Learning-based methods applied to real-time control using the integrated MARTe2/MDSplus framework

Yehoon Lee, Joshua Stillerman, Darren Garnier (MIT Plasma Science and Fusion Center)
Gabriele Manduchi (Consorzio RFX Padova)

14th IAEA TM on Control Systems, Data Acquisition, Data Management, and Remote Participation in Fusion Research

Table of Content

MARTe2-MDSplus framework

MARTe2
Real-time control framework

- Flexible, cross platform real-time execution environment
- Code quality / testing – MISRAC++:2008
- Based on MARTe used by existing tokamaks (JET, TCV, etc.)
- XML description
    - Compute Modules GAMs
    - IO GAMs
    - DataSources

MDSplus
Data management framework

- Widely used in fusion community
- Hierarchal data management for experimental data integrating:
  - Raw and processed data
  - Machine configuration
  - Shot related task management
  - etc.

- Generic MARTe2 objects can be defined as MDSplus devices in **common environments/languages**, such as Simulink, Python, etc.

- An "instance" of the MARTe2 process is automatically created for pulse shots, **saving the data and configurations as a MDSplus Tree.**

- Used in ITER Neutral Beam Testing Stand, planned for DTT.

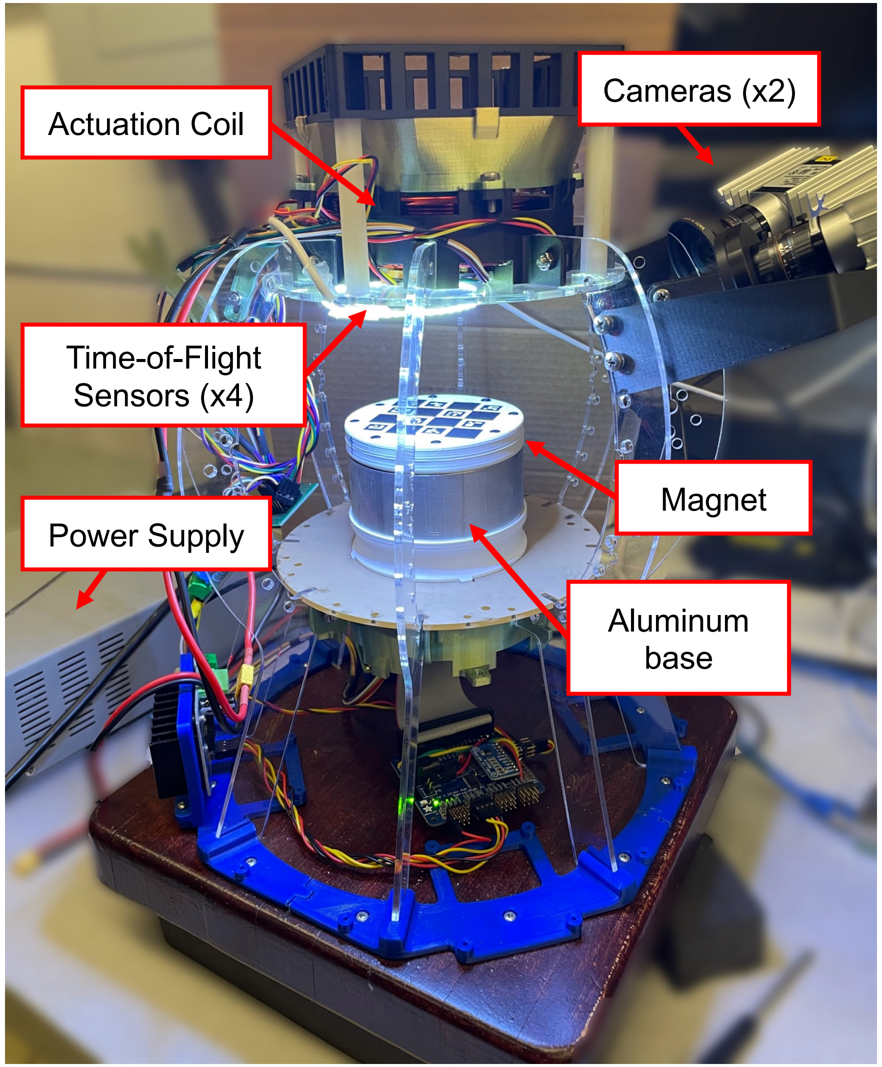
The Levitated Bagel

Magnetic levitation system as an emulator for fusion experiments to explore, test, and validate modern control methods.

Elements of real-time control systems to incorporate:

