academy Roadshow 2020

Stefan Marx, B.Eng.

ZeMA

Zentrum für Mechatronik und Automatisierungstechnik

Saarbrücken, 10.11.2020

Agenda

- 1 Use case introduction**
- 2 Conventiionell path planning**
- 3 Spline interpolation: manual selection of support points**
- 4 Motivation of AI-based path planning and processes**
- 5 Basics: Reinforcement Learning and Q-Learning**
- 6 Approach and Development of an AI based process**
- 7 Conclusion and Outlook**

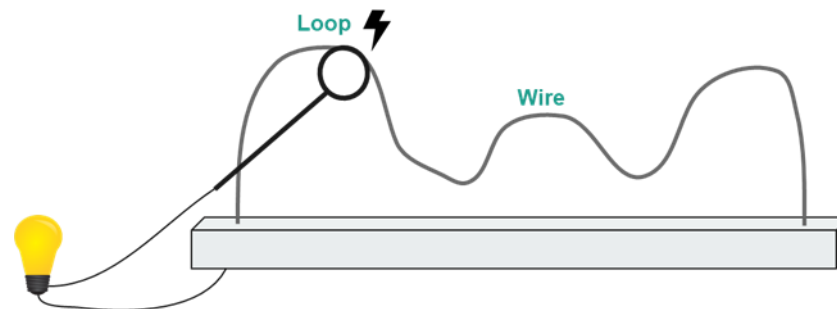
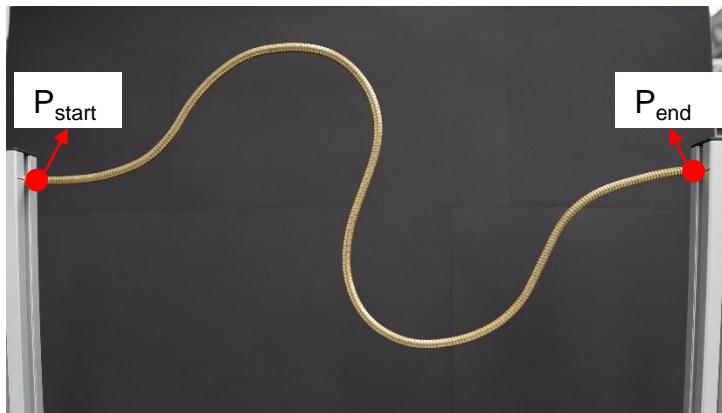
Agenda

- 1 Use case introduction**
- 2 Conventiionell path planning**
- 3 Spline interpolation: manual selection of support points**
- 4 Motivation of AI-based path planning and processes**
- 5 Basics: Reinforcement Learning and Q-Learning**
- 6 Approach and Development of an AI based process**
- 7 Conclusion and Outlook**

Use case Introduction

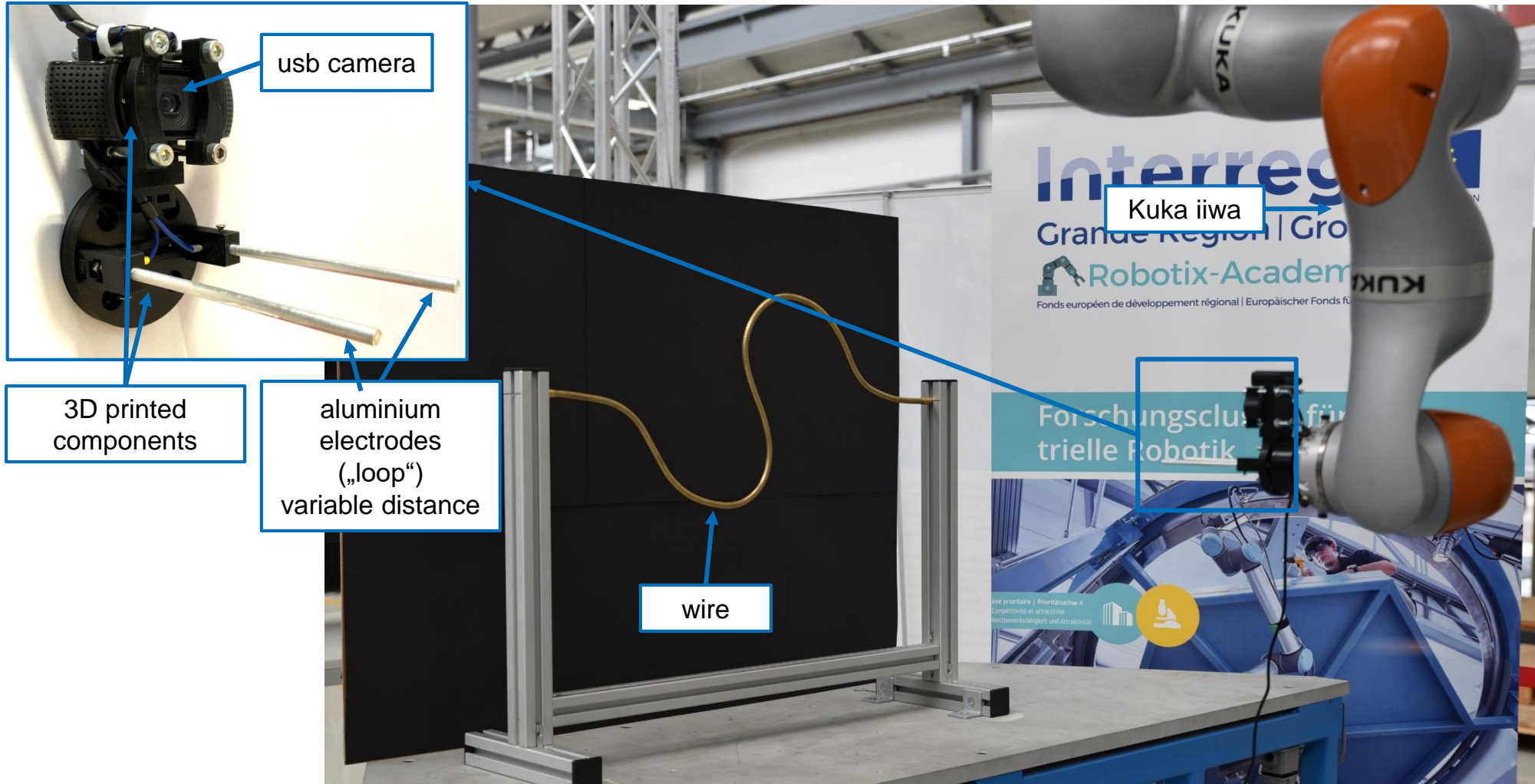


- play the Wire Loop Game with a robot
- start and end point of the wire are predefined
- wire can be curved arbitrarily between those points
- operational space: 2-dimensional plane (y, z)
 - robot move in predefined plane (y, z)
 - x is set and blocked



Reference: <https://www.kindpng.com>; www.robots.com/robots/kuka-lbr-iiwa-14-r820

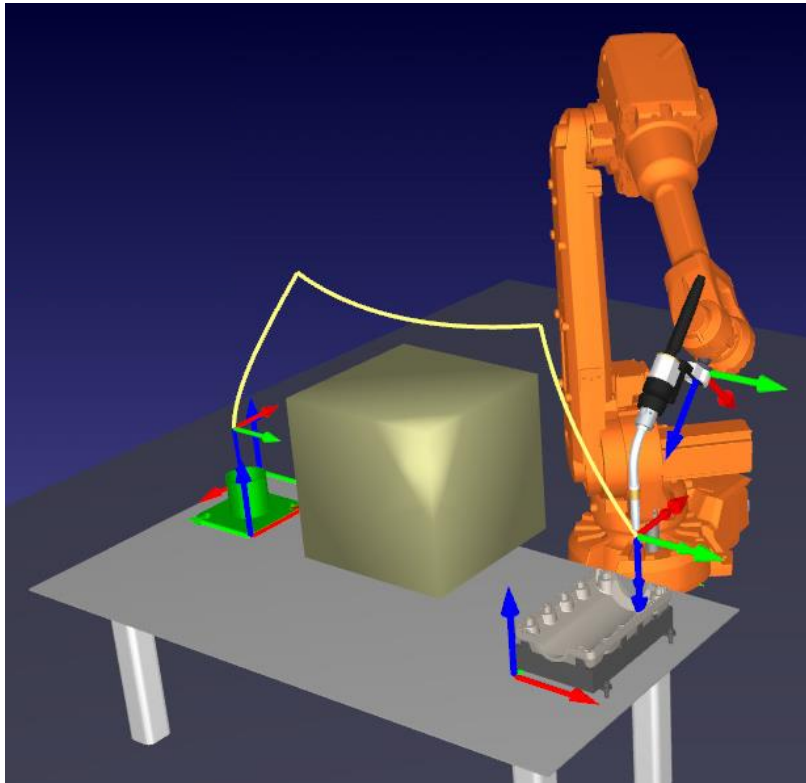
Experimental Setup



Agenda

- 1 Use case introduction
- 2 **Conventionell path planning**
- 3 Spline interpolation: manual selection of support points
- 4 Motivation of AI-based path planning and processes
- 5 Basics: Reinforcement Learning and Q-Learning
- 6 Approach and Development of an AI based process
- 7 Conclusion and Outlook

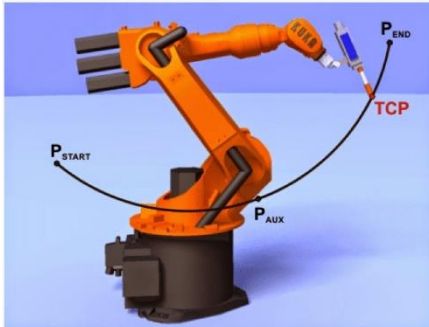
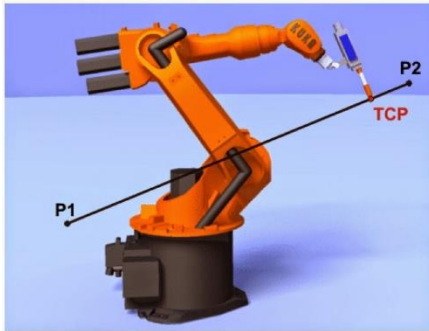
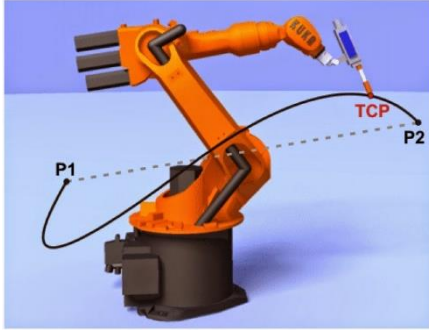
Path planning



- Path planning includes the geometrical description of a motion task under consideration of:
 - path or route conditions
 - time conditions (usually indirectly given by speed and acceleration)
 - robot kinematics (reachability of a pose)
 - collision areas
- several types of movement are available
 - point to point, linear, circular, spline
- robot movement results from a sequence of simple path segments

