Boeing Research & Technology

Dreaming
Collaborating
Innovating
Exploring
Trailblazing

Using the Metaverse in Metrology - Leveraging a general-purpose 3D physics engine for sensor data interpretation

Tobias Weber

Lead Engineer - Advanced Production Systems
Material and Manufacturing Technology
Boeing Research and Technology - Europe

October 25, 2024

Producing
Leading
Creating
Researching
Analyzing

Motivation

Images: https://www.mabi-robotic.com, https://broetje-automation.de/, www.boeing.com

- Industrial robots are a great opportunity for aerospace applications, as they combine reasonable costs, large work-volume and high flexibility
- The tight tolerances required in Aerospace and the high variability of the products pose a challenge.
- For certain processes, i.e. laser surface preparation and welding or machining we need to achieve 20 μm path tracking accuracy with a robot manipulator.

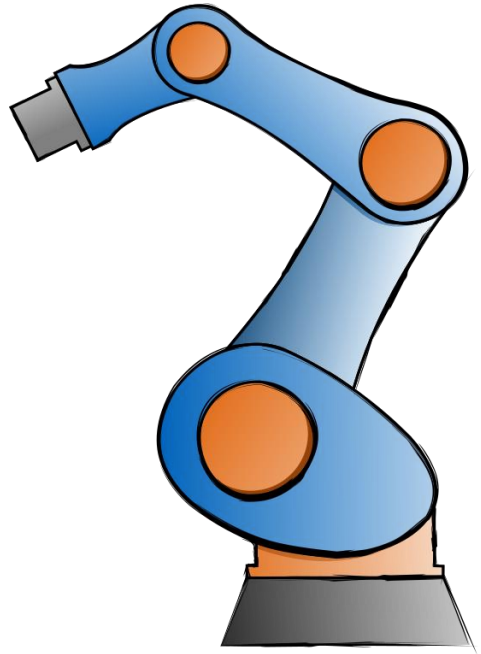
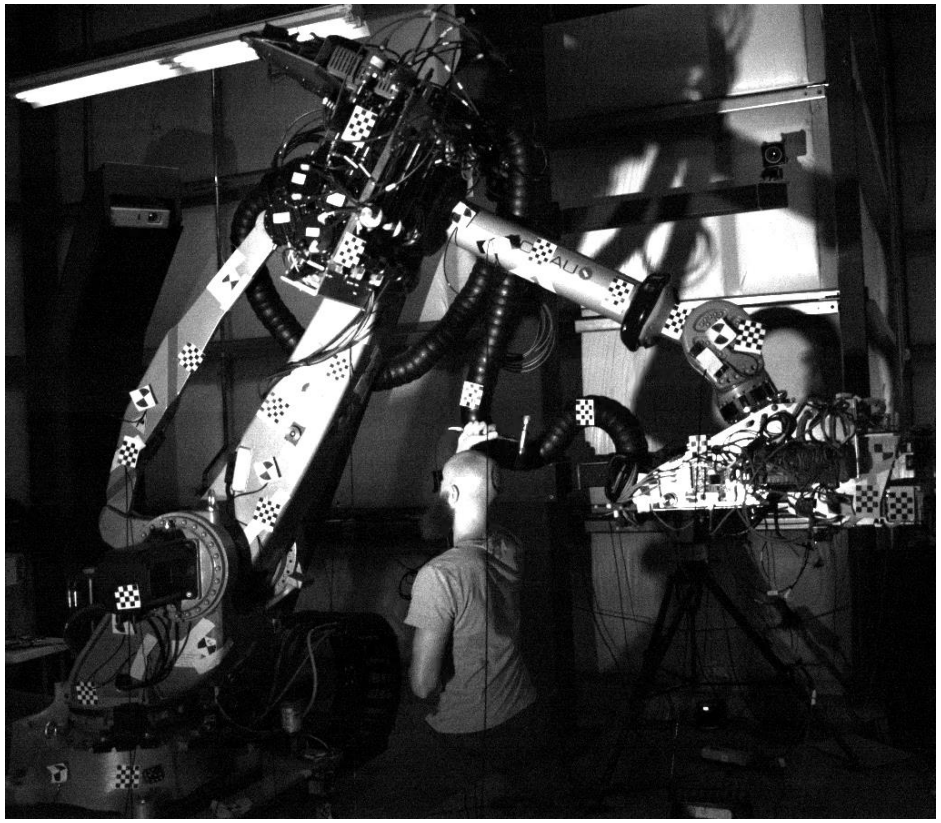
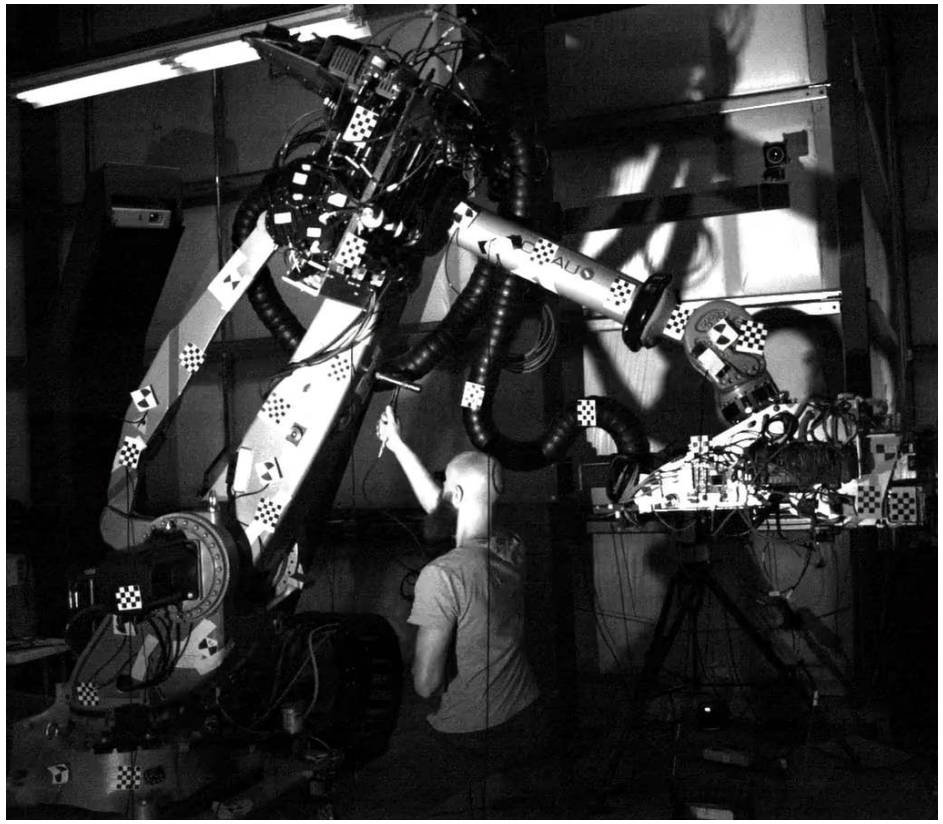


Image: Dr. Tobias Berninger, External Stabilization of Robot Manipulators

Measure Robot Dynamics (Motion Magnification)



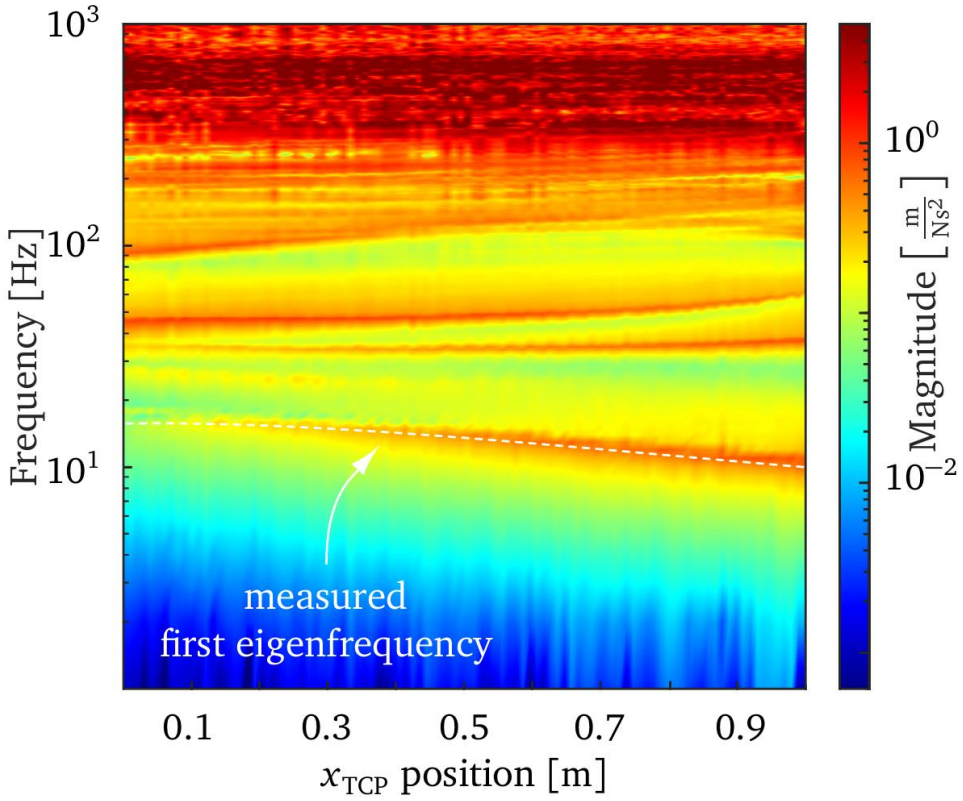
- System instrumented with accelerometers
- System instrumented with vision markers
- 6 impact directions using a hammer
- High-speed camera 8000FPS