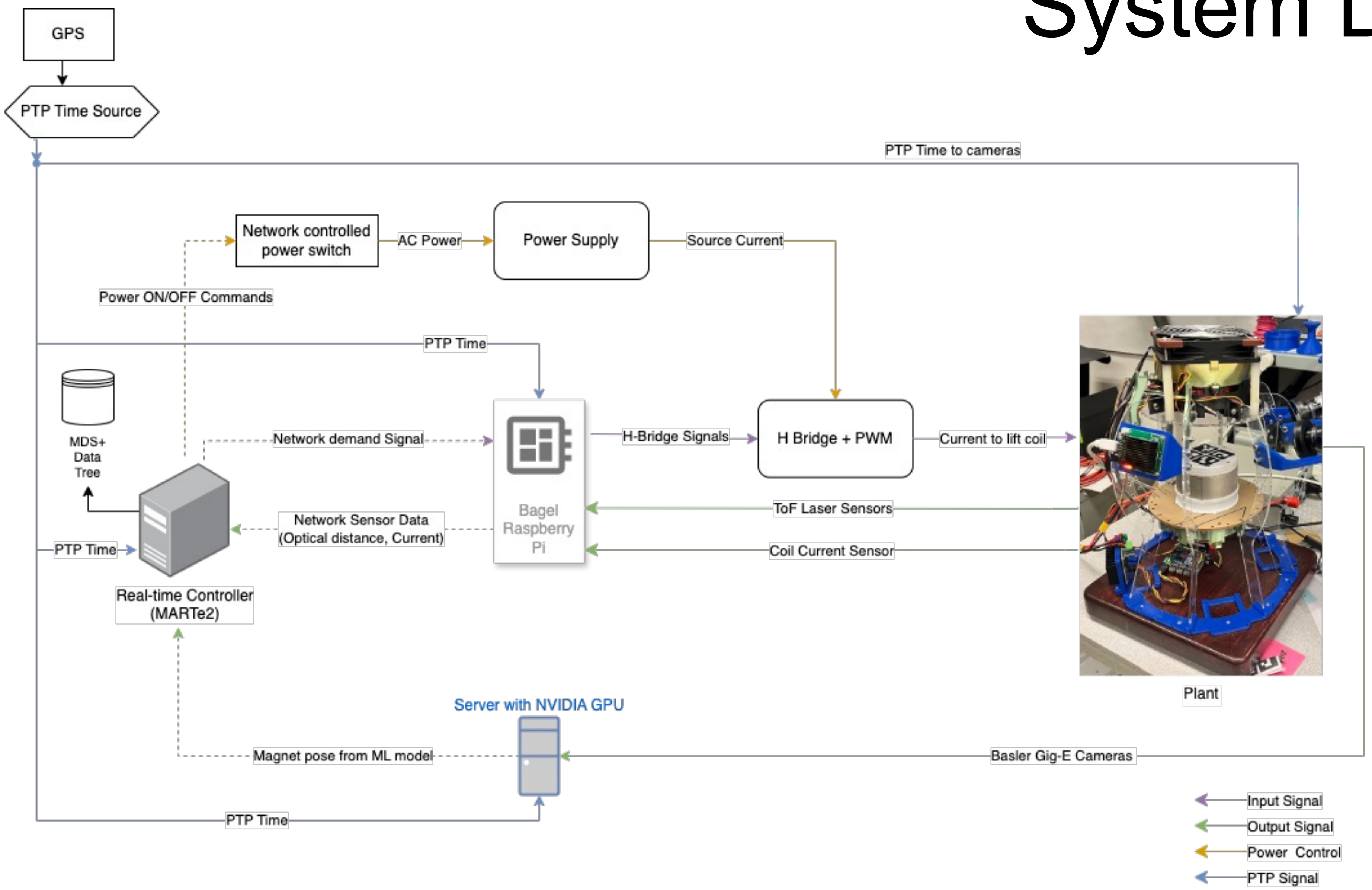
- MARTe2 process managed through MDSplus
- Networked, modular system
- Controller designed in MATLAB Simulink
- Synchronous multi-rate computation
- Asynchronous event-based activities
- Controller switching
- MIMO scalability
- etc.

Implemented a LQRi controller with Kalman Filter for tracking control of set reference trajectory



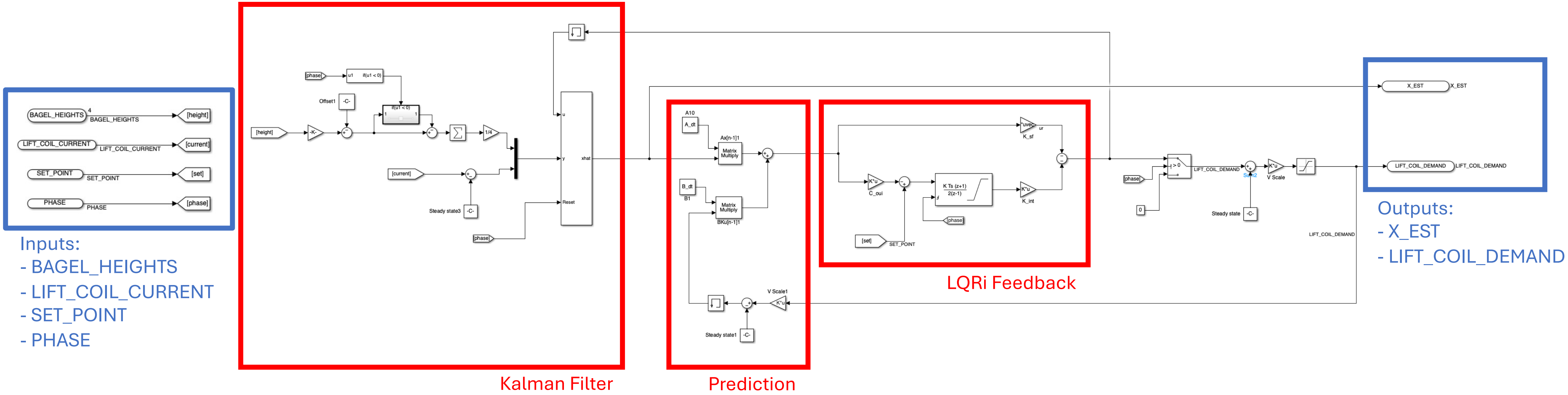
Physical set up of the levitated bagel

System Diagram



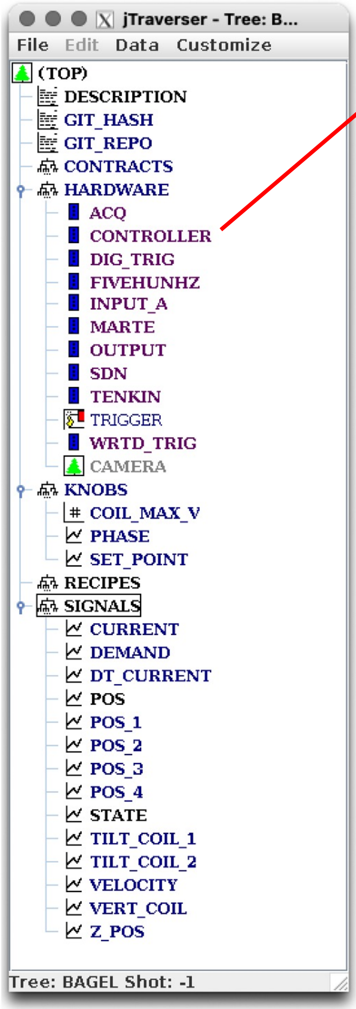
State space controller design

- Prediction-based LQRi controller with linear Kalman Filter
- Applied linearized state space model about initial position
- Discrete time step: 0.01 s (100 Hz)



