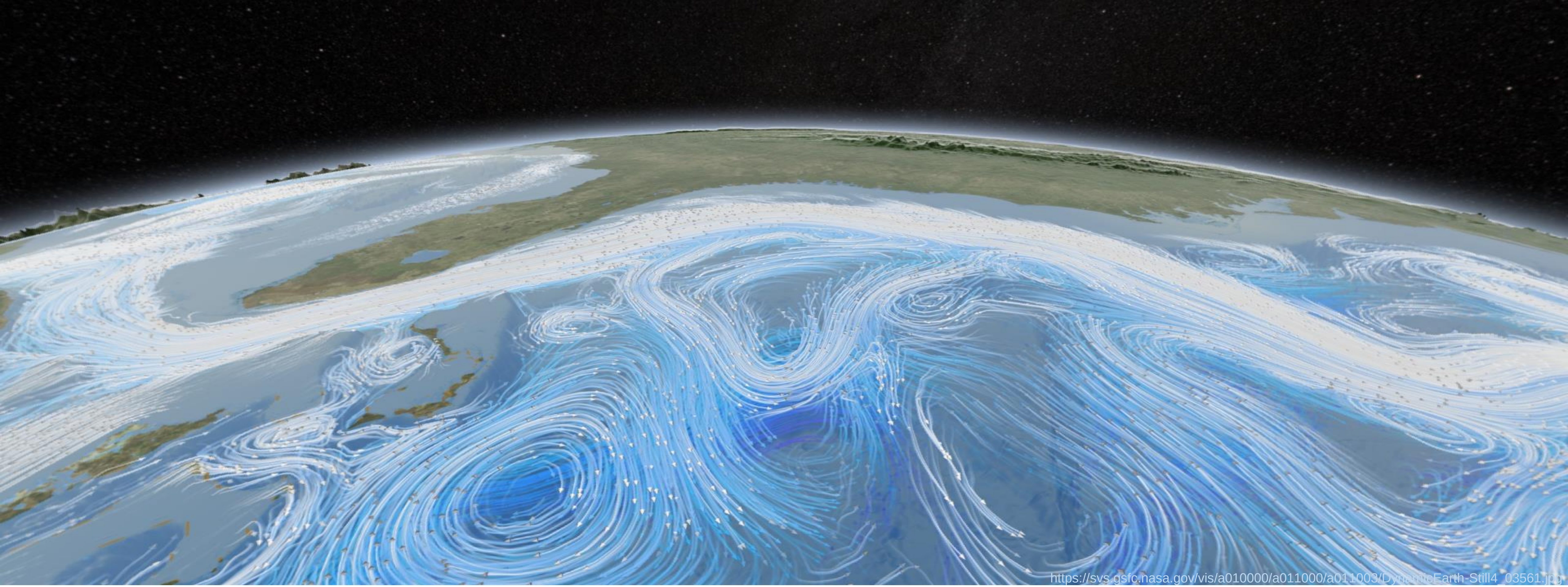


Intelligent Robotics

Tuning Cost Functions

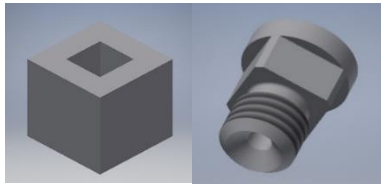
Philipp Ennen, M.Sc.