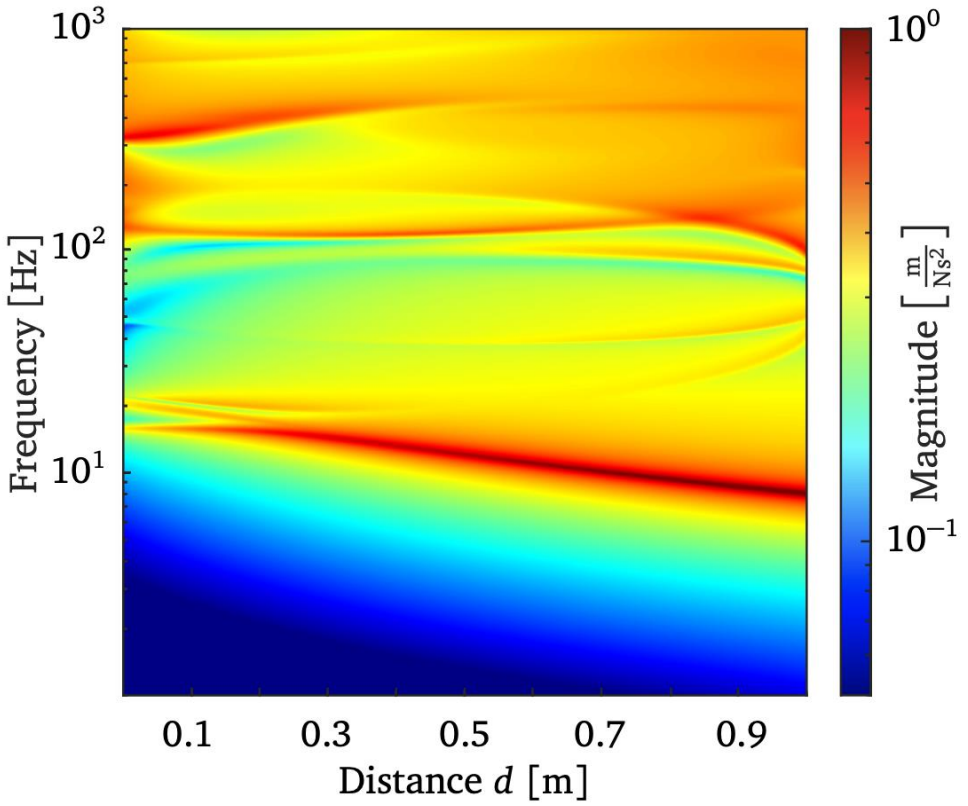
- Amplifying motion by factor of 200
- Filtering Out high frequency noise
- Filtering out static displacement
- 10 Modes <100Hz Identified: 5.2, 6.9,12, 22, 26, 33, 40, 59, 75, 85 Hz

Pose Dependent Eigenfrequencies (UR10)

measured dynamic behavior



simulated dynamic behavior

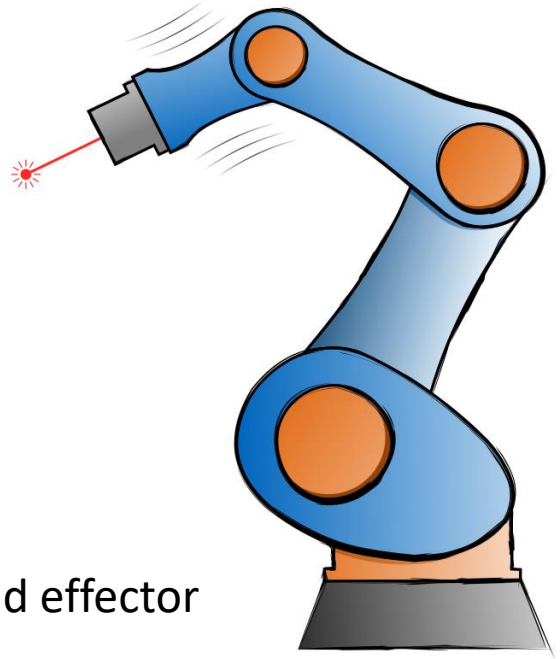


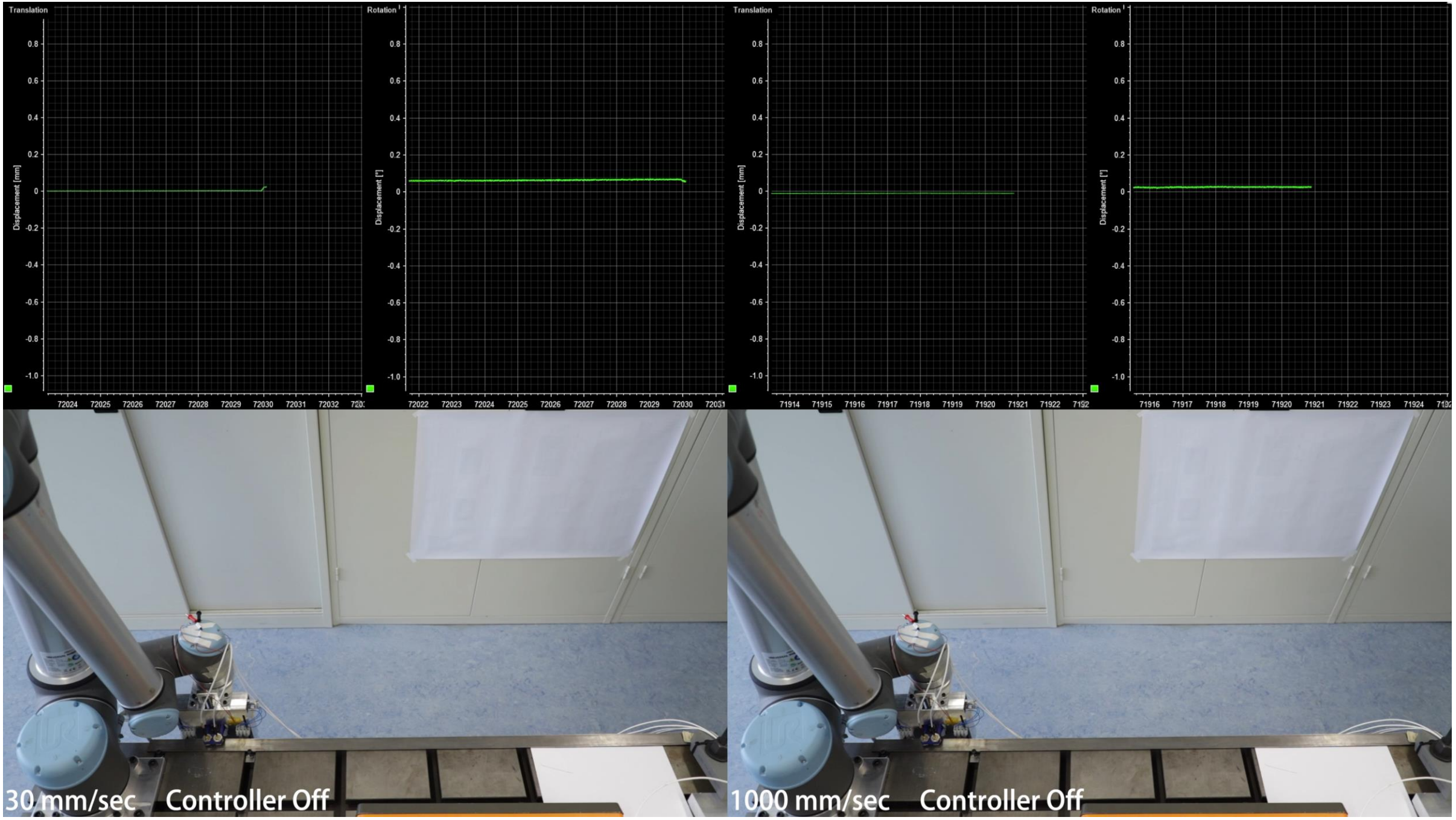
Simulation captures well first eigenfrequencies and modes, including their shift for different poses. But is not able to capture higher modes.