Reference: <https://robodk.com/blog/motion-planning-trend/>

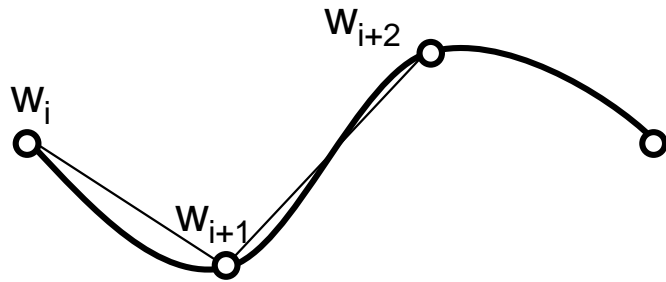
Conventional path planning



- point to point movement
 - interpolation of all axis between start and end point
 - the path between start and end point is geometrically undefined, velocity of the TCP is not constant
 - used for fast movement between to fixed points
 - e.g. part transport through the free space
- linear movement
 - robot guides the TCP along a straight path
 - constant velocity
 - path is predicable
 - e.g. used for joining processes
- circular movement
 - circle segment defined by three points
 - constant velocity
 - e.g. used in gluing or welding processes

Reference: [://mkmra2.blogspot.com/](http://mkmra2.blogspot.com/), Grundlagen Automatisierung

Conventional path planning



■ spline interpolation:

- interpolation method for generating "smooth" curves
- specification: consistency in position, gradient and curvature

– 5th-order polynomial:

- $w(\lambda) = \mathbf{a}_{i,5} \lambda^5 + \mathbf{a}_{i,4} \lambda^4 + \mathbf{a}_{i,3} \lambda^3 + \mathbf{a}_{i,2} \lambda^2 + \mathbf{a}_{i,1} \lambda^1 + \mathbf{a}_{i,0}$
- $\lambda = \lambda(s)$ (polynomial parameter)
- s (path parameter)

- path is mathematically described
- constant velocity
- e.g. used for machining of freeform surfaces

Programming the robot


■ Online:

– Teach-In:

- manually move robot to striking positions
- store positions and lodge interpolation type, velocity, acceleration, blending...

– Playback:

- force-torque-sensor at end effector
- robot manually guided along the path to be moved

- 
- commonly used
 - for easy, small applications
 - not flexible or adaptable


■ Offline:

– text based:

- programming in robot specific language

– graphic-supported methods:

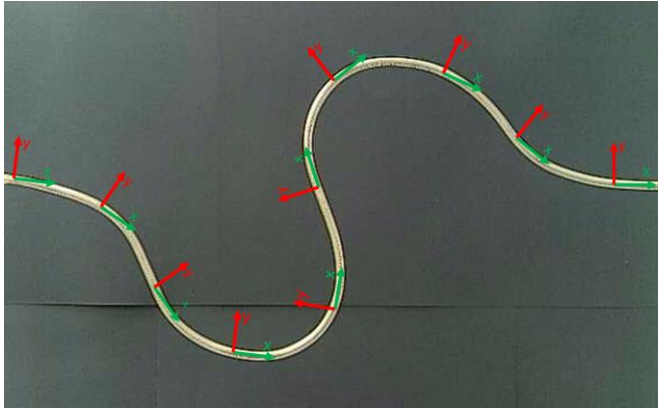
- Interactive input of the motion path on the screen
- textual input of the program sequences

- 
- more complex applications
 - adaptability depending on the programmed code

Agenda

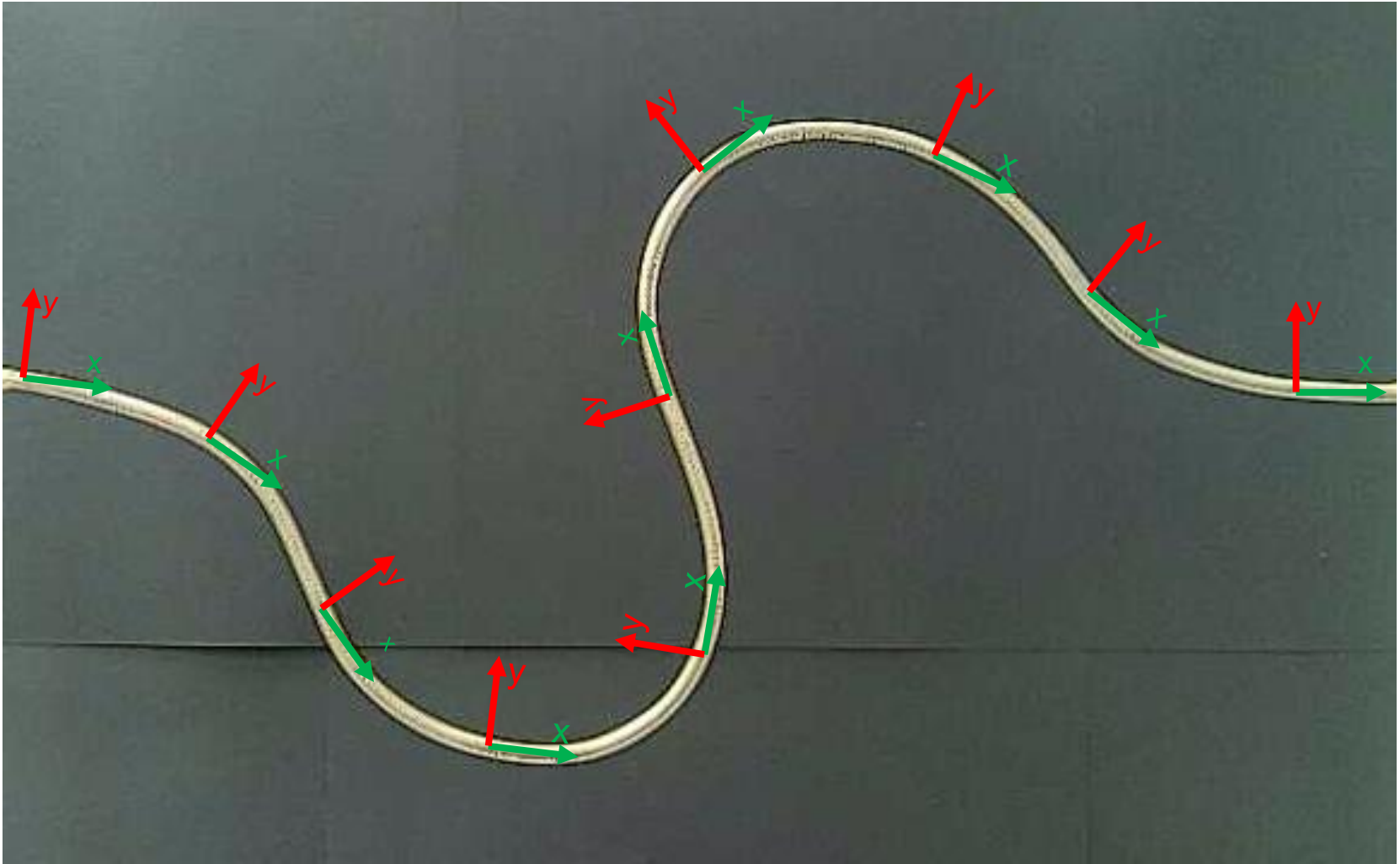
- 1** Use case introduction
- 2** Conventionell path planning
- 3** Spline interpolation: manual selection of support points
 - 3.1** Image processing
 - 3.2** Camera Calibration
 - 3.3** Implementation
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
- 7** Conclusion and Outlook

Spline interpolation: clicking points (1)



- Approach:
 - manually select the support point for the spline interpolation
 - visualise the result
 - calculate an error between spline and target path
- Input:
 - image of the motion problem (wire)
- Necessary preparations:
 - image processing
 - simplify the image to most important information
 - camera calibration
 - working plane to camera
 - camera to robot flange
 - allows transformation from pixel to mm or from pixel to robot base coordinate system

Spline points



Agenda

- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
 - 3.1** Image processing
 - 3.2** Camera Calibration
 - 3.3** Implementation
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
- 7** Conclusion and Outlook

The flowchart illustrates the image processing steps for path extraction:

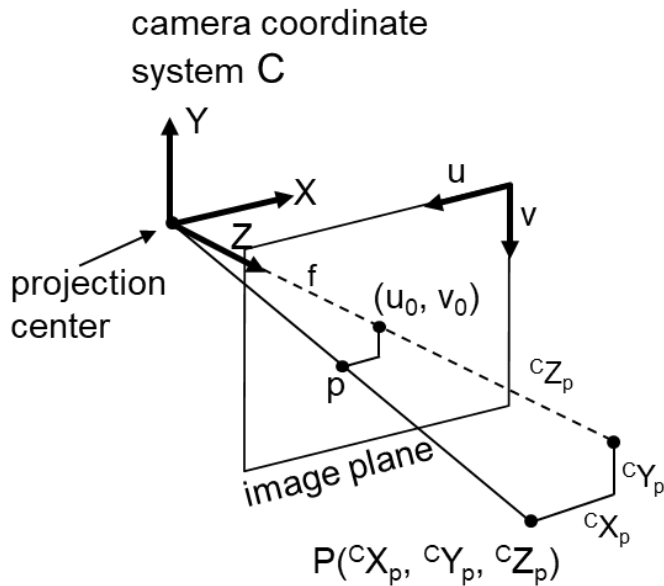
- a: Original Image**
- b: Cropping and Rotation by 180°**
- c: Conversion to Grayscale**
- d: Threshold Function** (apply threshold to get binary image)
- e: Average Function (column-wise)** (average column-wise)
- f: Average Function (column-wise + row-wise)** (add average row-wise)
- g: Final Pixel Path** (shortest path (cost function))
- h: Pixel Path Array** (convert in array)

The final output is a binary image showing the shortest path (g) and its corresponding pixel path array (h).