Simulink model of the levitated bagel controller

MARTe2/MDSplus Tree structure



HARDWARE:
- Devices and elements required in MARTe2 process

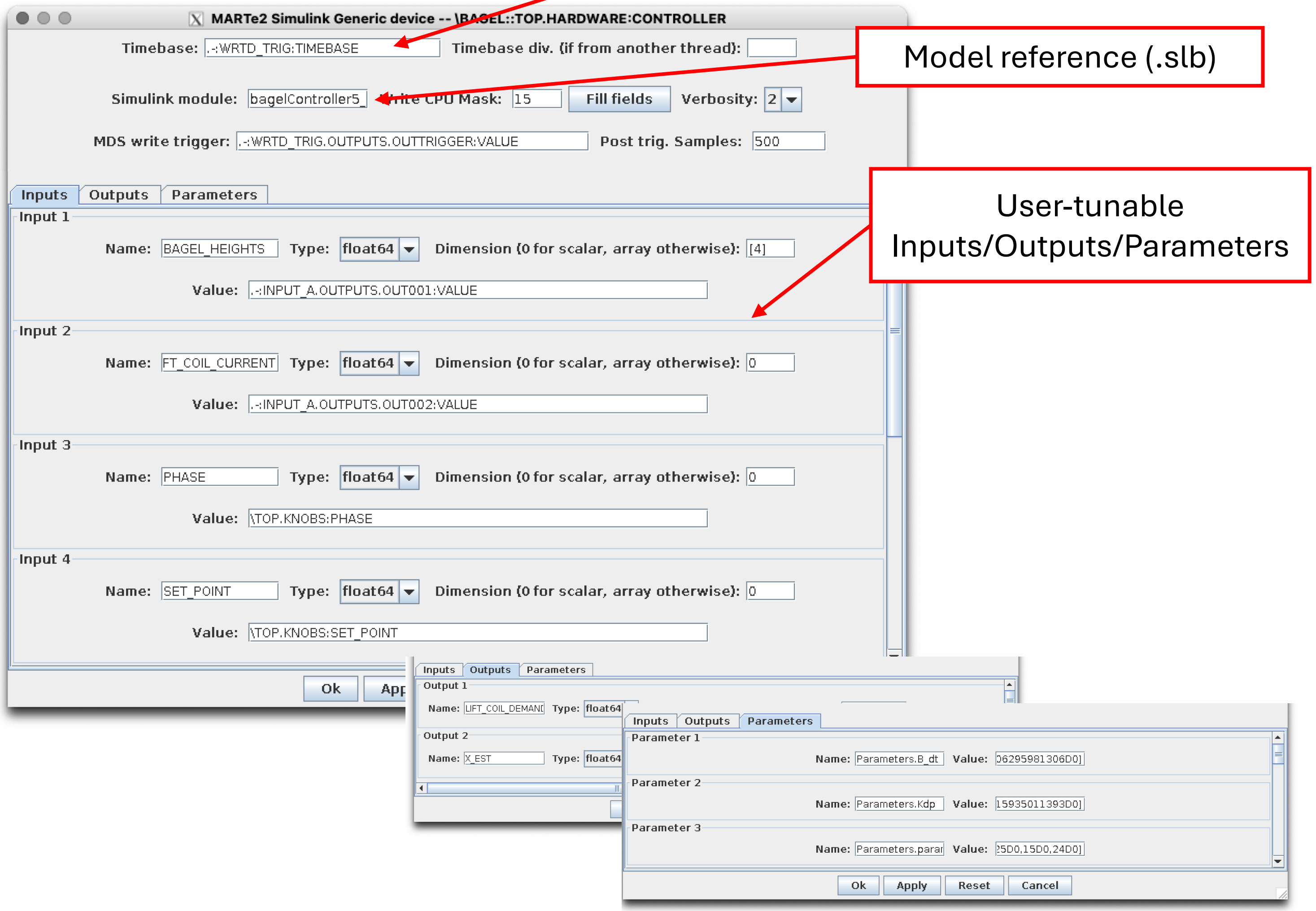
KNOBS:
- Tunable shot settings

SIGNALS:
- Defined operations on saved signals during shot.