For our accuracy criteria a model-based, open loop control is not suitable!

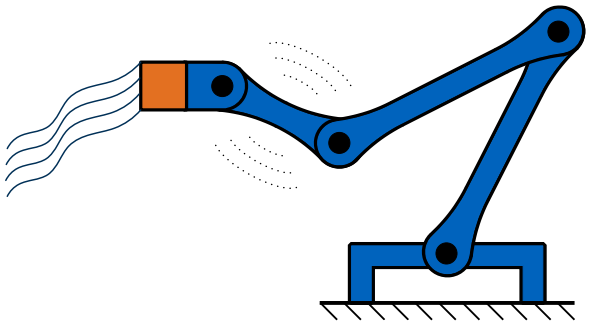
Robot Accuracy: Common Approaches

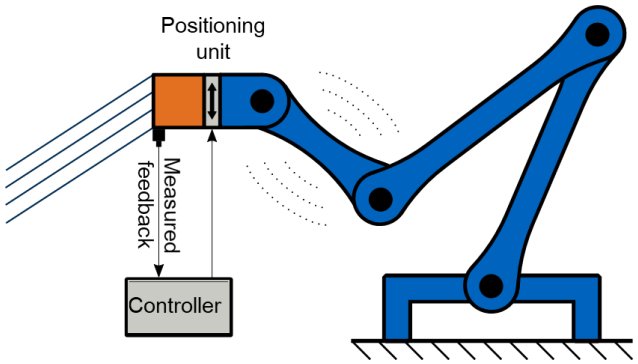
- Robot calibration methods
  - ➡ Usually ‘only’ improves pose accuracy by ~ 50% - 70%.
  - ➡ Smaller effect on path accuracy and dynamic error
- Improve robot control methods
  - ➡ Challenging non-collocated control problem that needs very accurate models
- Use additional sensors to improve arm-side control
  - ➡ Arm-side torque control bandwidth usually 10 – 20 Hz
- Use additional sensors for active (closed-loop) stabilization of the end effector
  - ➡ It becomes a sensor problem

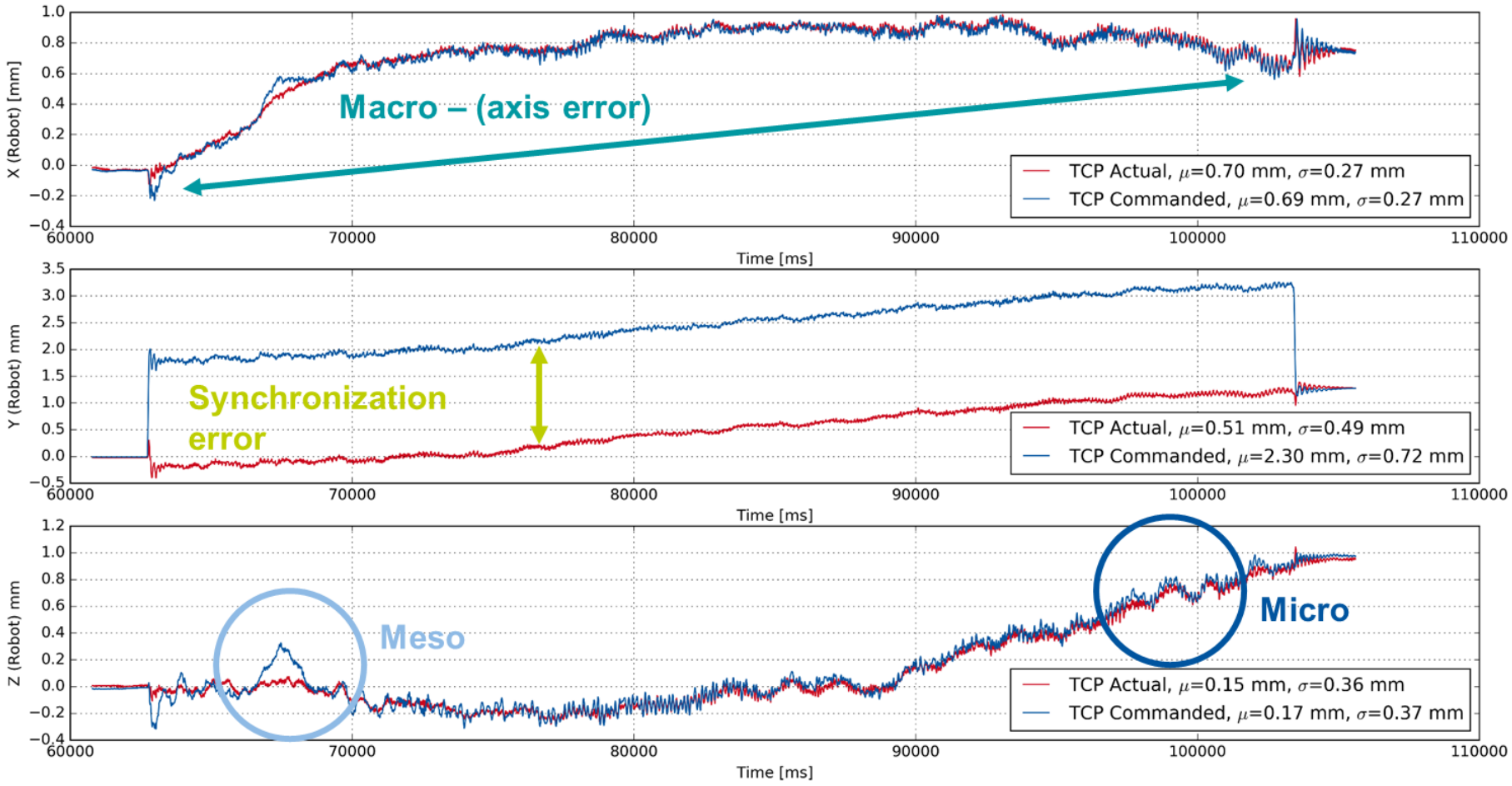




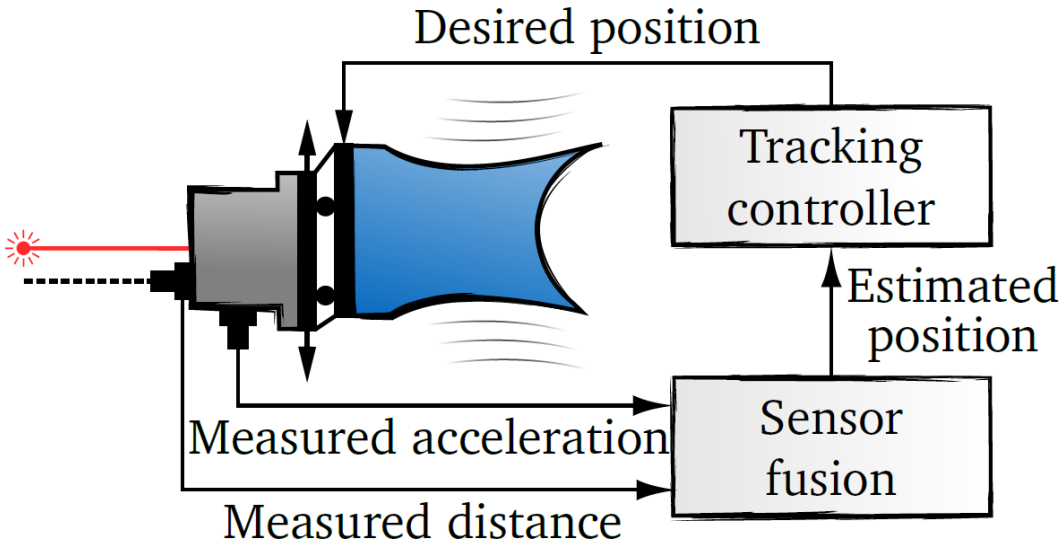
Trajectory misalignment on industrial robots