Agenda

- 1 Use case introduction
- 2 Conventiell path planning
- 3 Spline interpolation: manual selection of support points
 - 3.1 Image processing
 - 3.2 Camera Calibration
 - 3.3 Implementation
- 4 Motivation of AI-based path planning and processes
- 5 Basics: Reinforcement Learning and Q-Learning
- 6 Approach and Development of an AI based process
- 7 Conclusion and Outlook

Camera model: pinhole-camera



■ left:

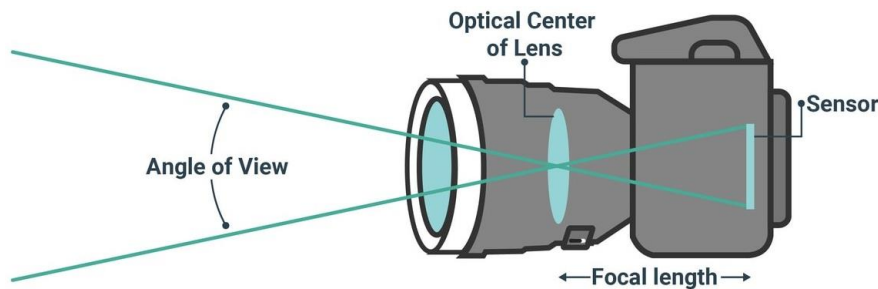
- camera coordinate system C (C-CS) with the origin in the projection center
- cZ -axis = optical axis

■ middle:

- image plan with axes u & v (unit = pixel)
- in distance f to the C-CS
- distance f = focal length

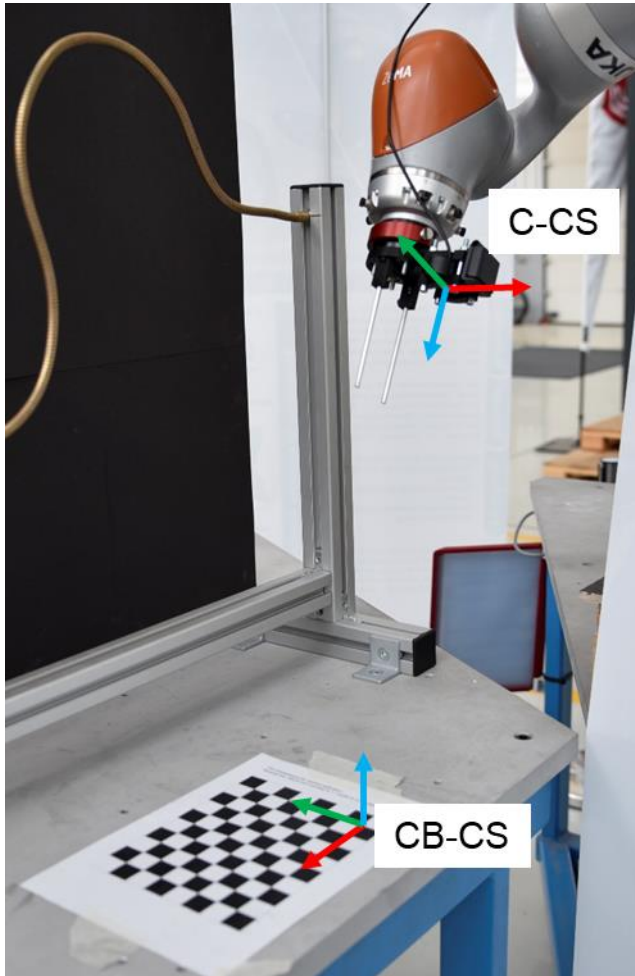
■ right:

- world coordinates



Quelle: <https://www.researchgate.net>, <https://capturetheatlas.com/what-is-focal-length/>

Camera calibration with OpenCV



■ calibration with the chessboard pattern

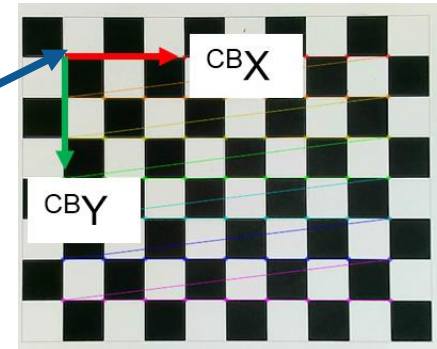
– input:

- number of rows and columns of the pattern -1
- edge length of a square in the pattern
- 15 images (or more) of the chessboard pattern from different positions and orientations

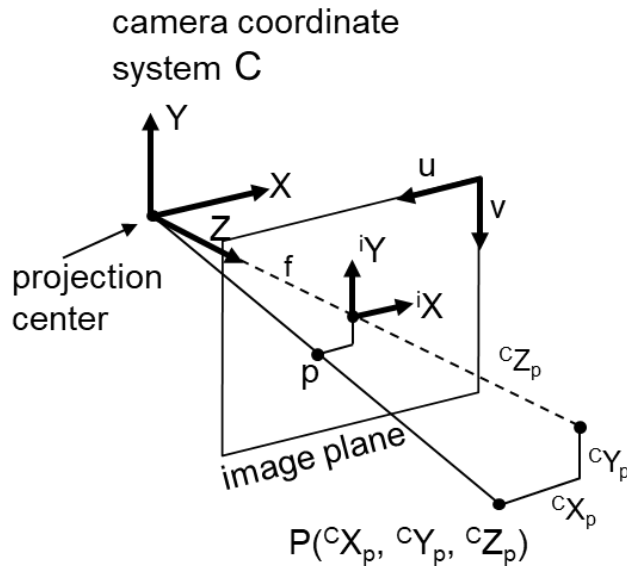
– output:

- camera matrix: 3x3, intrinsic parameters
- for every image: translational vector from C-CS to CB-CS
- for every image: rotational vector of the CB-CS regarding to the C-CS (Rodriguez vector)

Origin of the chessboard coordinate system (CB-CS)



Intrinsic parameters



- for the point p in the image plane:

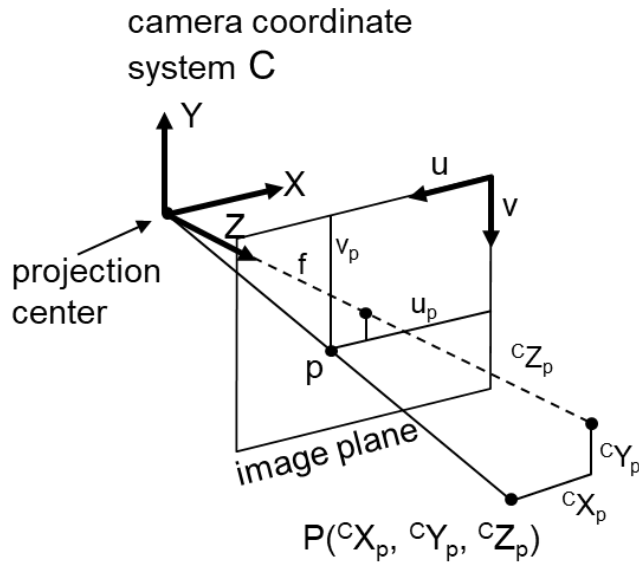
- $iX_p = f * \frac{cX_p}{cZ_p}$
 - $iY_p = f * \frac{cY_p}{cZ_p}$

- $$\begin{pmatrix} iX_p \\ iY_p \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cX_p/cZ_p \\ cY_p/cZ_p \\ 1 \end{pmatrix}$$

- iX_p und iY_p as well as cX_p , cY_p und cZ_p are in the chosen length unit (e.g. mm)
- the conversion from mm to pixels is done in a further step

1) mm oder eine beliebige andere Längenmaßeinheit

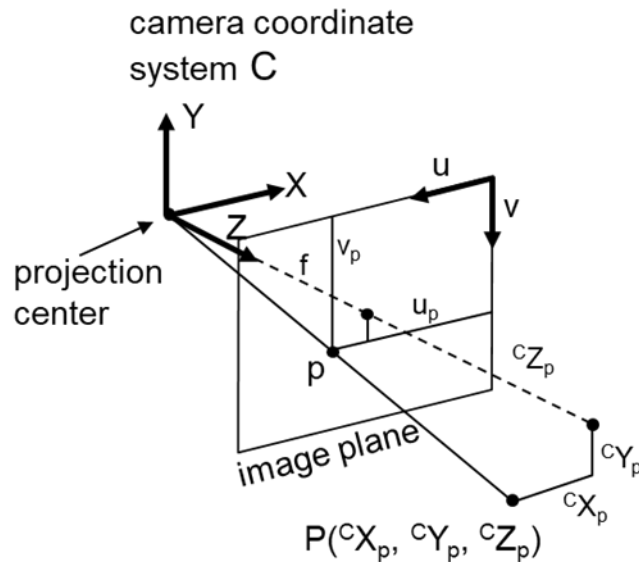
Intrinsic parameters



- conversion from mm to pixels :
 - factors s_x and s_y define the dimensions of a pixel in the image plane (unit: pixel/mm)
 - with u_0 and v_0 the coordinate system of the image plane is moved to the intersection of the image plane with the optical axis¹.
- for the point p in the image plane:
 - $u_p = f * s_x * \frac{cX_p}{cZ_p} + u_0 = f_x * \frac{cX_p}{cZ_p} + u_0$
 - $v_p = f * s_y * \frac{cY_p}{cZ_p} + v_0 = f_y * \frac{cY_p}{cZ_p} + v_0$
- $$\begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & u_0 \\ 0 & s_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cX_p/cZ_p \\ cY_p/cZ_p \\ 1 \end{pmatrix}$$
- $$\begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cX_p/cZ_p \\ cY_p/cZ_p \\ 1 \end{pmatrix}$$

1) Center of the image plane \neq Point of intersection of the image plane with the optical axis

Intrinsic parameters



- $$\begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cX_p/cZ_p \\ cY_p/cZ_p \\ 1 \end{pmatrix}$$

- $$\begin{pmatrix} cX_p/cZ_p \\ cY_p/cZ_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix}$$

- the equation only provides the ratios cX_p/cZ_p and cY_p/cZ_p
- to get point $P(cX_p, cY_p, cZ_p)$ further calculations are necessary
- limitations: The point P has to be in the plane of the chessboard pattern to get accurate results

Extrinsic parameters

- translational vector $\vec{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$

- rotational vector (Rodriguez vector) $\vec{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$

- \rightarrow transformation matrix ${}^C T_{SB}$ from C-CS to CB-CS: ${}^C T_{CB} = \begin{pmatrix} d_{11} & d_{12} & d_{13} & t_x \\ d_{21} & d_{22} & d_{23} & t_y \\ d_{31} & d_{32} & d_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- $\begin{pmatrix} {}^C X_p \\ {}^C Y_p \\ {}^C Z_p \\ 1 \end{pmatrix} = \begin{pmatrix} d_{11} & d_{12} & d_{13} & t_x \\ d_{21} & d_{22} & d_{23} & t_y \\ d_{31} & d_{32} & d_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} {}^{CB} X_p \\ {}^{CB} Y_p \\ 0 \\ 1 \end{pmatrix} \rightarrow {}^{CB} Z_p = 0$ in the plane of the chessboard pattern

Calculation of point P(${}^C X_p$, ${}^C Y_p$, ${}^C Z_p$)

$$\begin{bmatrix} {}^C X_p / {}^C Z_p \\ {}^C Y_p / {}^C Z_p \\ 1 \end{bmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}$$

$$\begin{aligned} \rightarrow & \begin{aligned} & \blacksquare {}^C X_p = a * {}^C Z_p \\ & \blacksquare {}^C Y_p = b * {}^C Z_p \\ & \blacksquare {}^C Z_p = {}^C Z_p \end{aligned} \end{aligned}$$

$$\begin{bmatrix} {}^C X_p \\ {}^C Y_p \\ {}^C Z_p \\ 1 \end{bmatrix} = \begin{pmatrix} d_{11} & d_{12} & d_{13} & t_x \\ d_{21} & d_{22} & d_{23} & t_y \\ d_{31} & d_{32} & d_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} {}^{CB} X_p \\ {}^{CB} Y_p \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \rightarrow & \begin{aligned} & \blacksquare {}^C X_p = d_{11} * {}^{CB} X_p + d_{12} * {}^{CB} Y_p + t_x \\ & \blacksquare {}^C Y_p = d_{21} * {}^{CB} X_p + d_{22} * {}^{CB} Y_p + t_y \\ & \blacksquare {}^C Z_p = d_{31} * {}^{CB} X_p + d_{32} * {}^{CB} Y_p + t_z \end{aligned} \end{aligned}$$

$$\blacksquare a * {}^C Z_p = d_{11} * {}^{CB} X_p + d_{12} * {}^{CB} Y_p + t_x$$