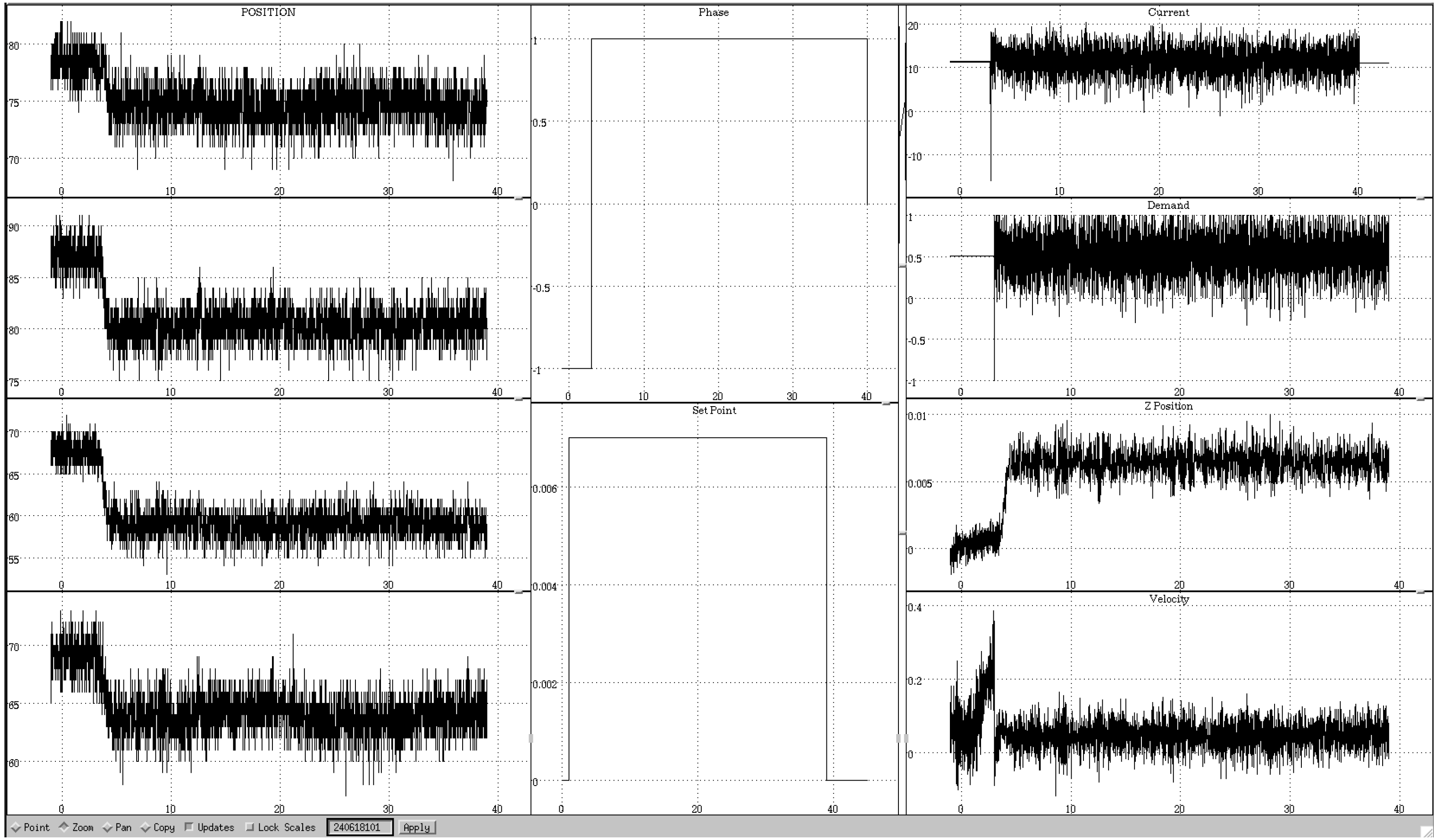
Synchronous Timebase

Model reference (.slb)

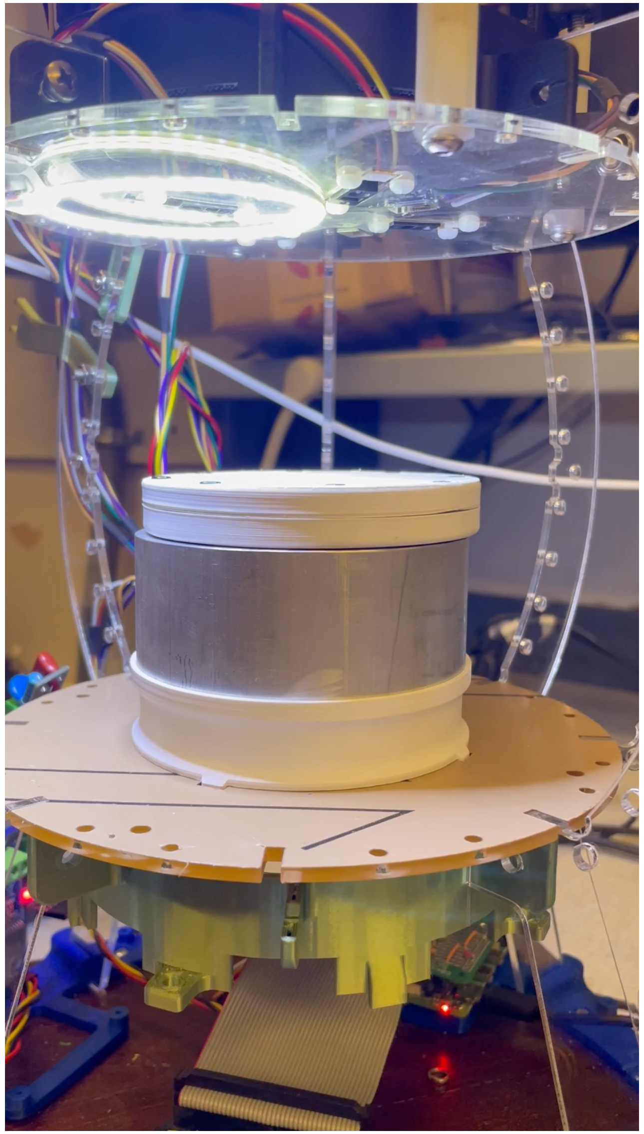
User-tunable Inputs/Outputs/Parameters



Magnetic levitation



Measured data of magnetic levitation

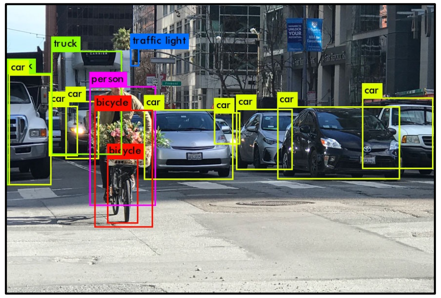

Example of magnetic levitation

Vision-based Observer

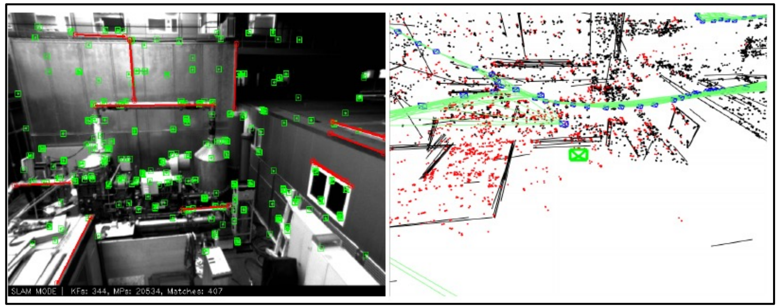
1. **Provide 6D pose estimation.**
   Used as improvements on lower-dimension systems.

2. **Simple example of data-driven methods.**
   Decoupled from physics and can be developed in parallel.