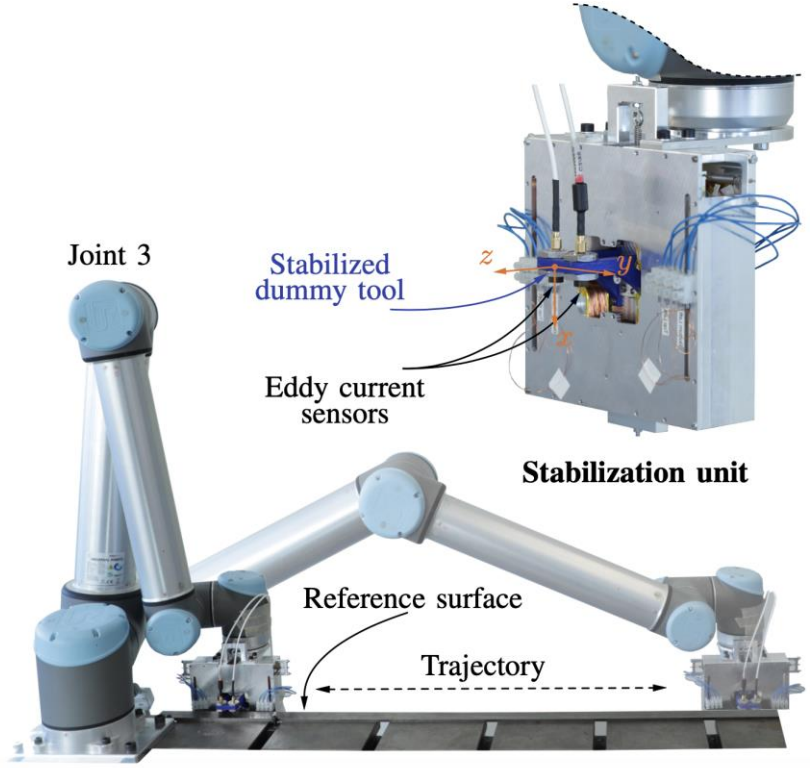


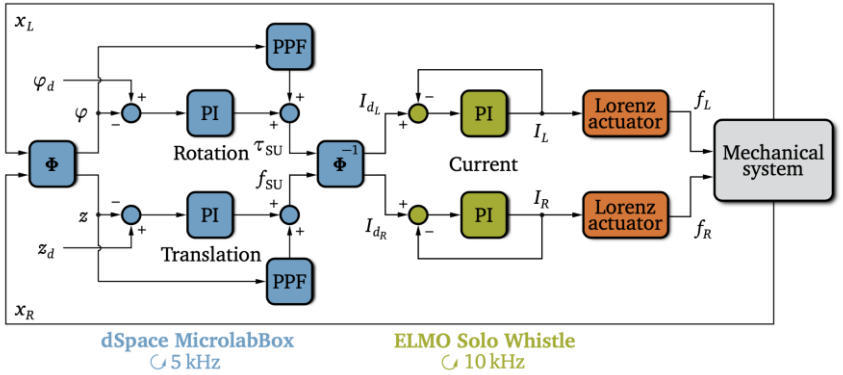


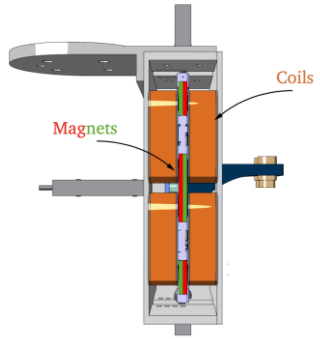


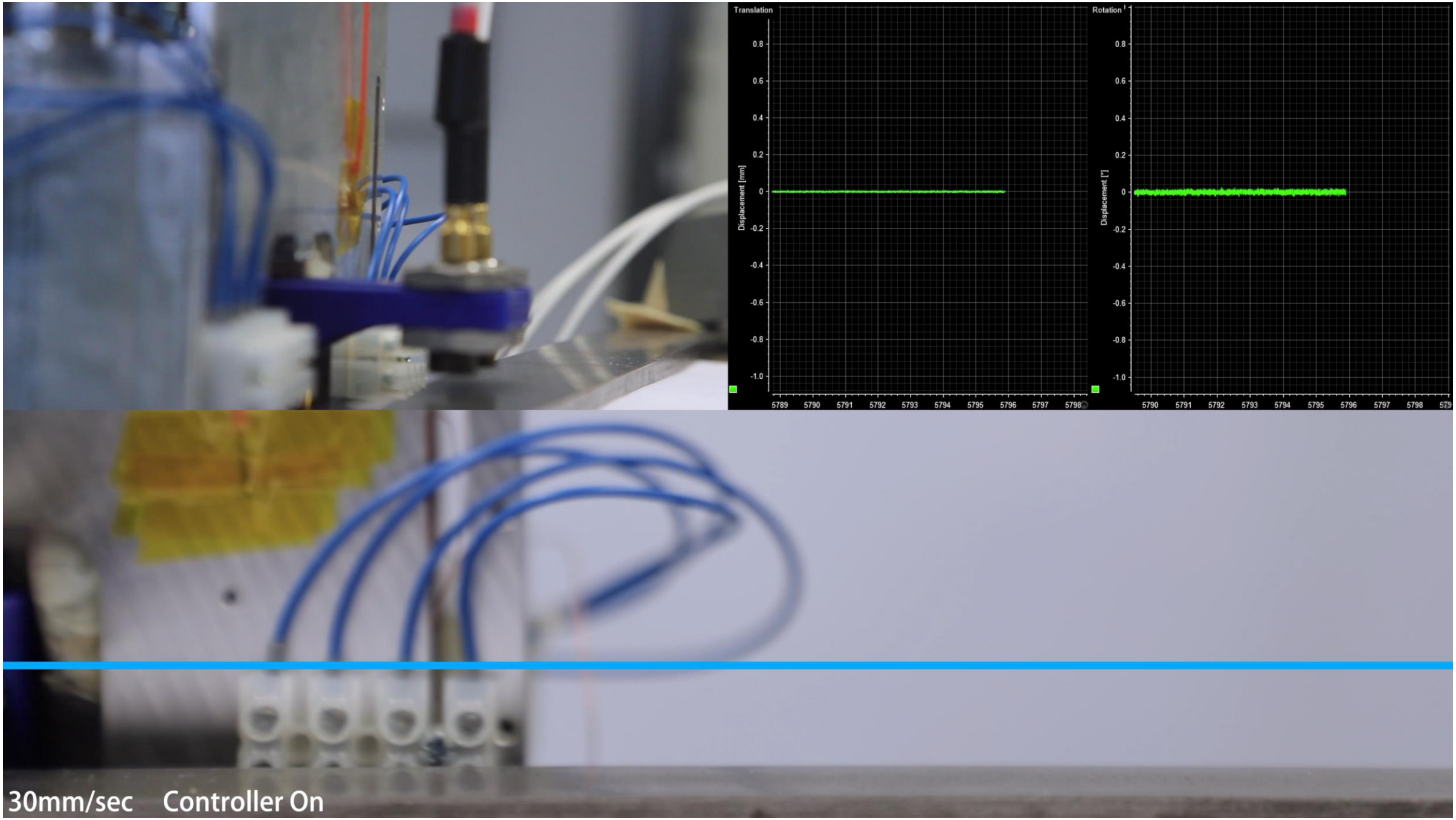
Closed Loop Stabilization of Robots, Example