Learning Motor Skills for Assembly Tasks

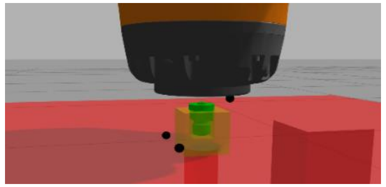
Motor skills for assembly tasks

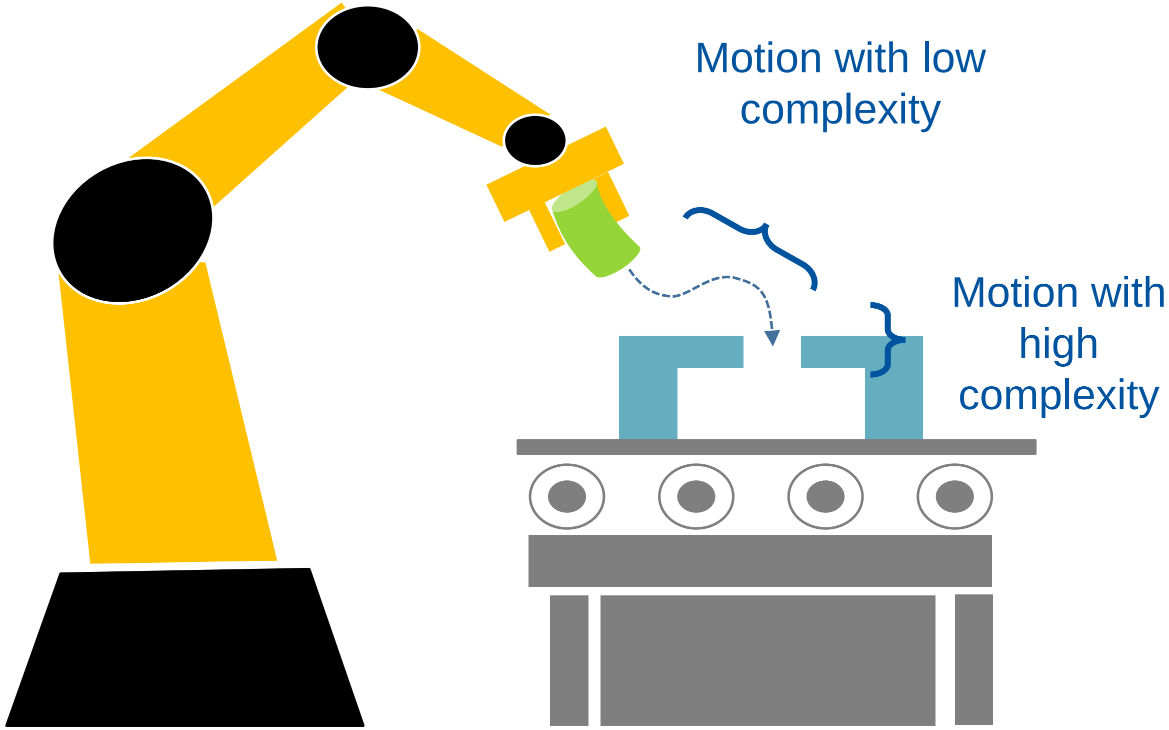
- Assembly tasks
  - Low joining tolerances (<0.5 mm)
  - Careful joining of parts
  - Wanted collision at the destination
  - Complexity of movement near the target is increased compared to the start position
  - Reaching the final target position is more important than a cost-effective path

Parts



Goal position





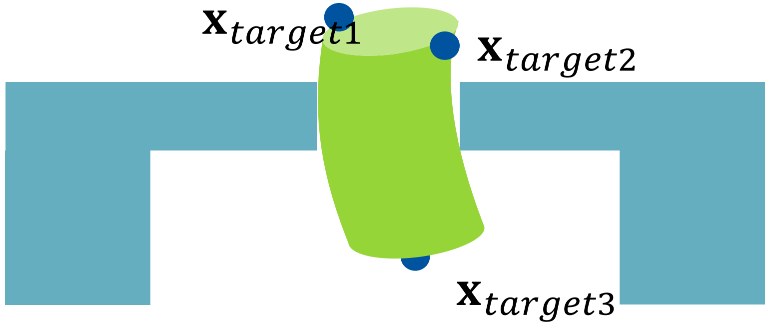
Motor skills for assembly tasks – Goal Description

- Goal description for assembly tasks
  - Minimal torques
  - Minial distance to goal state
  - Reach the final position

- Individual weighting of the action costs for each robot joint

- Description of the target position via virtual points at the destination
  - Three points := position and orientation is fixed
  - One point := only position is fixed

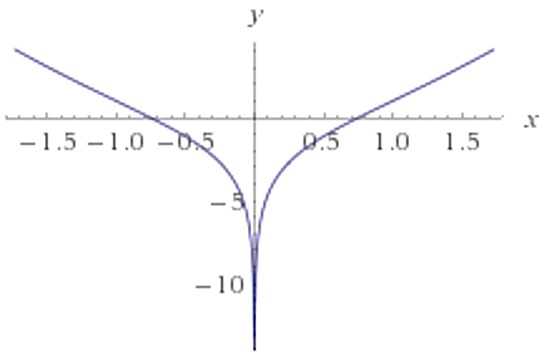
$$J = \underbrace{w_u \sum_0^N l(u_t)}_{\text{action costs}} + \overbrace{w_x \sum_0^N l(x_t)}^{\text{state costs}} + \underbrace{w_{xf}\, l_f(x_N)}_{\text{Final state costs}}$$