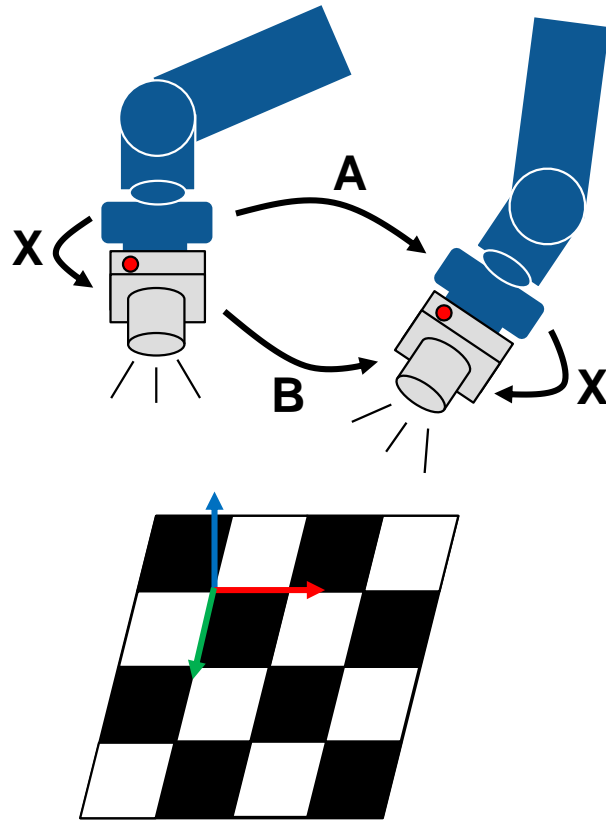
$$\blacksquare b * {}^C Z_p = d_{21} * {}^{CB} X_p + d_{22} * {}^{CB} Y_p + t_y$$

$$\blacksquare {}^C Z_p = d_{31} * {}^{CB} X_p + d_{32} * {}^{CB} Y_p + t_z$$

$$\begin{aligned} \rightarrow & \begin{aligned} & \blacksquare t_x = -d_{11} * {}^{CB} X_p - d_{12} * {}^{CB} Y_p + a * {}^C Z_p \\ & \blacksquare t_y = -d_{21} * {}^{CB} X_p - d_{22} * {}^{CB} Y_p + b * {}^C Z_p \\ & \blacksquare t_z = -d_{31} * {}^{CB} X_p - d_{32} * {}^{CB} Y_p + 1 * {}^C Z_p \end{aligned} \end{aligned}$$

$$\begin{bmatrix} {}^{CB} X_p \\ {}^{CB} Y_p \\ {}^C Z_p \end{bmatrix} = \begin{pmatrix} -d_{11} & -d_{12} & a \\ -d_{21} & -d_{22} & b \\ -d_{31} & -d_{32} & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Camera to flange calibration

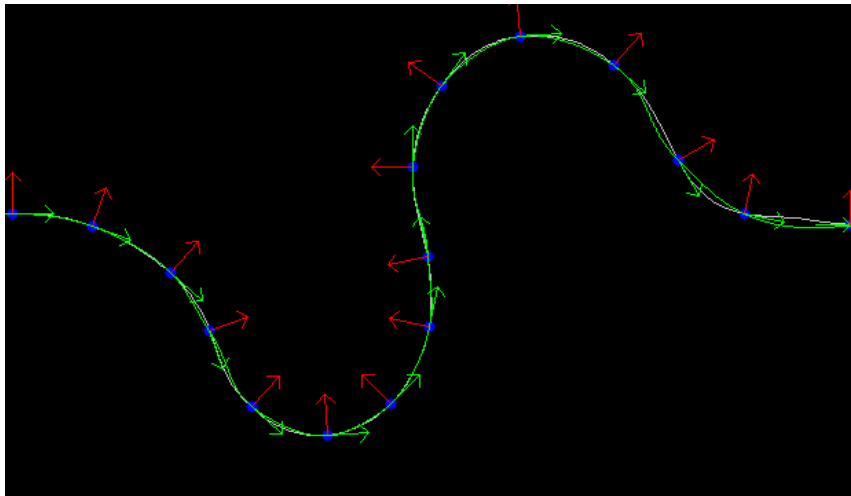
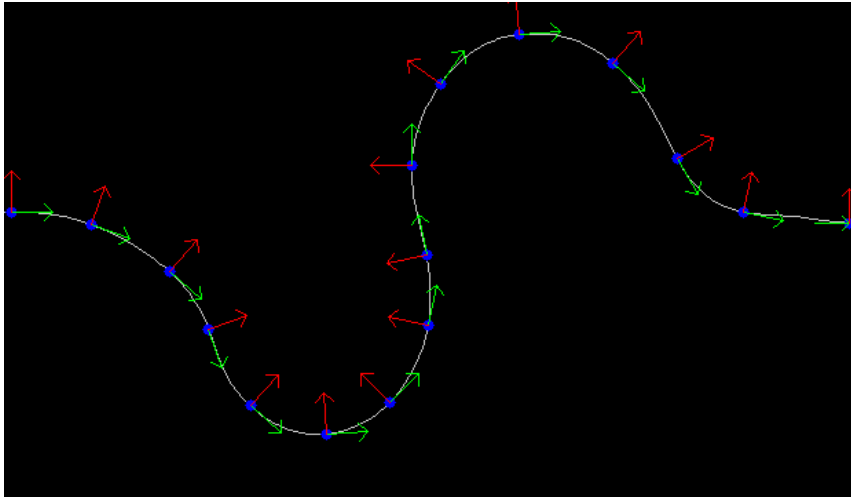


- Calibrate chessboard to camera from four different positions and orientations:
 - store transformation matrixes ${}^{C,i}T_{CB}$ ($i=1..4$)
 - store flange positions ${}^BT_{F,i}$ ($i=1..4$)
- Calibration is done by solving the equation
 - $AX = XB$
 - X : transformation matrix flange to camera CT_F
 - A : transformation matrix flange to flange
 - B : transformation matrix camera to camera
- for more information see paper:
 - Best-t method for the calibration of 3D objects using a laser line sensor mounted on the flange of an articulated robot

Agenda

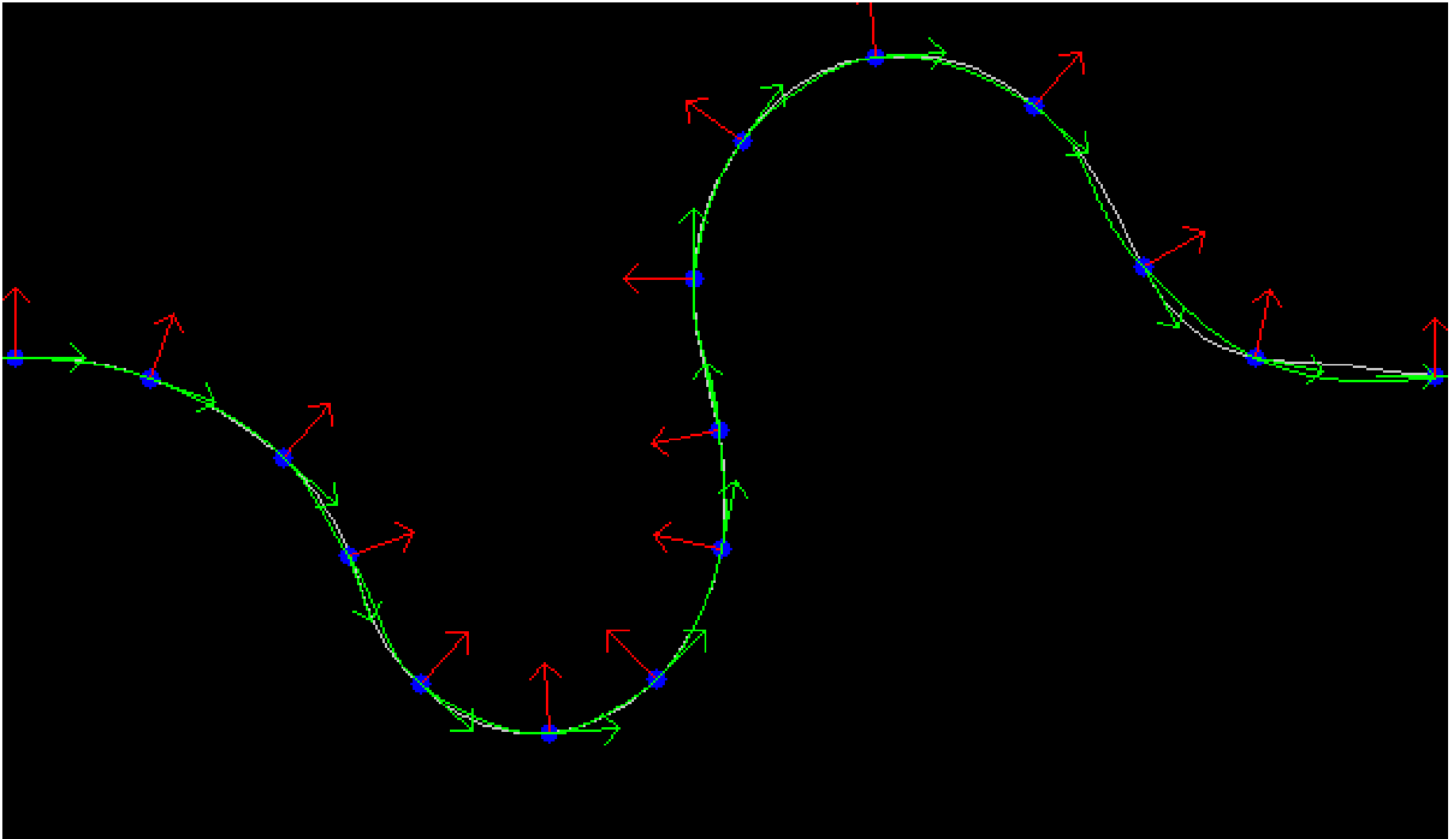
- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
 - 3.1** Image processing
 - 3.2** Camera Calibration
 - 3.3** Implementation
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
- 7** Conclusion and Outlook

Spline interpolation: clicking points (2)