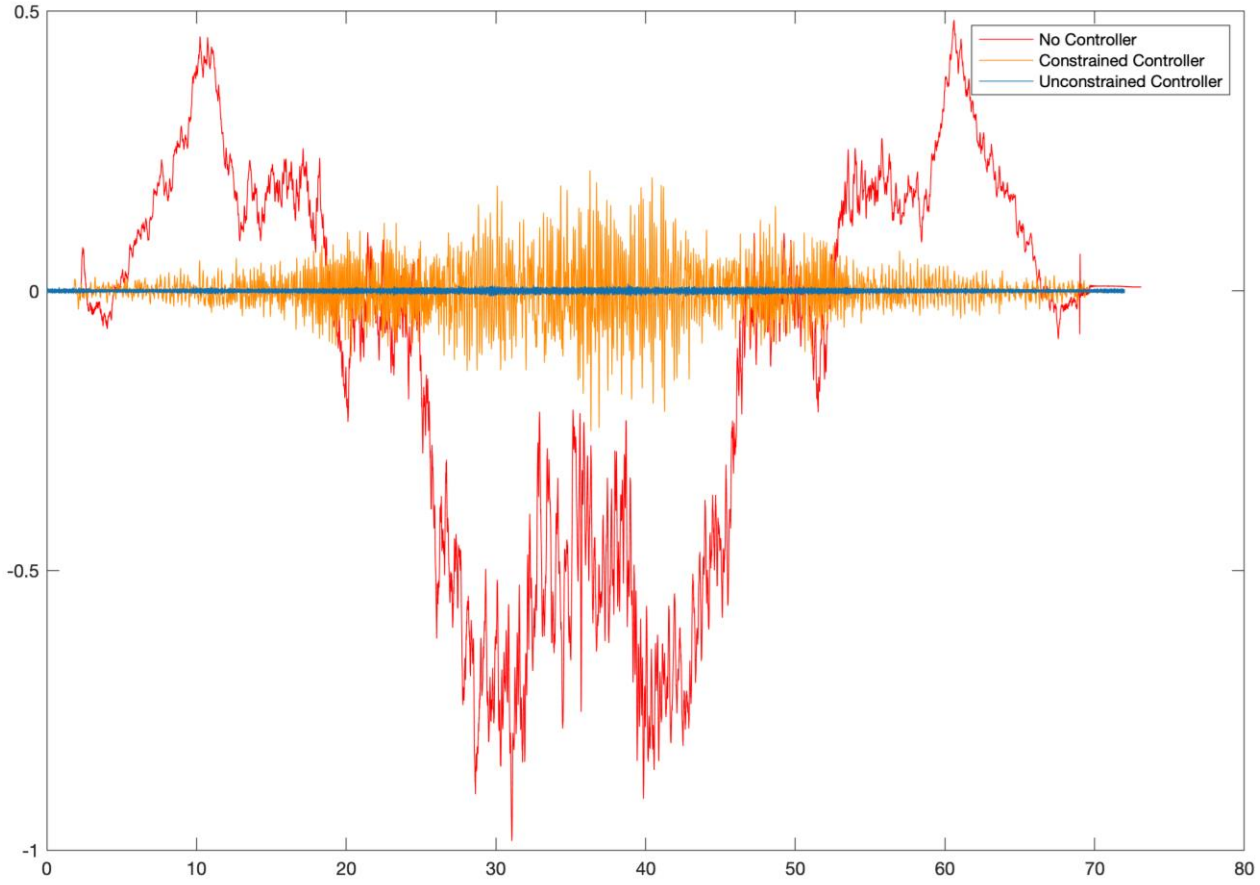


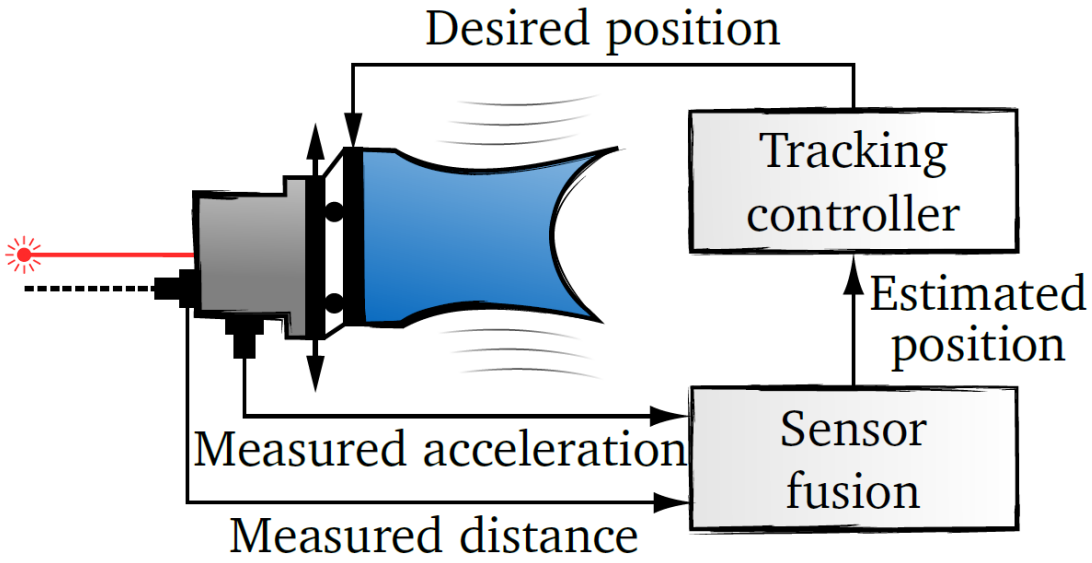






Sensor System Requirements



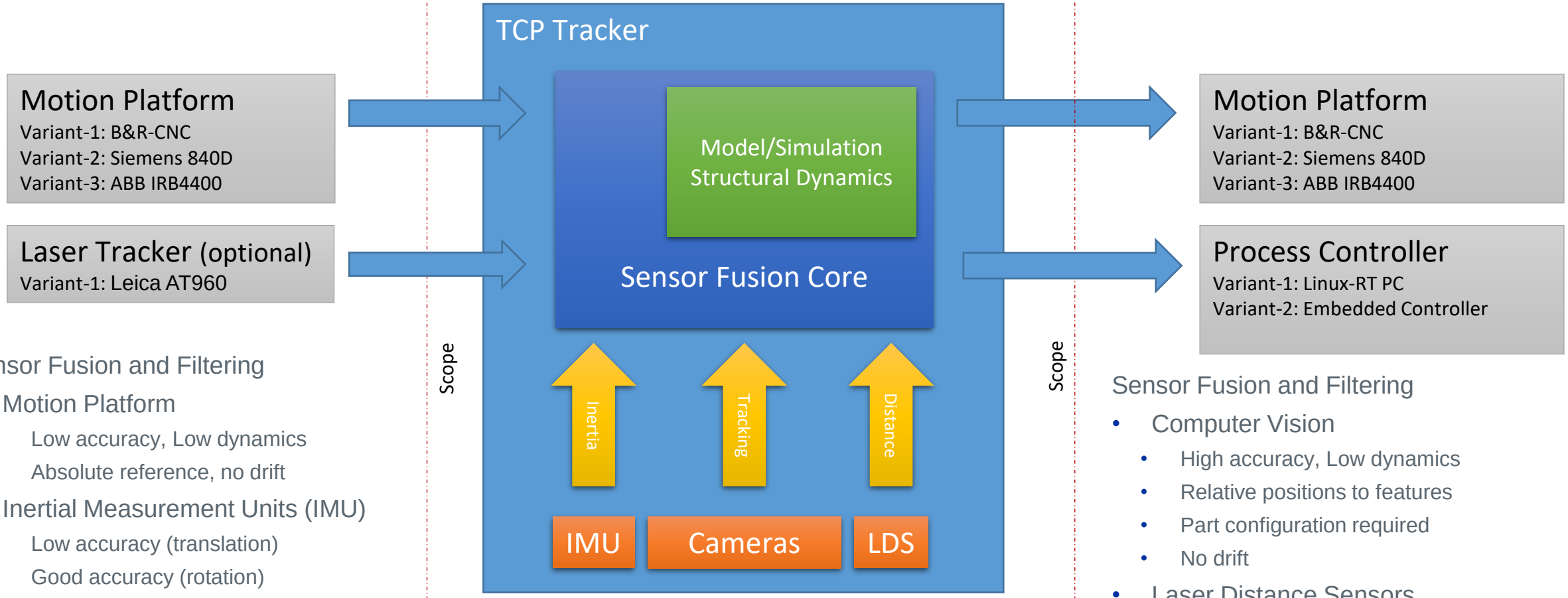


Parameter	Unconstraint	Constraint
Sensor Delay	0.8 ms	15 ms
Sensor Rate	3.5 kHz	0.5 kHz
Position Controller Rate	3.5 kHz	0.5 kHz
Current Controller Rate	10 kHz	8 kHz

TCP Tracking requirements: +/-20µm, +/- 0.2°,
sampling rate 1kHz, latency 1ms

TCP: Tool Center Point

Multi Sensor System to track robot TCP in motion



Sensor Fusion and Filtering

- Motion Platform
  - Low accuracy, Low dynamics
  - Absolute reference, no drift
- Inertial Measurement Units (IMU)
  - Low accuracy (translation)
  - Good accuracy (rotation)
  - High dynamics
  - Relative measurements
  - prone to drift (2nd order integration)

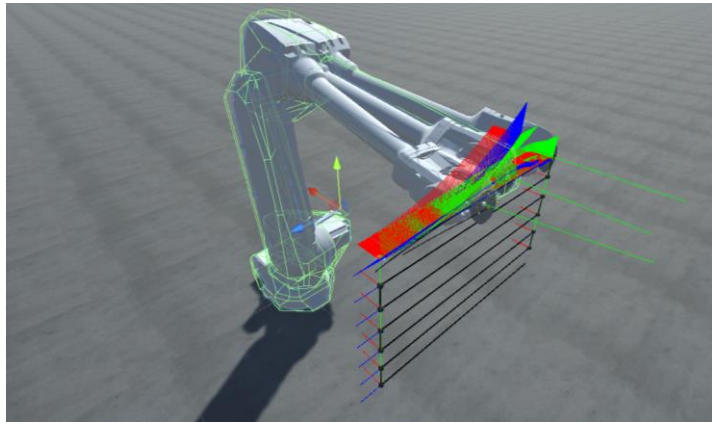
6DoF, positions +/-20µm, orientation +/- 0.2°
sampling rate 1kHz, latency 1ms

Sensor Fusion and Filtering

- Computer Vision
  - High accuracy, Low dynamics
  - Relative positions to features
  - Part configuration required
  - No drift
- Laser Distance Sensors
  - High accuracy (distance to surface)
  - High dynamics
  - As-is surface required
  - No drift

TCP: Tool Center Point

Kalman Filter Design



New estimate
(posterior)
$x_k = \bar{x}_k + K(z - \bar{z}_k)$
$P_k = (1 - K)\bar{P}_k$

Kalman Filter

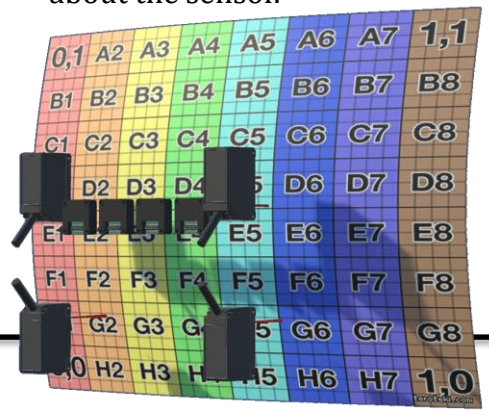
Calculates Kalman Gain (K) based on information on the covariances on states, models and measurements used, to enable an optimal blend between forecast and measurement.