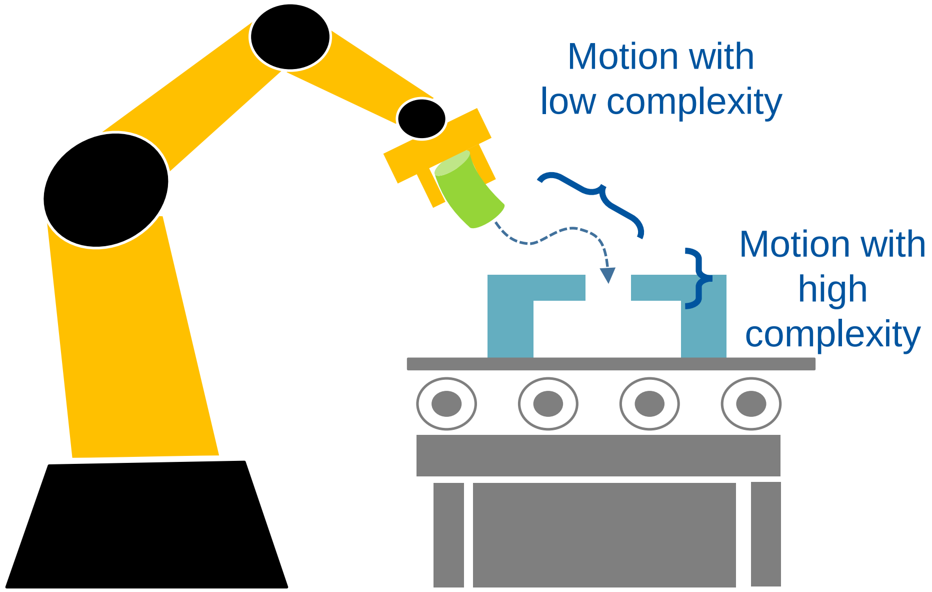


Motor skills for assembly tasks – Goal Description

- Calculation of state costs via
  - Quadratical term
  - Logarithmic term

Disproportionate weighting of the distance change in the target range





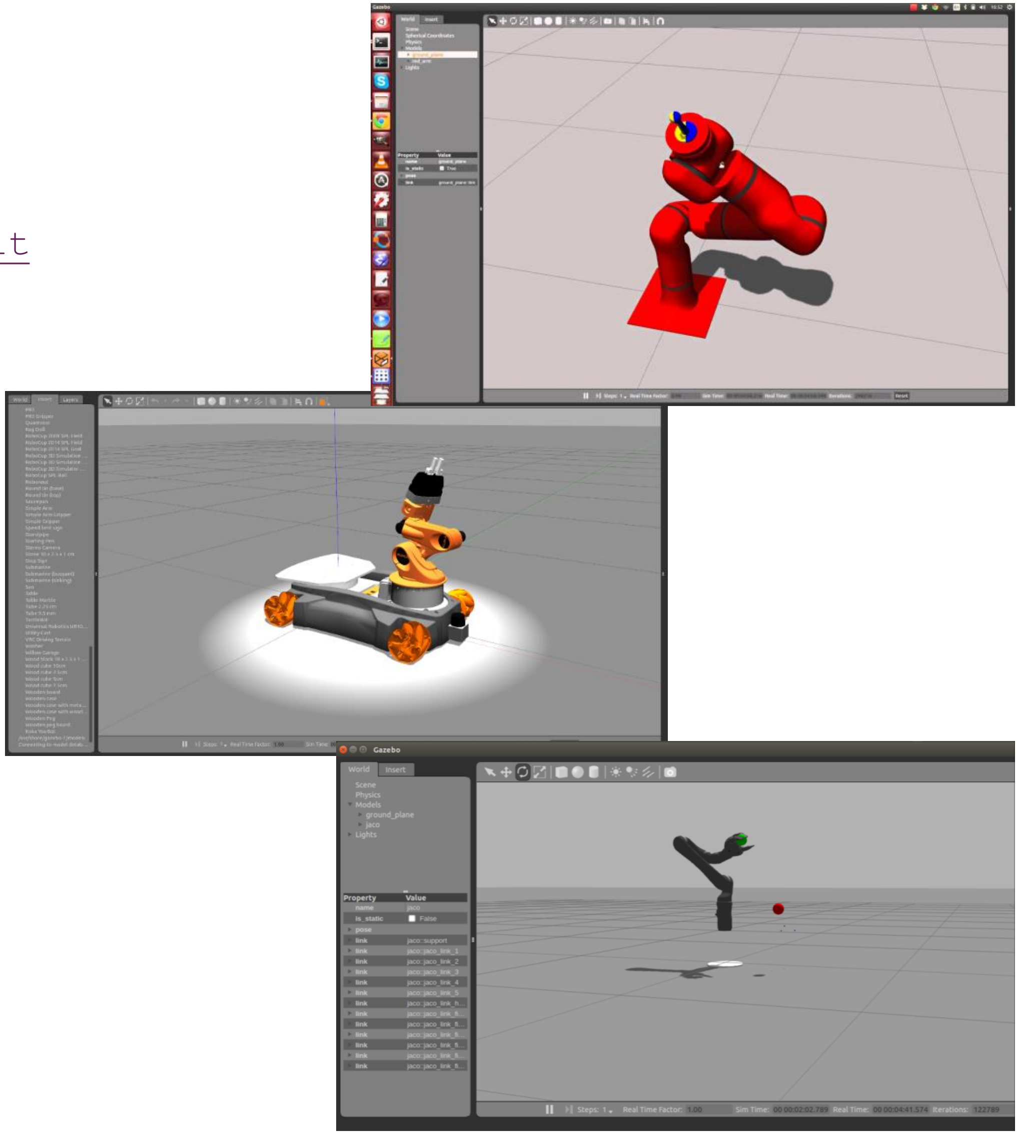
$$l(d) = \overbrace{l_1 d^2}^{\text{Quadratical term}} + \underbrace{l_2 \log(\, d^2 + \alpha)}_{\text{Logarithmic term}}$$