Human pose estimation



Object detection



Position tracking

1. System set-up

↓

2. Data acquisition/analysis

↓

3. Model development

↓

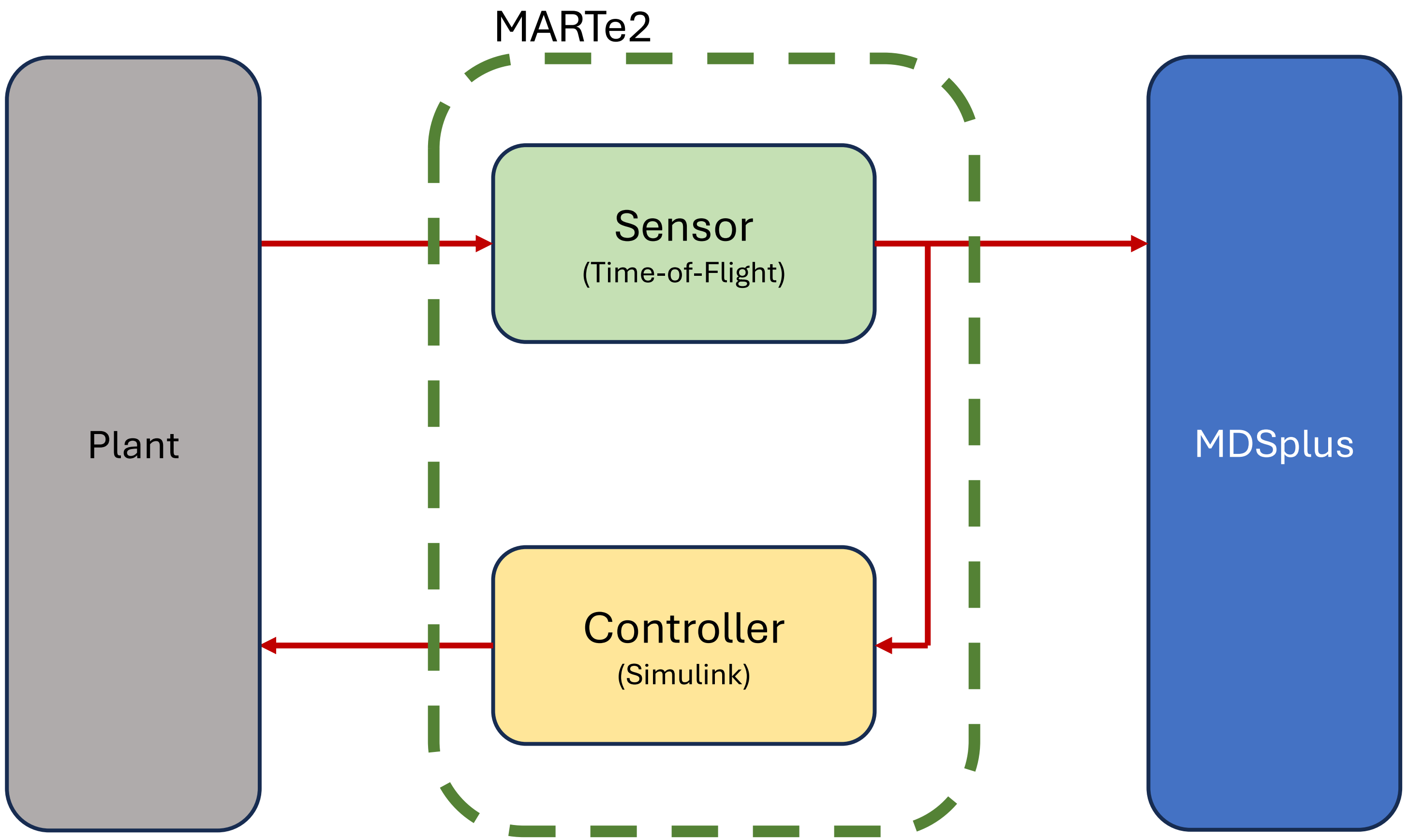
4. Real-time implementation

Modelling process

Images from:
https://stasiuk.medium.com/pose-estimation-metrics-844c07ba0a78
https://medium.com/@pedroazevedo6/object-detection-state-of-the-art-2022-ad750e0f6003
https://medium.com/@dcasadoherraez/introduction-to-visual-slam-chapter-1-introduction-to-slam-a0211654bf0e

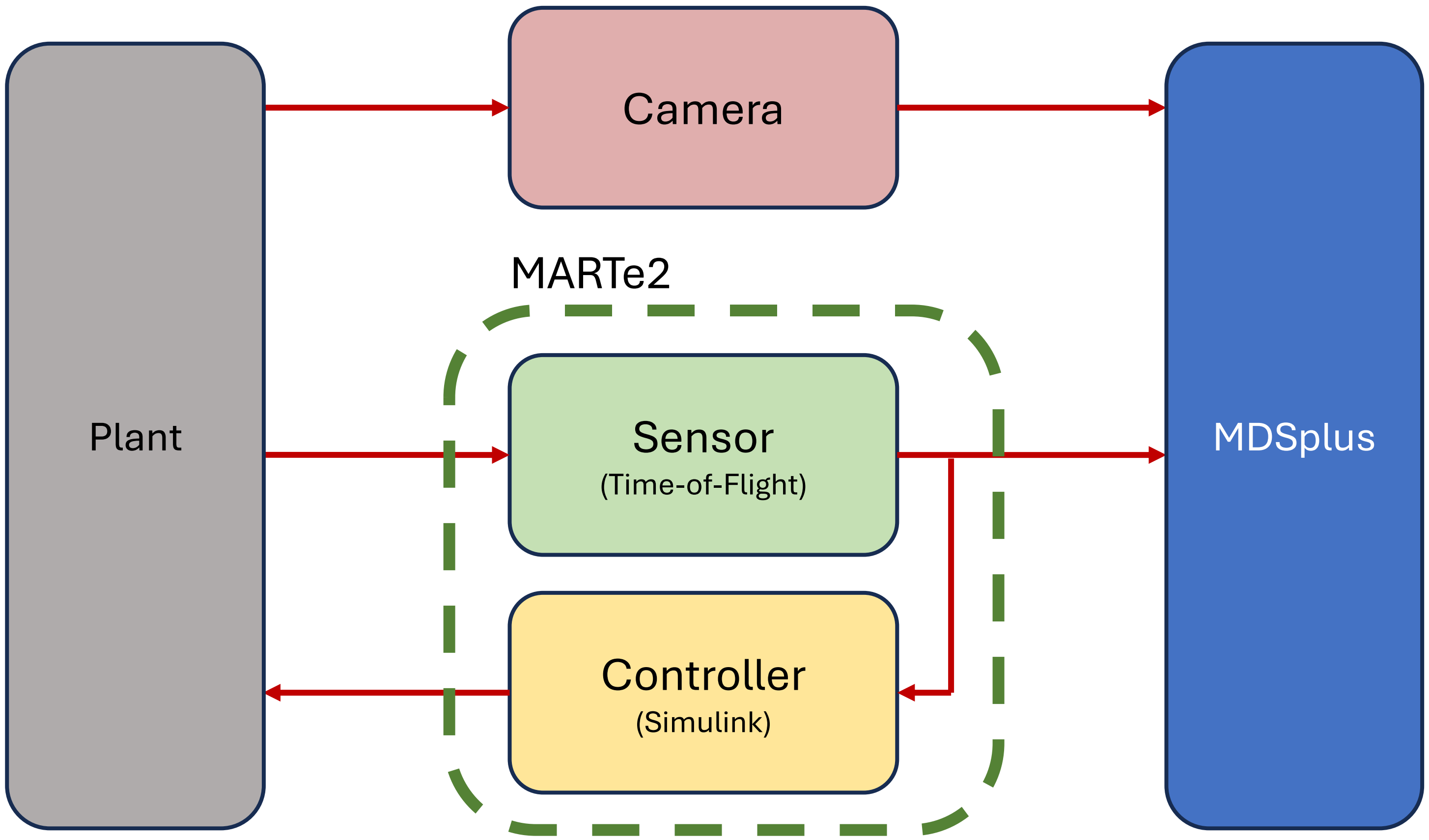
Model Development Process (1/4)