Measurement(s)
z
R

Predicted sensor values
(prior)
$\bar{z}_k = H_k\bar{x}_k + v_k$

Sensor Model

Creates sensor forecast for a given state based on information about the sensor.



State transition
(prediction)
$\bar{x}_k = Fx_{k-1} + w_k + c$

State Transition Model

Predicts the current state of the system, based on previous state, process model, noise and control vector

Predicted state
(prior)
$\bar{x}_k$
$\bar{P}_k$

Previous state
(posterior)
x_{k-1}
P_{k-1}

x

state

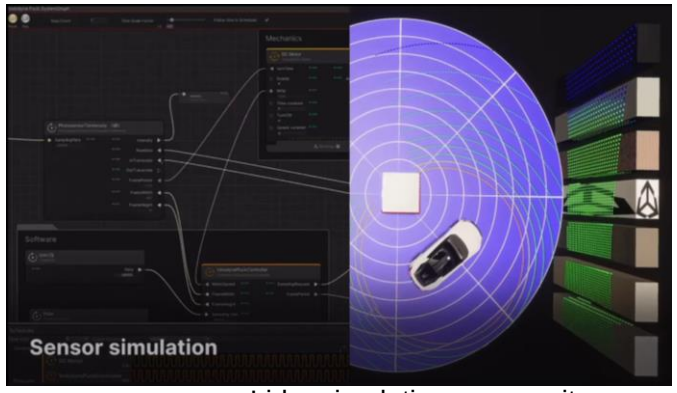
time

t

TCP: Tool Center Point

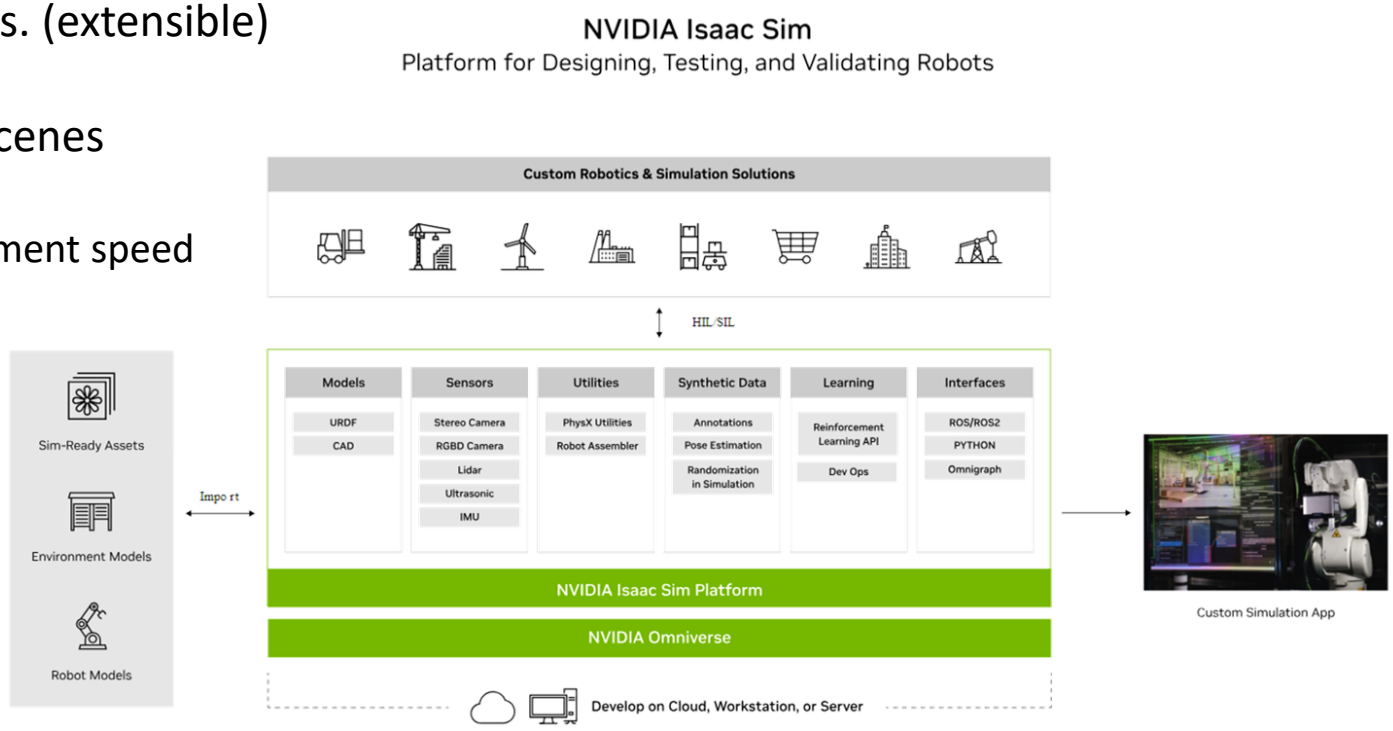
General Purpose 3D Engines

Many different engines exit that are suitable for this task:
Gazebo, NVIDIA Omniverse - IsaacSim, Unity-3D, Unreal etc.



Lidar simulation, www.unity.com

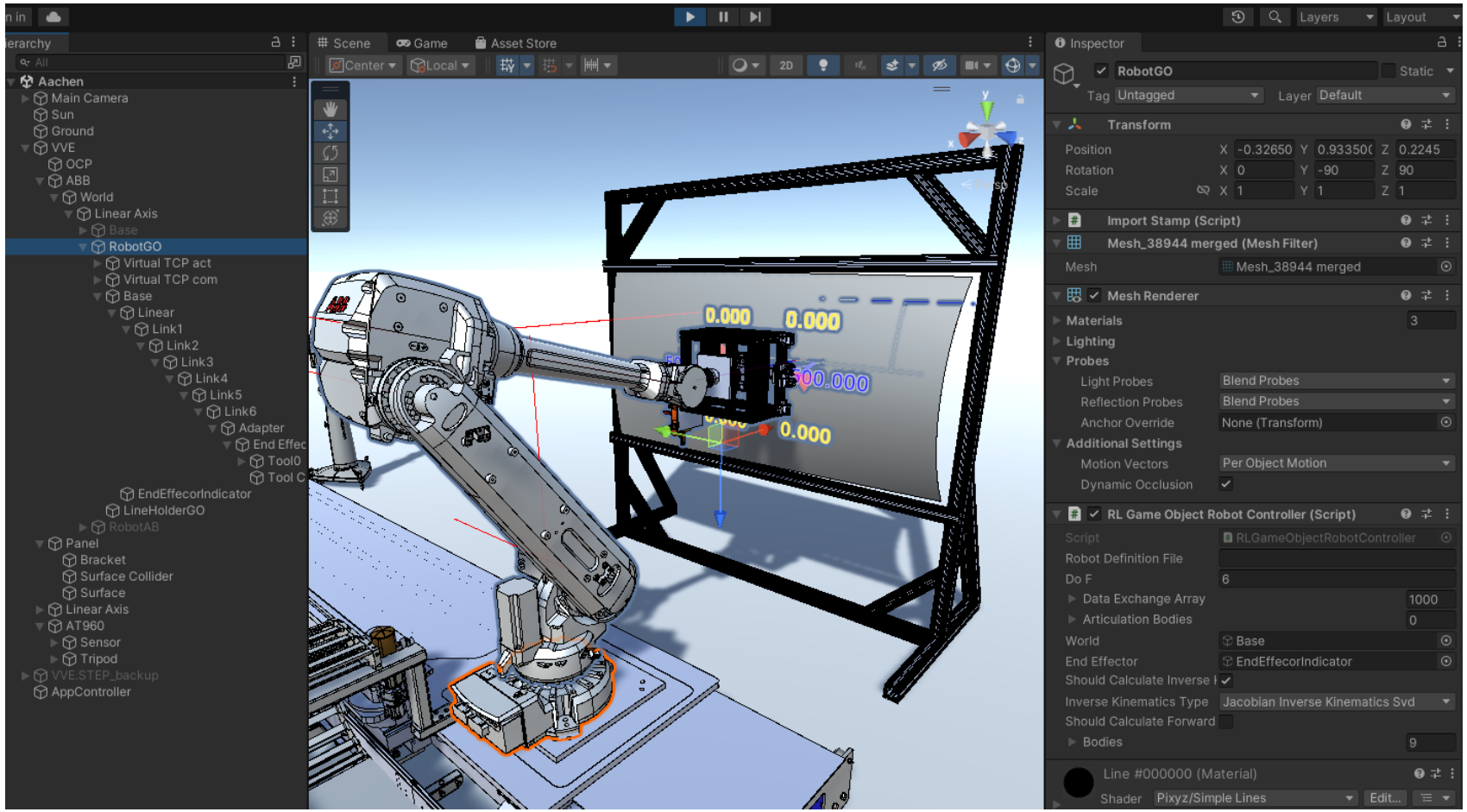
- Has 3D modelling and graphics, to accelerate debugging and plausibility checks
- Has physics engine (Rigid Body or Soft Body) to simulate mechanics (gravity, collision, inertia)
- Has a highly customizable API using industry standard scripts or programing languages (python, C++, C#).
  - Good availability of developers, documentation and tutorials
  - Supports integration with 3rd party libraries. (extensible)
- Typically designed for real-time usage
- Come with authoring tools to model complex scenes
- Pros:
  - Ergonomics, developer availability and development speed
  - Optimized for speed
  - Customizable and extensible
  - Connectivity
  - Cost (changing)
- Cons:
  - Favors speed for accuracy!
  - Some unusual conventions (LHS vs. RHS)
  - Typically limited to tessellated geometry
  - Long-term availability & support