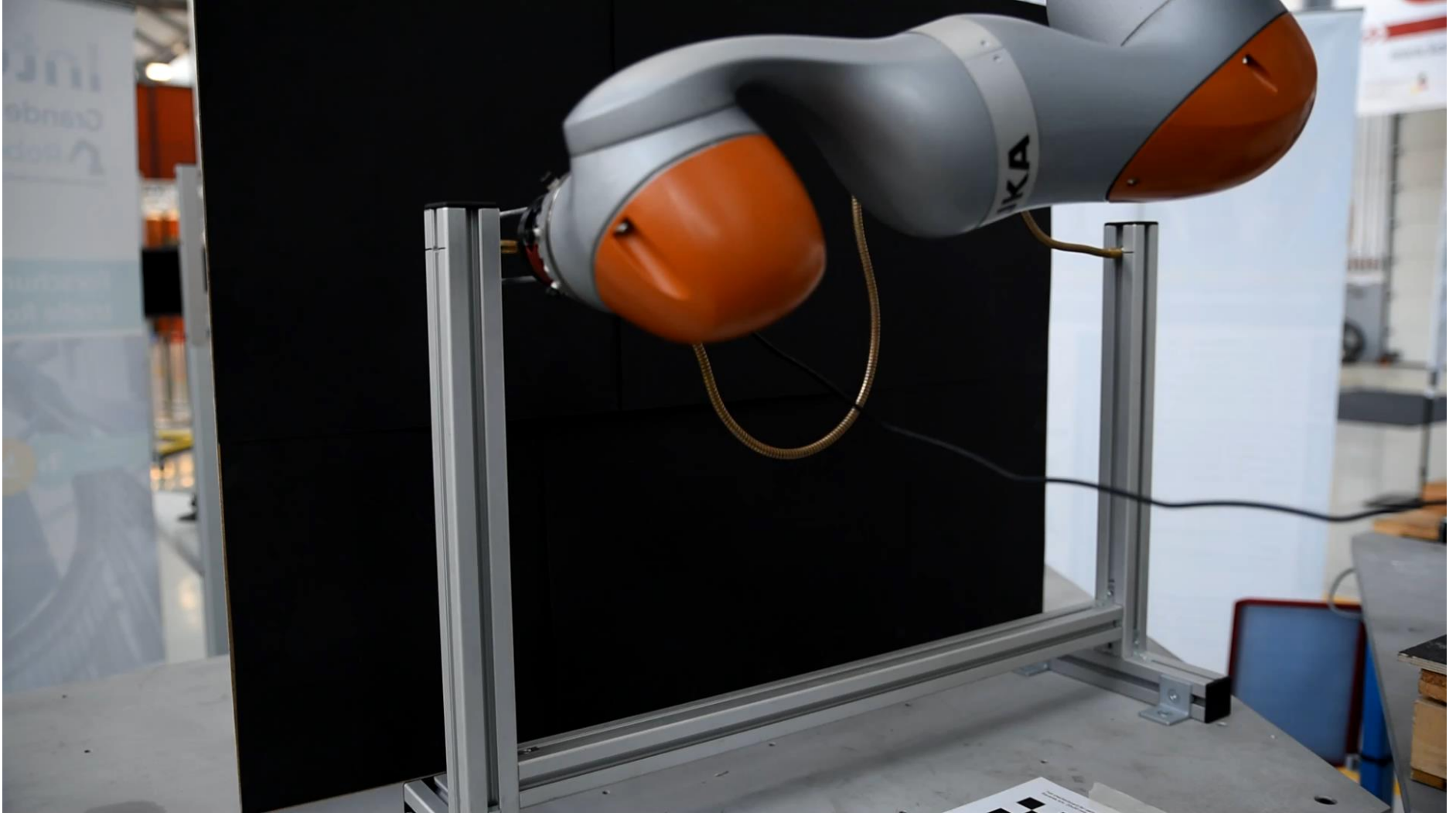


- 1. step: image processing
 - provide a edited image of the motion problem (wire)
- 2. step: define the support points for spline interpolation
 - choose support points by clicking
- 3. step: calculate spline
 - 5th-order spline
- 4. step: checking:
 - visually: fade in the spline in the edited image
 - calculate the maximum variance to the wire
 - in this example: max variance ≈ 9.5 mm
- 5. step: after accepting the result, send spline to robot
 - using MQTT connection

Spline interpolation: clicking points (2)



Implementation



Agenda

- 1 Use case introduction
- 2 Conventiionell path planning
- 3 Spline interpolation: manual selection of support points
- 4 Motivation of AI-based path planning and processes**
- 5 Basics: Reinforcement Learning and Q-Learning
- 6 Approach and Development of an AI based process
- 7 Conclusion and Outlook

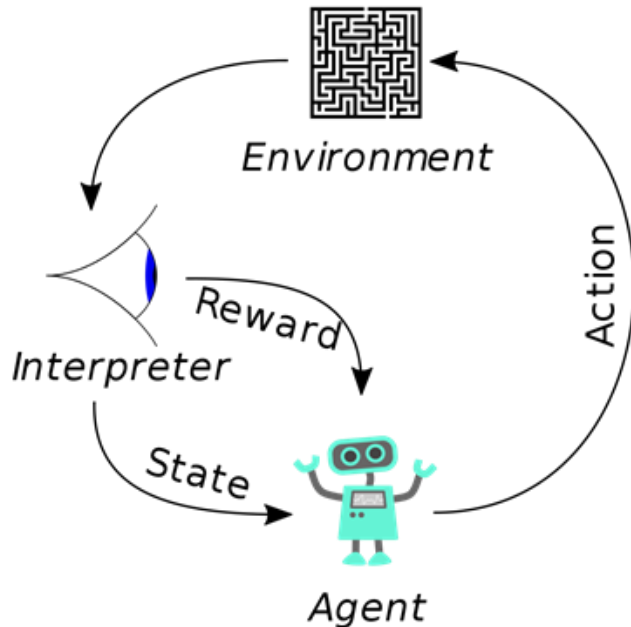
Motivation of AI-based path planning

- Conventional path planning
 - robot path results from **fixed programmed** support points connected with simple path segments:
 - linear, circular, ptp
 - path is given by the programmer
 - not flexible or adaptable
- Complex tasks may need some intelligent path planning methods
 - robot/robot application decides by itself how to move
- AI algorithms to implement intelligent path planning
 - Reinforcement Learning:
 - reinforcement learning agent registers changes in its environment
 - motion planning adapts to the changed conditions
- → **AI algorithm learns to orientate the TCP!**

Agenda

- 1 Use case introduction
- 2 Conventiionell path planning
- 3 Spline interpolation: manual selection of support points
- 4 Motivation of AI-based path planning and processes
- 5 Basics: Reinforcement Learning and Q-Learning**
- 6 Approach and Development of an AI based process
- 7 Conclusion and Outlook

Reinforcement Learning



■ Reinforcement Learning:

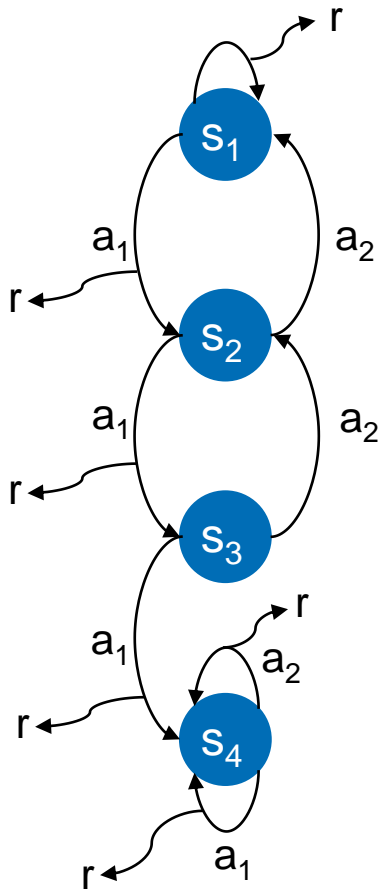
- Generic term for a range of machine learning methods
- Agent learns from interactions with its environment
- goal of the agent: find a strategy (policy) that maximizes the total (future) reward
- trial-and-error learning in a dynamic environment

■ Q-Learning:

- Method from the category Reinforcement Learning
- Model-free method:
 - Agent does not need information about the concept of the problem in advance
 - Effects of executed actions are only considered in afterwards
- Goal: mapping states to actions
- Key elements
 - Q-function (for learning)
 - Q-table (knowledge base)

Reference: <https://upload.wikimedia.org>; Einführung in das Reinforcement Learning

Q-Learning: Q-Function



- Based on the idea of Markov Decision Processes (MDPs)

- models the quality of an state-action pair

- $Q : s \times a \rightarrow \mathbb{R}$

- s : State

- a : Action

- Result: Q-value $Q(s, a)$:

$$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * \max_{a'} Q(a', s'))$$

known
information

new
information

- r : short-term reward

- $\max_{a'} Q(a', s')$: estimate of optimal long-term reward

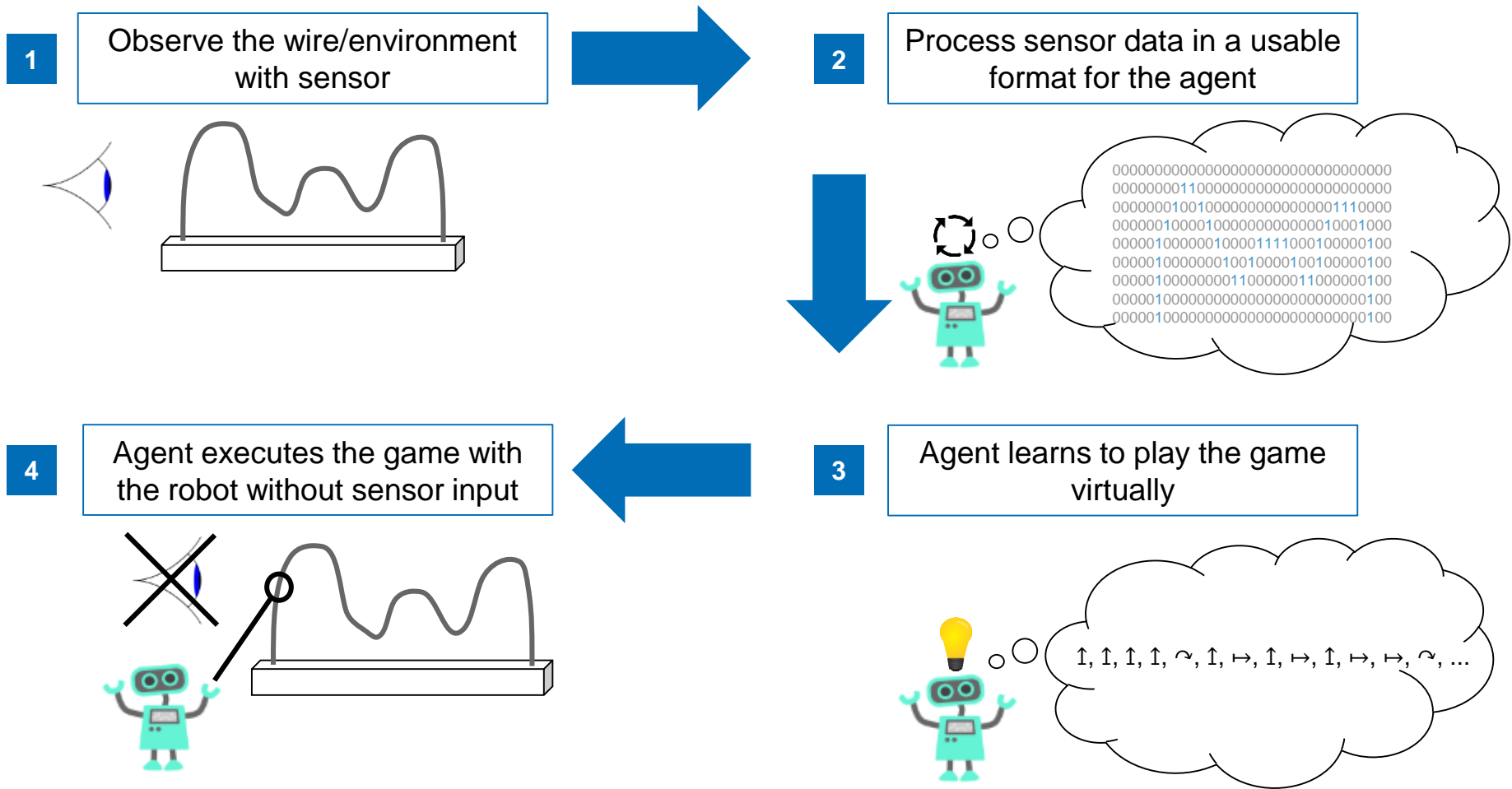
- α : learning rate

- γ : discount factor

Agenda

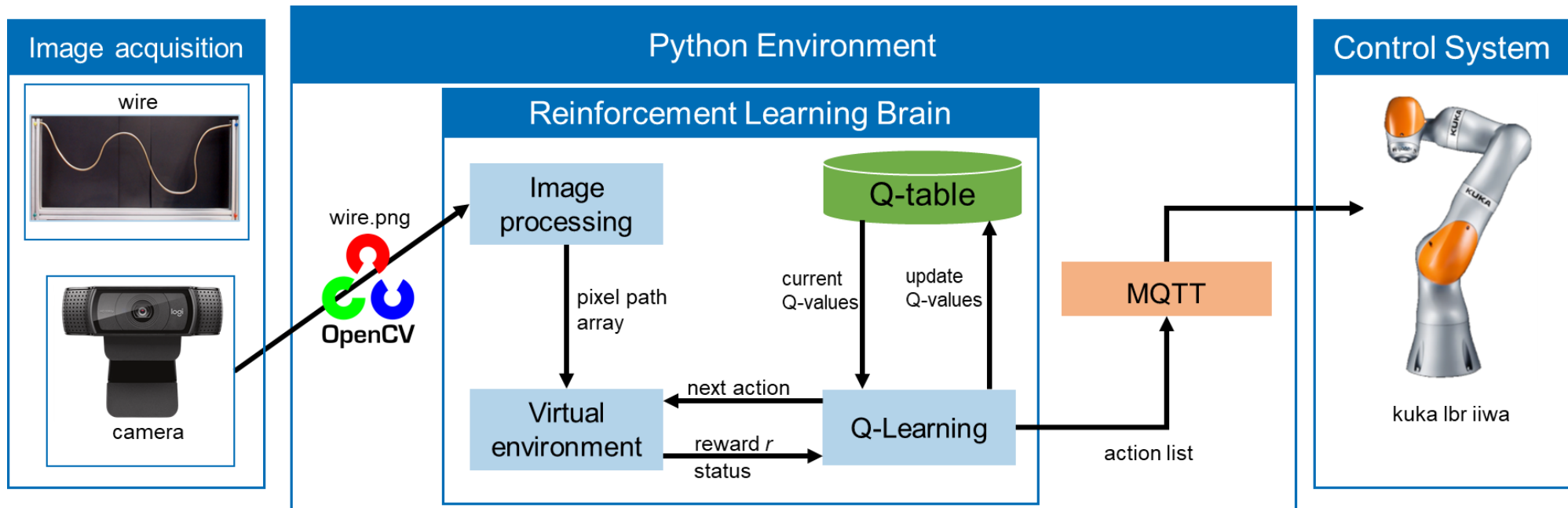
- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Basic Approach



Reference: <https://upload.wikimedia.org>; <https://www.kindpng.com>

Basic Approach



Reference: Logitech; OpenCV, www.robots.com/robots/kuka-lbr-iiwa-14-r820

Agenda

- 1** Use case introduction
- 2** Conventionell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

The flowchart illustrates the image processing steps for path extraction:

- a: Original Image**
- b: Cropping and Rotation by 180°**
- c: Conversion to Grayscale**
- d: Threshold Function** (apply threshold to get binary image)
- e: Average Function (column-wise)** (average column-wise)
- f: Average Function (column-wise + row-wise)** (add average row-wise)
- g: Final Pixel Path** (shortest path (cost function))
- h: Pixel Path Array** (convert in array)

The final output is a binary image showing the shortest path (g) and its corresponding pixel path array (h).

Agenda

- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Virtual Environment (VE)

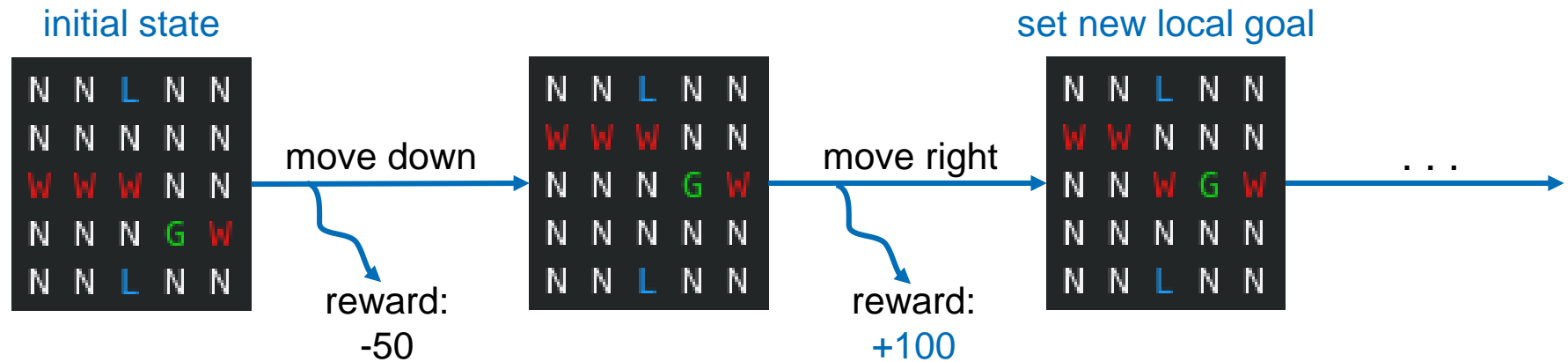


- Simulation of the wire loop game
- Training environment for the Q-learning algorithm
- Enormous time saving when training the algorithm
- Reduced to a 5x5 matrix of the pixel path array (state)
- Limits the perception of the algorithm
- Helps to generalize problem
- TCP is always in the middle (position (3, 3))

$$S = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

N = nothing
W = wire
L = loop
G = goal

Virtual Environment



Reward-Function:

■ -100 for:

- Collision
- 5x5 Matrix out of bounds
- local goal out of bounds

■ -50 for:

- every action/movement

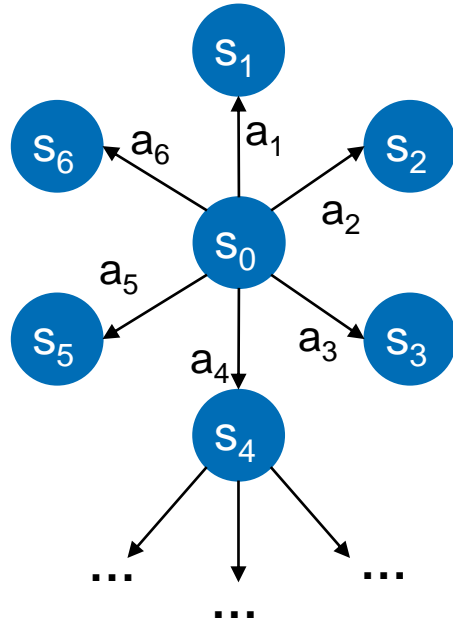
■ +100 for:

- reaching a local goal

Agenda

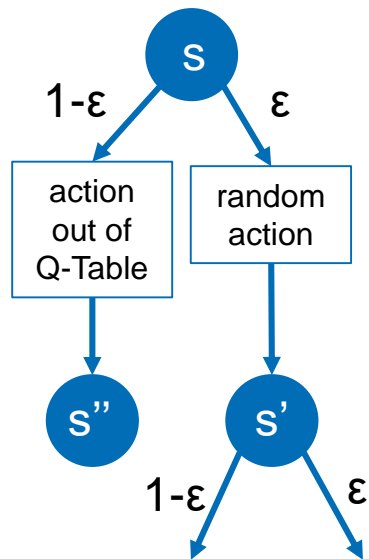
- 1** Use case introduction
- 2** Conventionell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Q-Learning: Q-Function



- $Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * \max_{a'} Q(a', s'))$
- Learning rate $\alpha = 0,3$
 - slow adaption to new information
 - Focus on already acquired knowledge
 - compensates weaknesses in the reward-function
- Discount-factor $\gamma = 0,9$
 - Aims at a long-term strategy
 - solve game with a minimum number of actions

Q-Learning: Algorithmus



for-loop: (to set number of learning episodes)

get initial state s from VE

while-loop:

select action a :

- with probability ϵ : random action a
- with probability $1-\epsilon$: action a out of Q-Table

perform action a in VE

- get reward r and new state s' from VE

update q-value (q-function)

reduce ϵ (every episode)

- **for**-loop
 - episode ends when the game fails (collision...)
- **while**-loop
 - is repeated until game fails (collision...)

Q-Learning: Q-Table


$$S = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

- Look-up-table
- store all achieved states with corresponding Q-values
- States are assigned a hash value
 - $\text{hash}(S) = 00200\ 00000\ 11110\ 00003\ 00200 + \text{id (last action)}$
- Q-Values :
 - initialized with random values from the interval $[0; 1]$
 - are adapted in the Q-learning algorithm

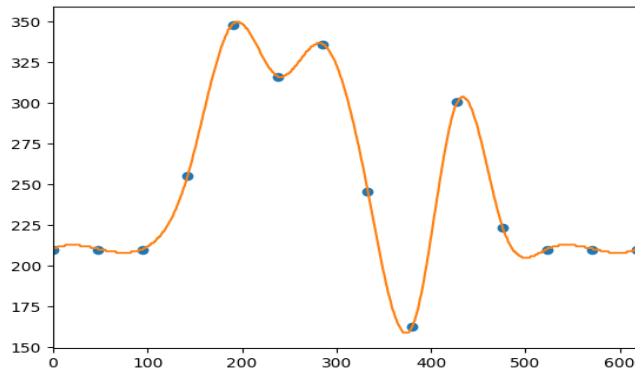
Q-Learning: Q-Table

		actions					
		up	right	down	left	rotate cw	rotate ccw
states	0021300...	-0.6308...	-0.6308...	-0.6395...	39.9437...	-0.5259...	-0.5716...
	0001000...						
	0100001...						
	1000010...						
	0021011...						
	...						