1. **System set-up**
2. Data acquisition/analysis
3. Model development
4. Real-time implementation



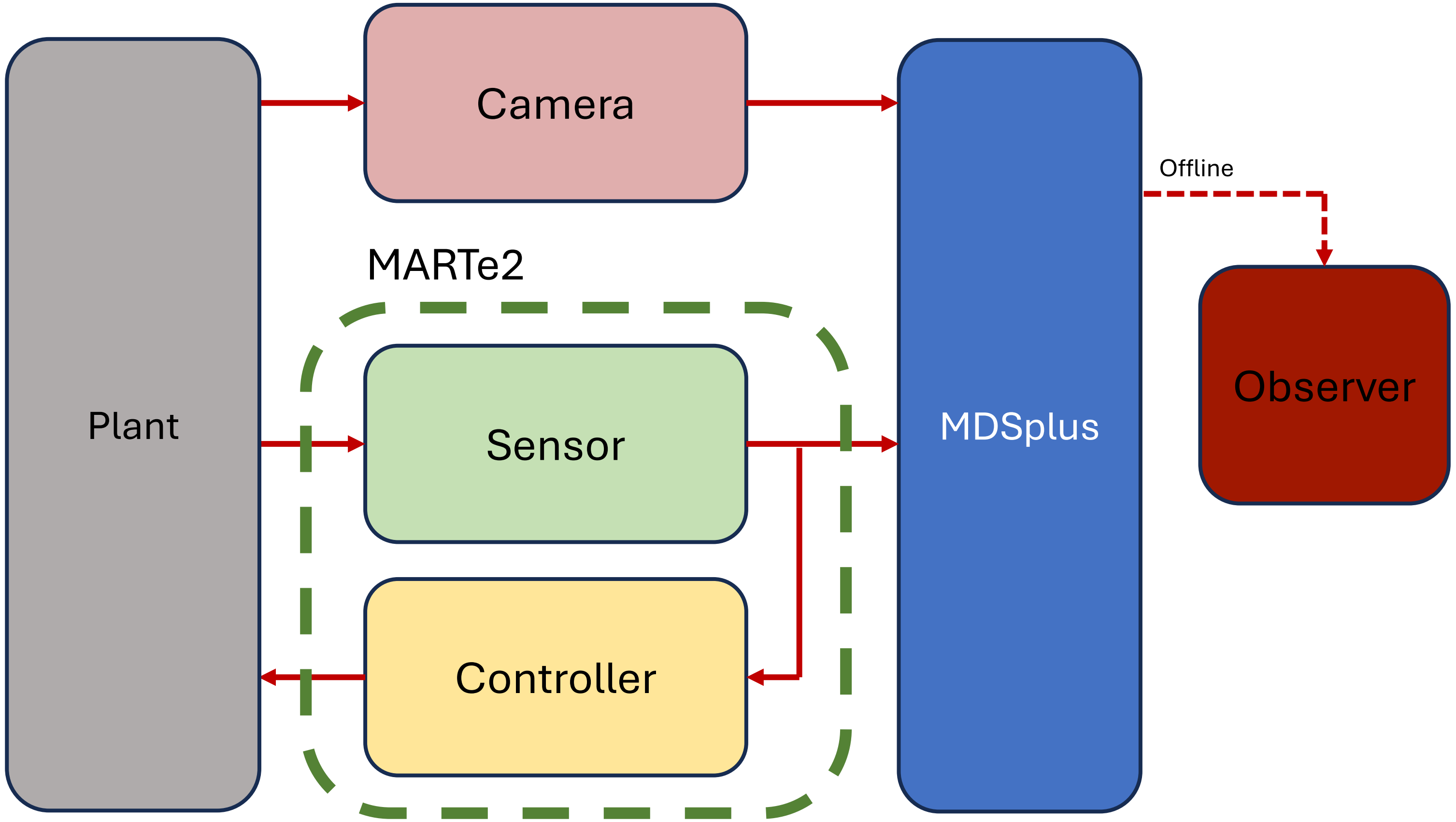
Model Development Process (2/4)