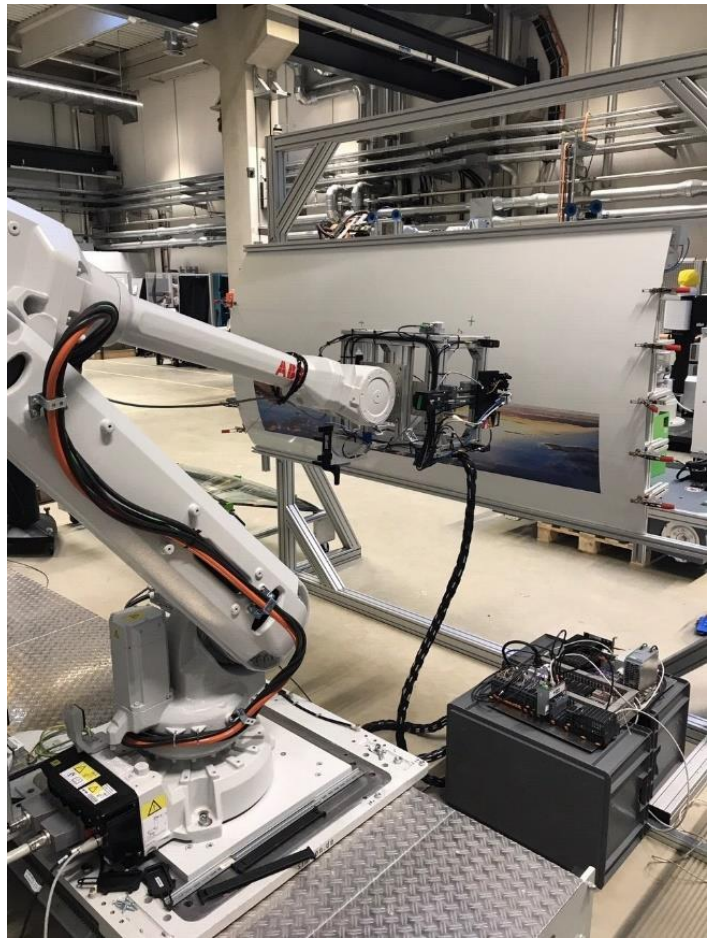
https://developer.nvidia.com/isaac/sim

Simulating the Test Bench

Digital Twin of the WZL Test Setup



Robot setup at RWTH Aachen

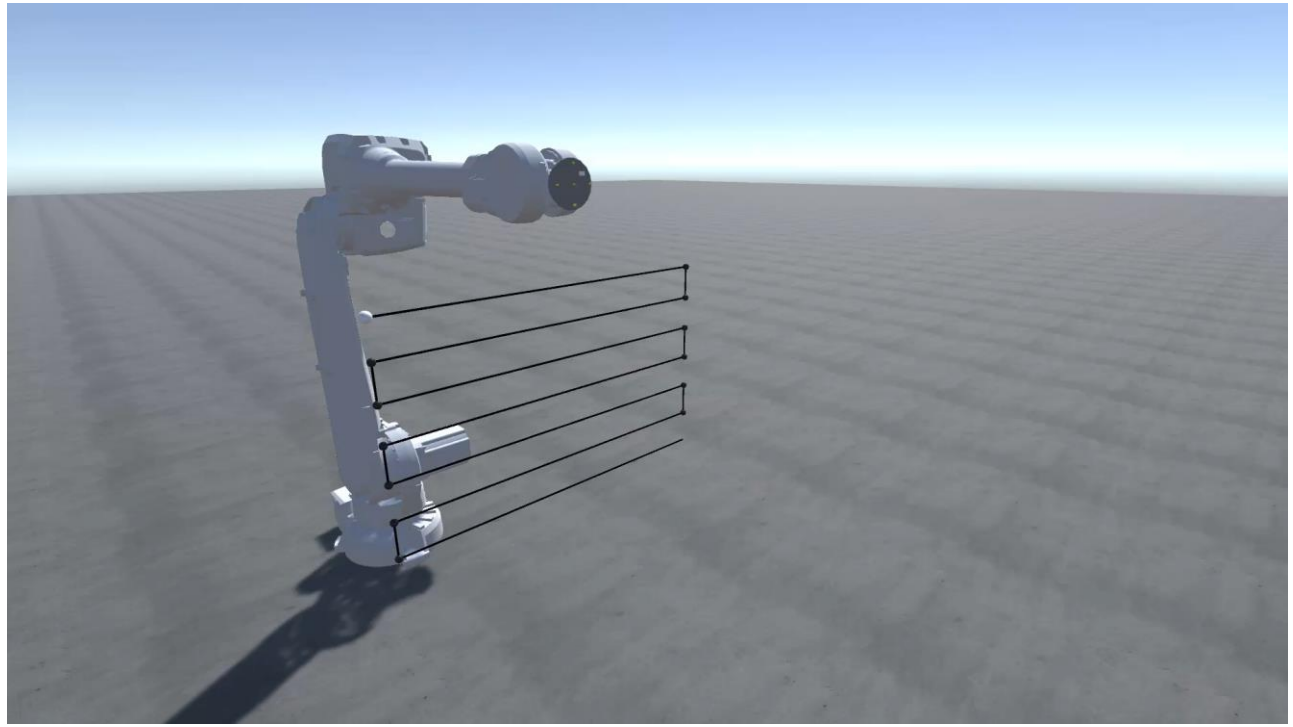


- Documents the experiments
- Enables plausibility checks (i.e. Coordinate Systems)
- Record and Replay real-time experiments
- Create arbitrary amount of synthetic test cases

- Have a perfect ground truth mode
- Modelling the Digital Twin provides a lot of learning on the problem

Simulating the robot

- State Transition: What is the most-likely next pose of the robot?
- Forward Kinematics:
  - PhysX for simulating the robots motion
  - Compares well against Mathworks/SimScape
  - Implementing own models: i.e. current controller, motor cogging (ripple)
- Take care,
  - Significant modifications to the physics settings might be required:



Chair of Robotics, Artificial Intelligence and Real-Time Systems

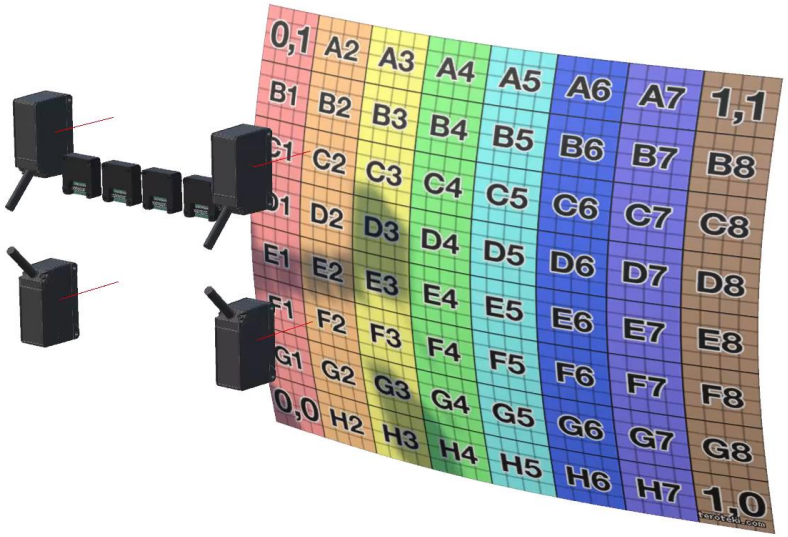
Josip Josifovski
Dr. Markus Rickert
Prof. Dr.-Ing. Alois Knoll