Task 3 – Simulation Environment

Use Gazebo simulation

- Install simulation packages
  - `cd catkin_ws/src`
  - `git clone https://github.com/philippente/jaco_gazebo.git`
  - `git clone https://github.com/philippente/jaco_description.git`
  - `cd ..`
  - `catkin_make`

- Start Gazebo Simulator:
  - `roslaunch jaco_gazebo gps_jaco_gazebo.launch`



Task 3 – Installation procedure

Download source code (do it in your home directory: `cd ~`):
- `git clone https://github.com/philippente/task3_costs.git`

Edit .bashrc to set environment variables:
- `gedit ~/.bashrc`

At the end of file, the lines should look like this:
- `source /opt/ros/kinetic/setup.bash`
- `source /home/<USERNAME>/catkin_ws/devel/setup.bash`
- `export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/opt/ros/kinetic/share:/opt/ros/kinetic/stacks:/home/<USERNAME>/task3_costs:/home/<USERNAME>/task3_costs/src/gps_agent_pkg`
- `export GAZEBO_MODEL_PATH=/home/<USERNAME>/catkin_ws/src/jaco_gazebo/models:${GAZEBO_MODEL_PATH}`
- `export GAZEBO_RESOURCE_PATH=/home/<USERNAME>/catkin_ws/src/jaco_gazebo/models:${GAZEBO_RESOURCE_PATH}`

Check if the blue part of the source folder and ROS_PACKAGE_PATH is correct!

Then save it and close it. Source the .bashrc (load the environment variables):
- `source ~/.bashrc`

Now, compile some stuff:
- `cd task3_costs`
- `sh compile_proto.sh`
- `catkin_make`

Task 3

Tune cost functions!

Task 1: Let the robot learn to reach the orange ball

- Start the learning procedure
  - `cd task3_costs`
  - `python python/gps/gps_main.py jaco_example`
- How fast does the robot learn?

Task 2: Adjust the cost parameters

- Open following file in PyCharm:
  - `task3_costs/experiments/jaco_example/hyperparams.py`
- Adjust the cost parameters! (cf. code →
  - change the red parameters (`wu, l1, l2, alpha, …`)
- Start the learning procedure
  - What influences of the red parameters can you observe?

```
torque_cost = {
    'type': CostAction,
    'wu': 5e-3 / PR2_GAINS,
}

fk_cost1 = {
   'type': CostFK,
    # Target end effector is subtracted out of EE_POINTS in ROS so goal
    # is 0.
    'target_end_effector': np.zeros(3 * EE_POINTS.shape[0]),
    'wp': np.ones(SENSOR_DIMS[END_EFFECTOR_POINTS]),
    'l1': 0.1,
    'l2': 10.0,
    'alpha': 1e-6,
    'experiment_ID': common['experiment_ID'],
    'dir':common['cost_log_dir'],
}

fk_cost2 = {
    'type': CostFK,
    'ramp_option': RAMP_FINAL_ONLY,
    'target_end_effector': fk_cost1['target_end_effector'],
    'wp': fk_cost1['wp'],
    'l1': 1.0,
    'l2': 15.0,
    'alpha': 1e-6,
    'wp_final_multiplier': 25.0,
    'experiment_ID': common['experiment_ID'],
    'dir':common['cost_log_dir'],
}

algorithm['cost'] = {
    'type': CostSum,
    'costs': [torque_cost, fk_cost1, fk_cost2],
    'weights': [1.0, 1.0, 1.0],
}
```


Intelligent Robotics | IMA/ZLW & IfU
01.03.2018 | www.ima-zlw-ifu.rwth-aachen.de

Tasks for today and tomorrow

- Task 1:
  - Implement an LQR Backward and Forward pass
  - Try to understand it!
  - Test it with our test method

- Task 2:
  - Implement linearization of the dynamic model
  - Try to understand it!
  - Test it with our test-method
  - Test it on the Box2D Scenario

- Task 3:
  - Test it with Kinova Jaco 2 in simulation
  - Adjust cost function