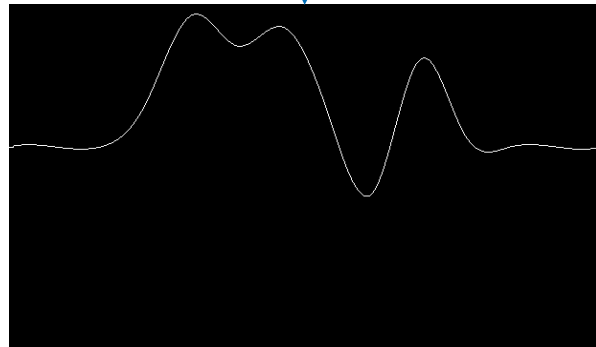
resulting
action



Q-Learning: Generation of training data



spline interpolation



resulting pixel path

■ two options:

- manual: with pictures of wire
→ time consuming
- automatic: artificially created pixel paths
→ efficient, flexible

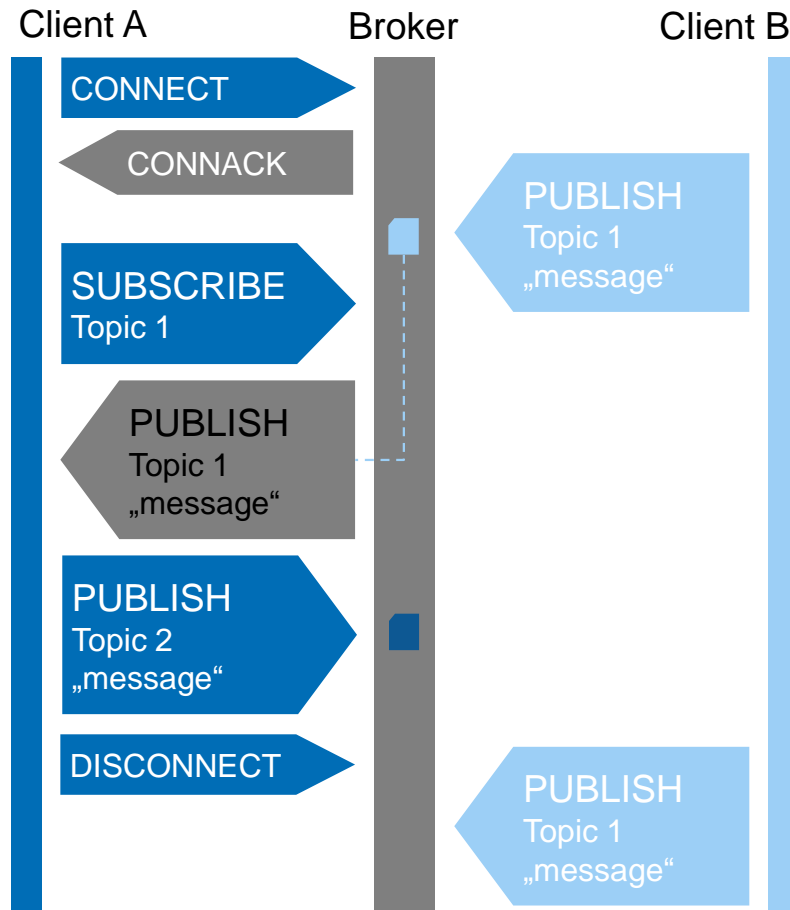
■ Automatic :

- Cartesian coordinate system with 14 points
- cubic spline through all points
- convert spline in a pixel path image

Agenda

- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Control concept



■ MQTT (Message Queuing Telemetry Transport)

- network protocol for M2M-communication
- client-server-principle
- server (broker) manages message traffic
- sender and receiver connect to broker
- topic declares subject of a message

■ Properties:

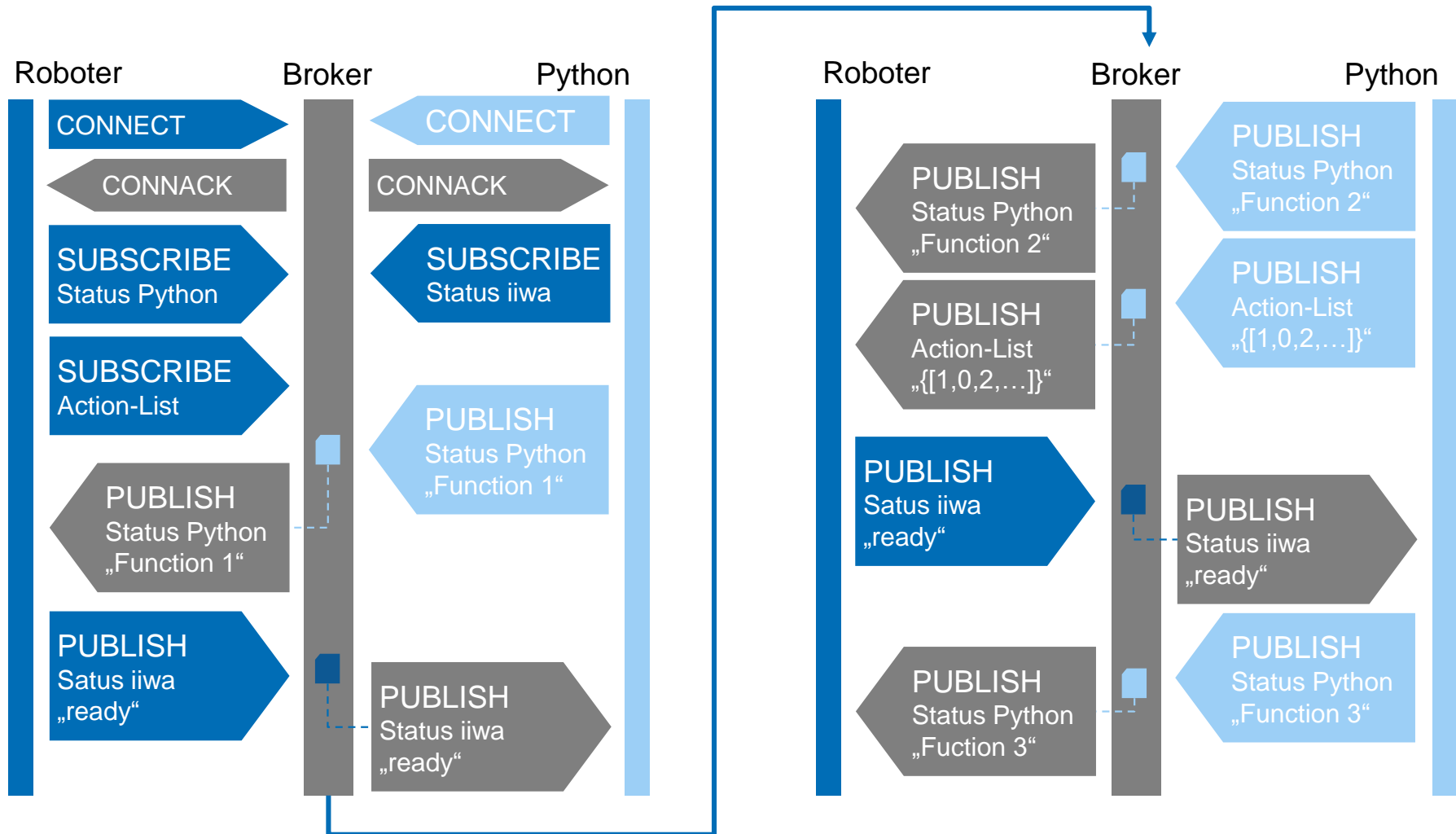
- QoS (Quality of Service): (0, 1, 2)
- Last Will
- Retained Messages

→ Two programs necessary

- Python-programm on the PC
- Roboter-application (Java)

Quellen: <http://mqtt.org/>, <https://www.informatik-aktuell.de>, angelehnt an <https://de.wikipedia.org/wiki/MQTT>