Chair of Applied Mechanics

Tomas Slimak
Dr. techn. Andreas Zwölfer
Prof. Dr. Daniel J. Rixen

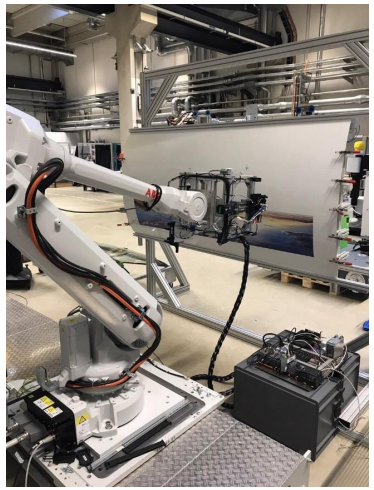
Simulating the sensors

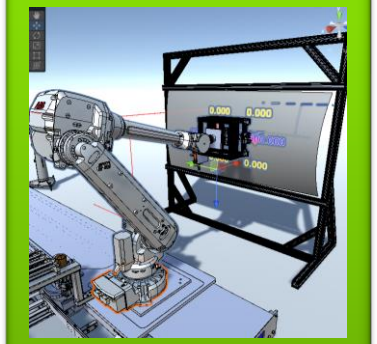
- Laser Distance Sensors or Lasertracker
  - One line of code to simulate using Raycasts
- Camera Based Sensor:
  - Implemented with Cameras and RenderTextures
- Inertial Measurement Units, Load Cells
  - Either using a physics backend as PhysX, which gives a good resolution. But it applies a set of constrain on modelling the scene.
  - Or use own differentiation, much more flexible, but may come with unintended results.
- Take Care:
  - Accuracy, Heuristics and State Based Physics



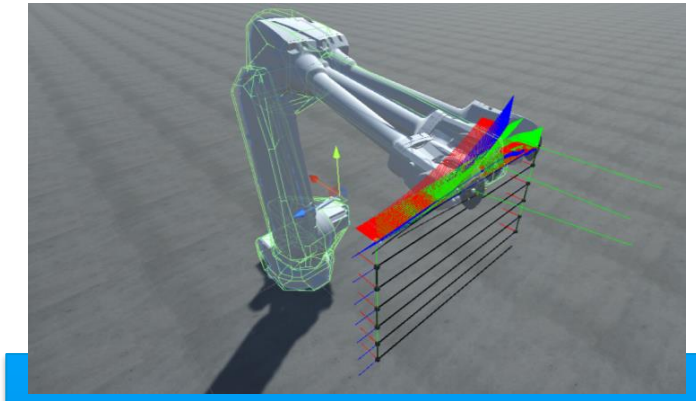

```
21    private void measureDistance()
22    {
23        if (Physics.Raycast(transform.position, transform.forward, out hit, maxViewDistance, layerMask))
24        {
25            //In Range
26            distance = transform.InverseTransformDirection(hit.point - transform.position).magnitude / resolution;
27            contact = true;
28        }
29        else
30        {
31            //Out of Range
32            contact = false;
33            distance = float.NaN;
34        }
35    }
```

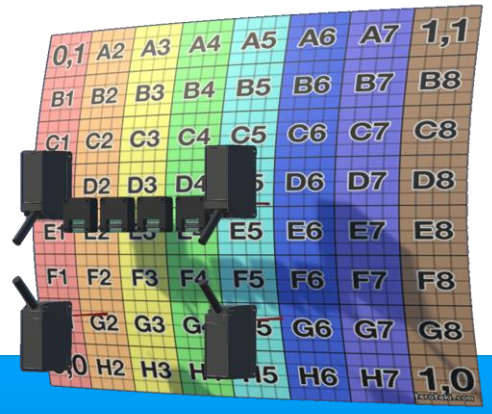
Using the Metaverse in Metrology -
Leveraging a general-purpose 3D physics engine for sensor data interpretation





Data Exchange: UDP – Streams





$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k$$

UNITY
Physics model

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$$

Update [edit]

Innovation or measurement pre-fit residual $\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}$

Innovation (or pre-fit residual) covariance $\mathbf{S}_k = \mathbf{H}_k\hat{\mathbf{P}}_{k|k-1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k$

C# Filter

Optimal Kalman gain $\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1}$

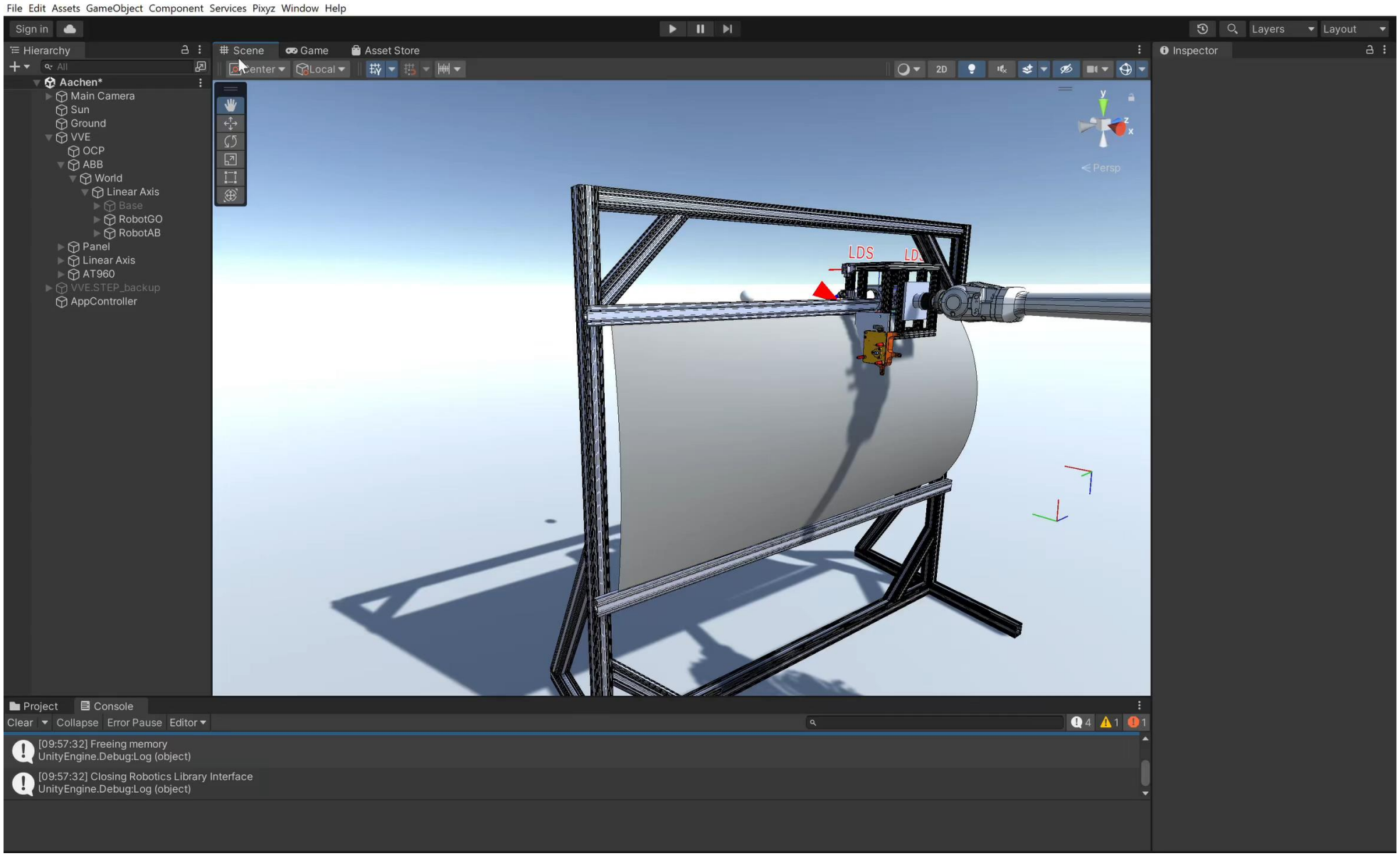
Updated (*a posteriori*) state estimate $\mathbf{x}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$

Updated (*a posteriori*) estimate covariance $\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\hat{\mathbf{P}}_{k|k-1}$

Measurement post-fit residual $\tilde{\mathbf{y}}_{k|k} = \mathbf{z}_k - \mathbf{H}_k\mathbf{x}_{k|k}$

UNITY – Digital Twin / Test Bench (document, record, replay, simulate)

Simulating the Test Bench



Boeing Research & Technology