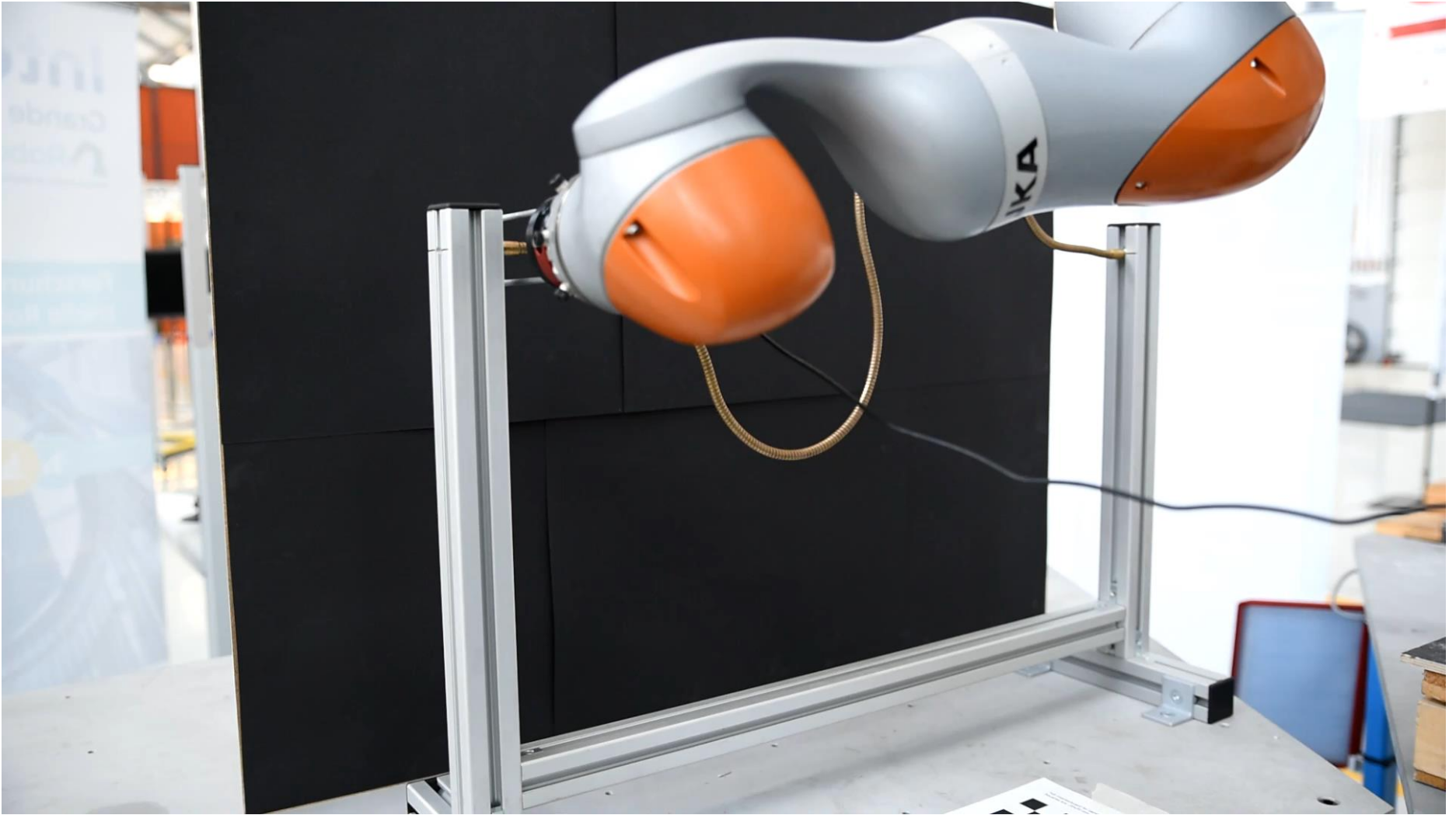
MQTT statt ROS



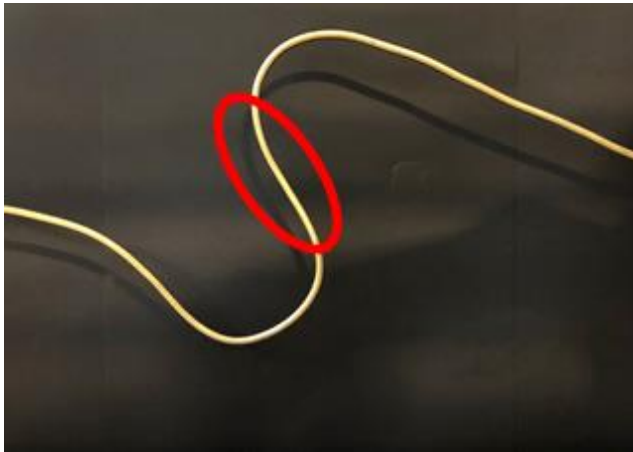
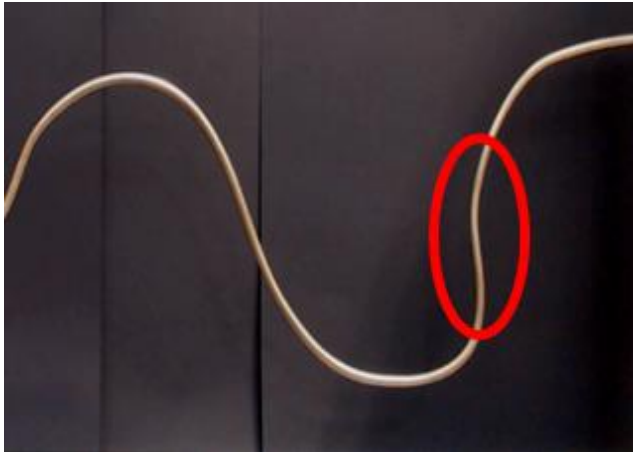
Agenda

- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Implementation



Evaluation



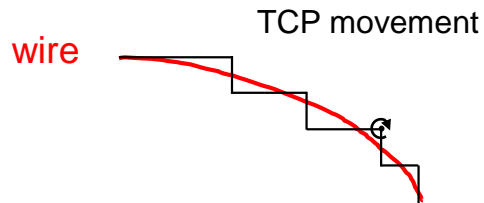
- System is robust and reliable
- Usually no training phase necessary for playing new wire configurations
- Robot passes even difficult sections
- Problems/weaknesses:
 - changing lighting conditions affect image processing
 - difficulties when driving through tight curves
 - discontinuous movement of the robot
 - rotation interrupts translational movements

Agenda

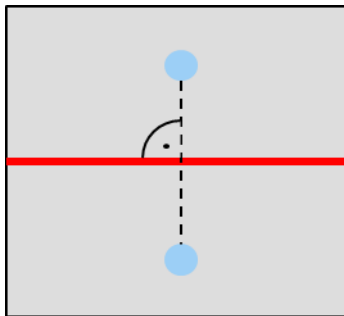
- 1** Use case introduction
- 2** Conventiell path planning
- 3** Spline interpolation: manual selection of support points
- 4** Motivation of AI-based path planning and processes
- 5** Basics: Reinforcement Learning and Q-Learning
- 6** Approach and Development of an AI based process
 - 6.1** Image processing
 - 6.2** Virtual Environment
 - 6.3** Q-learning
 - 6.4** Robot Control
 - 6.5** Implementation and Evaluation
 - 6.6** Post Processing
- 7** Conclusion and Outlook

Improvements

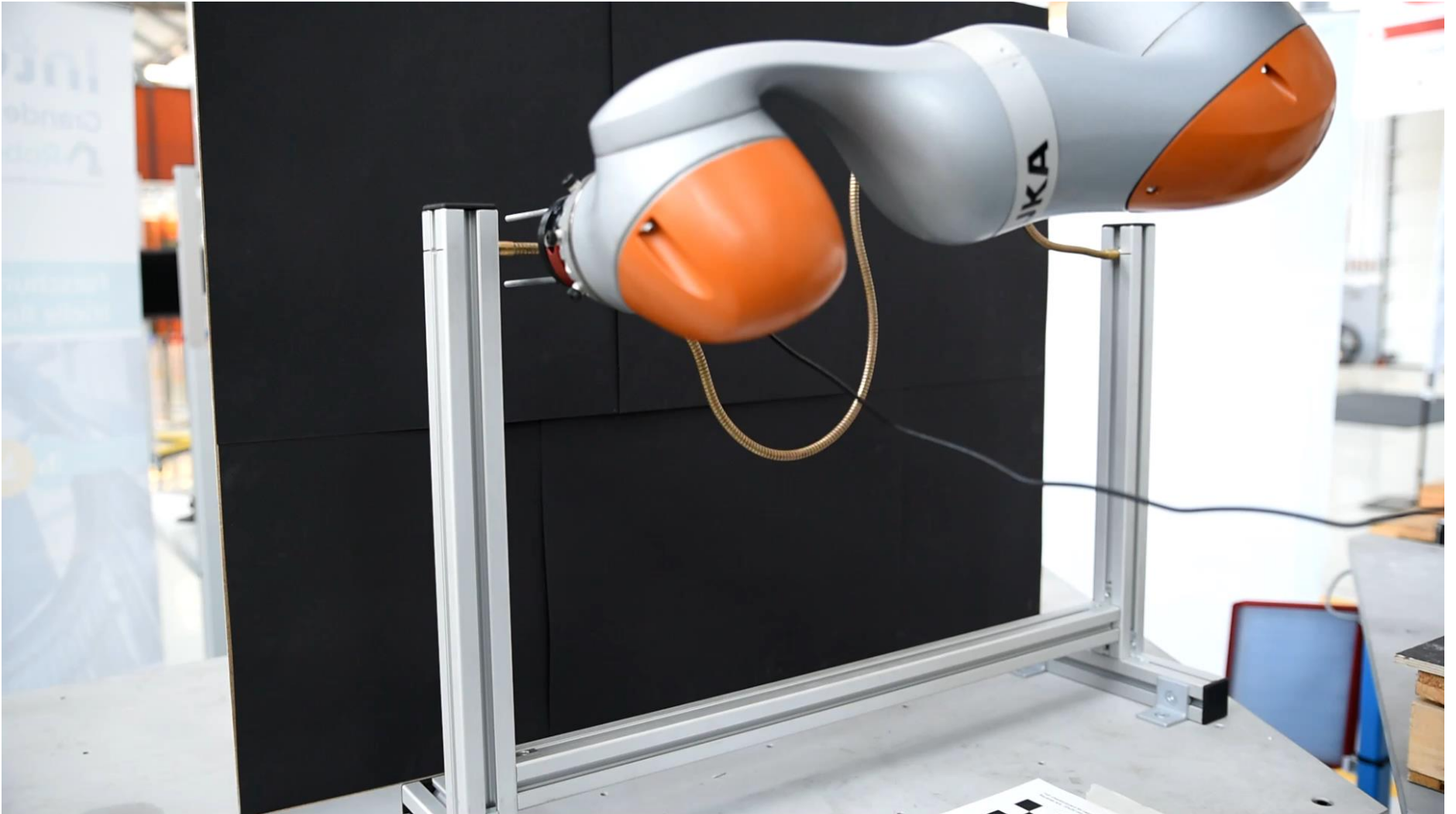
- Postprocessing of the action list:
 - distribute the rotation to the translational movements
 - no more interruption for rotations
 - filter the action list to get smoother movement
 - delete unnecessary waypoints
 - moving average



- Expand the virtual environment / reward-function
 - evaluation of the angle between loop (electrodes) and wire
 - release positive reward for actions that lead to 90° angle



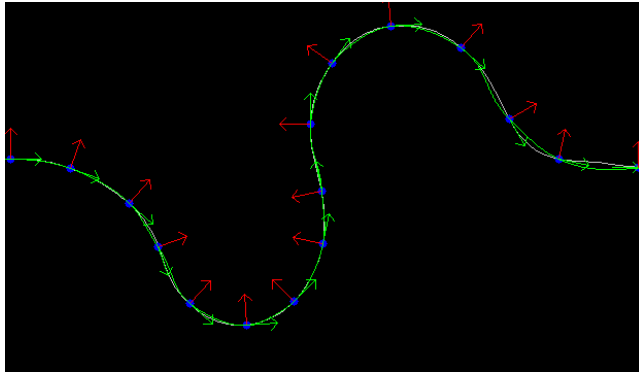
Improvements



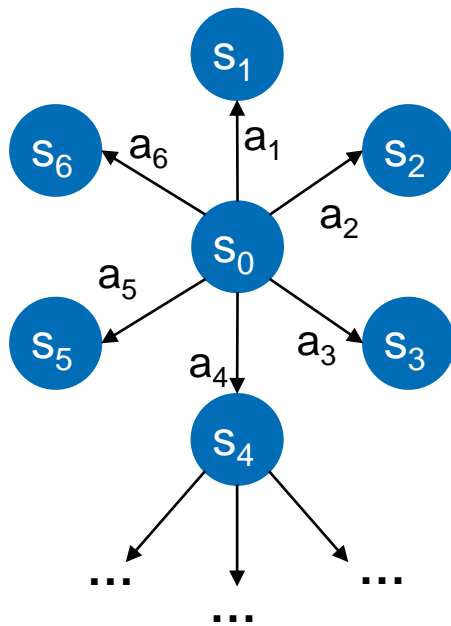
Agenda

- 1 Use case introduction
- 2 Conventionell path planning
- 3 Spline interpolation: manual selection of support points
- 4 Motivation of AI-based path planning and processes
- 5 Basics: Reinforcement Learning and Q-Learning
- 6 Approach and Development of an AI based process
- 7 Conclusion and Outlook

Conclusion and Outlook



- Spline interpolation with manual selection of the support points
 - interactive method, easy to use
 - visualisation makes the process transparent
 - path can be executed with constant velocity
 - Outlook: automatize the process of support point selection



- Path planning based on reinforcement learning:
 - new approach with some potential
 - modeling the VE is very complex
 - result depends on the quality of the preliminary work and the VE
 - Q-learning alone is not powerful enough (Deep Q-learning)
 - Outlook: improve current system, evaluate potential for future projects