1. System set-up
2. **Data acquisition/analysis**
3. Model development
4. Real-time implementation



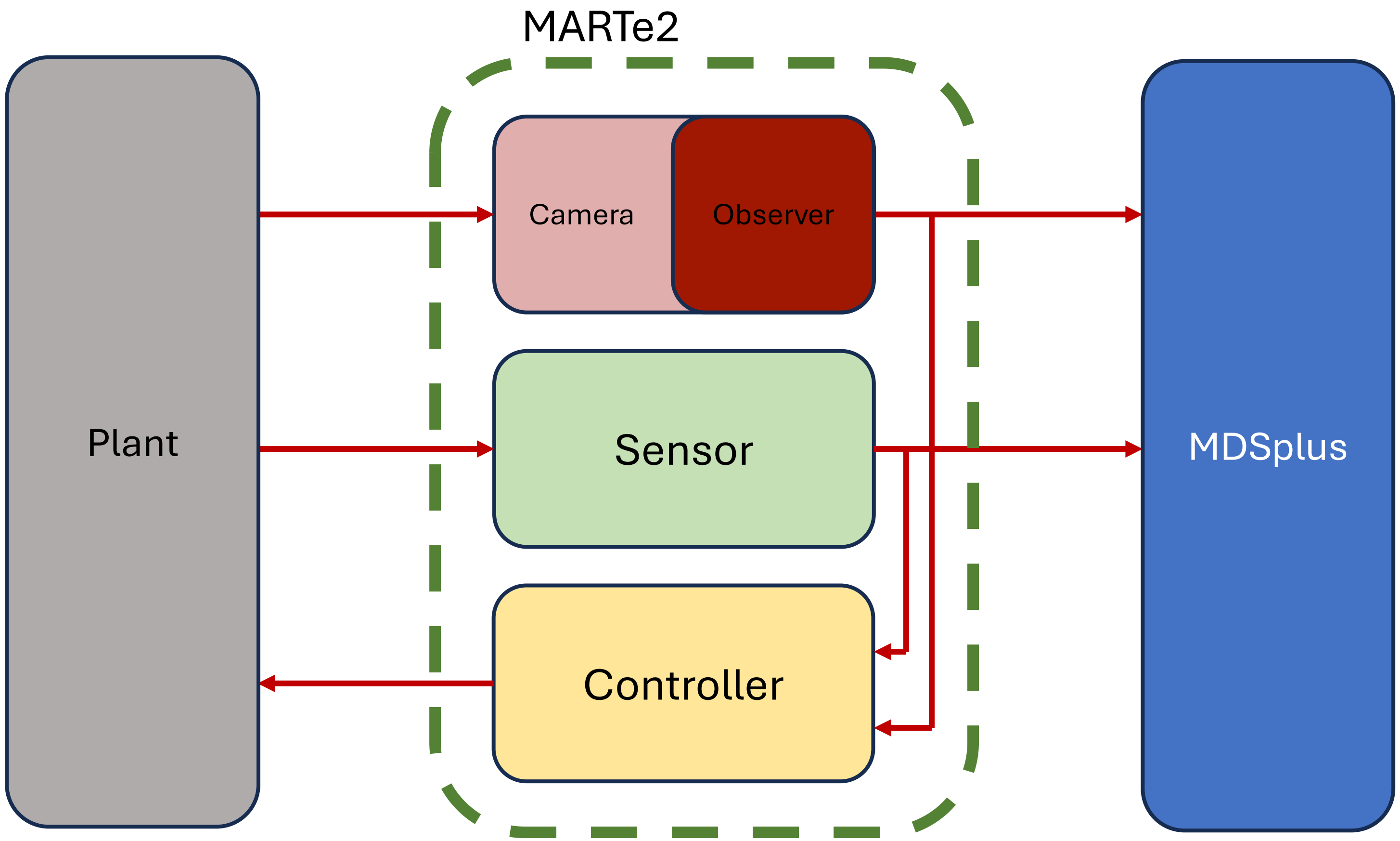
Model Development Process (3/4)

1. System set-up
2. Data acquisition/analysis
3. **Model development**
4. Real-time implementation



Model Development Process (4/4)

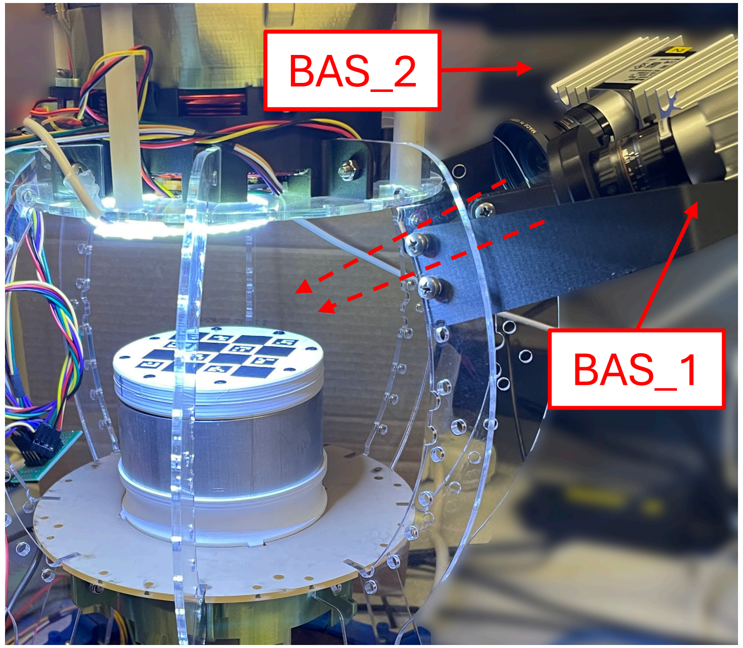
1. System set-up
2. Data acquisition/analysis
3. Model development
4. **Real-time implementation**



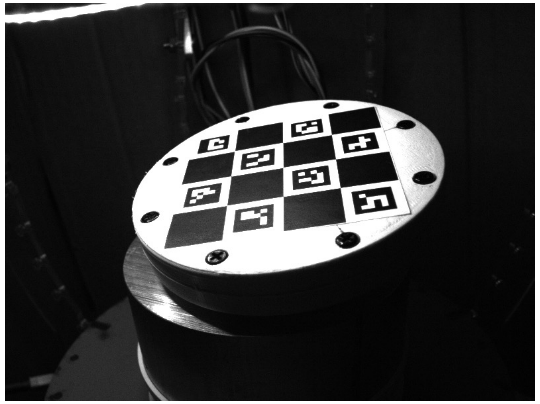
Camera infrastructure setup

Camera model:

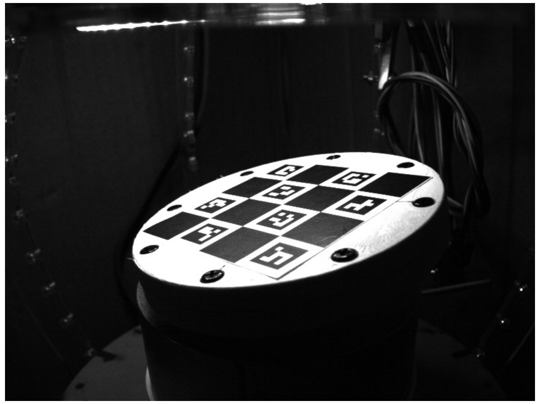
Basler acA800-200gm



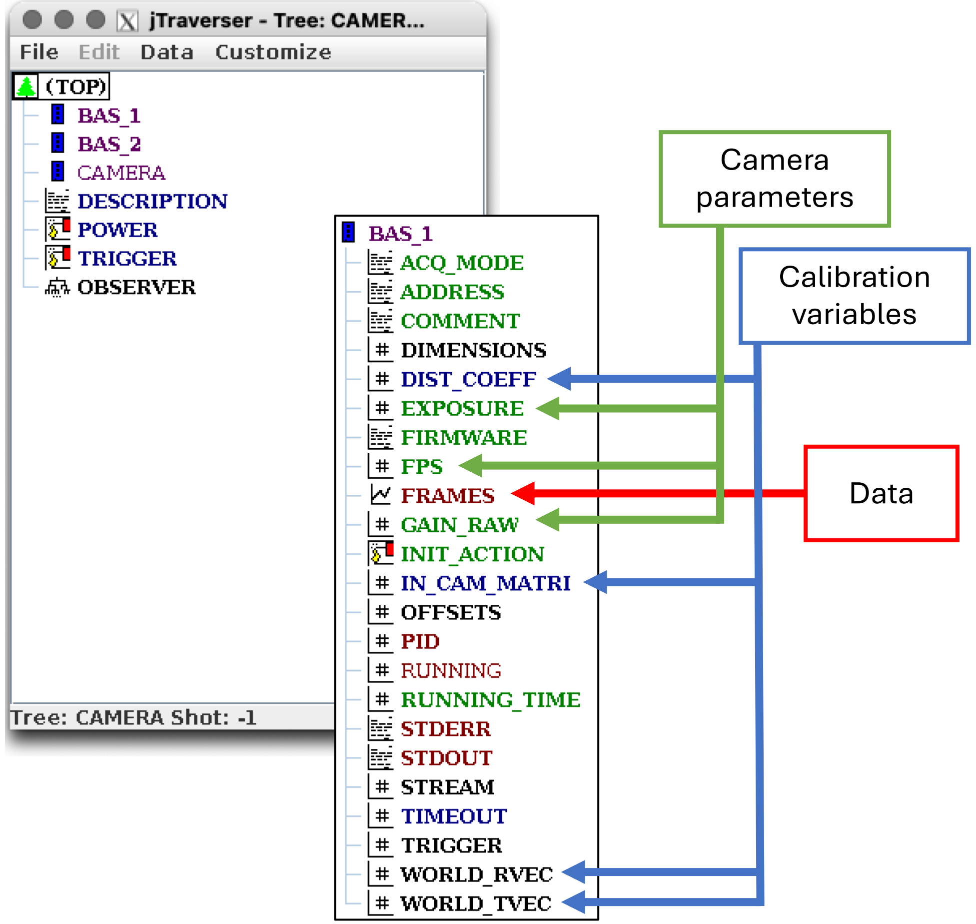
Camera configuration



BAS 1



BAS 2

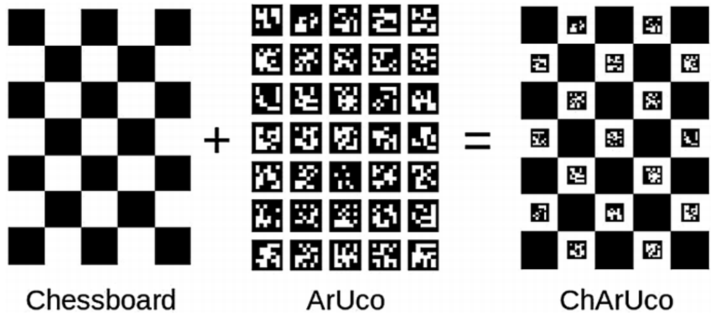


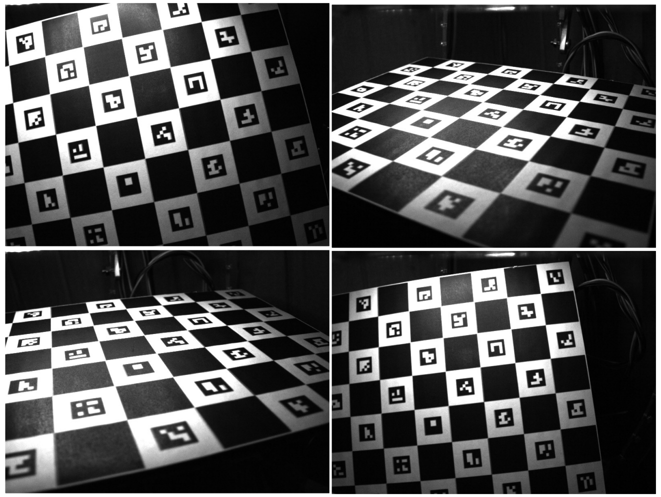
Camera tree structure

OpenCV ChArUco patterns

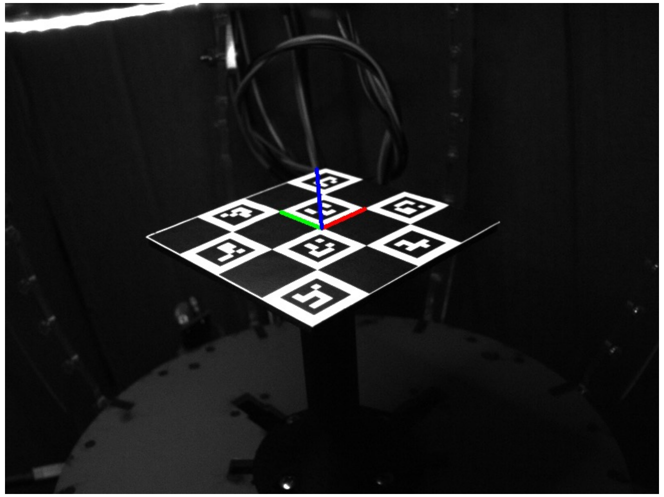
Fiducial marker for commonly used in computer vision.

- Corner recognition of chessboard + Robustness of ArUco
- Identifies ID and position of inner corners of chessboard
- Utilized in camera calibration
  → **Compute intrinsic camera matrix, distortion coefficients**
- Computes rotational and translational vectors in camera frame
  → **Define world reference frame with base stand.**





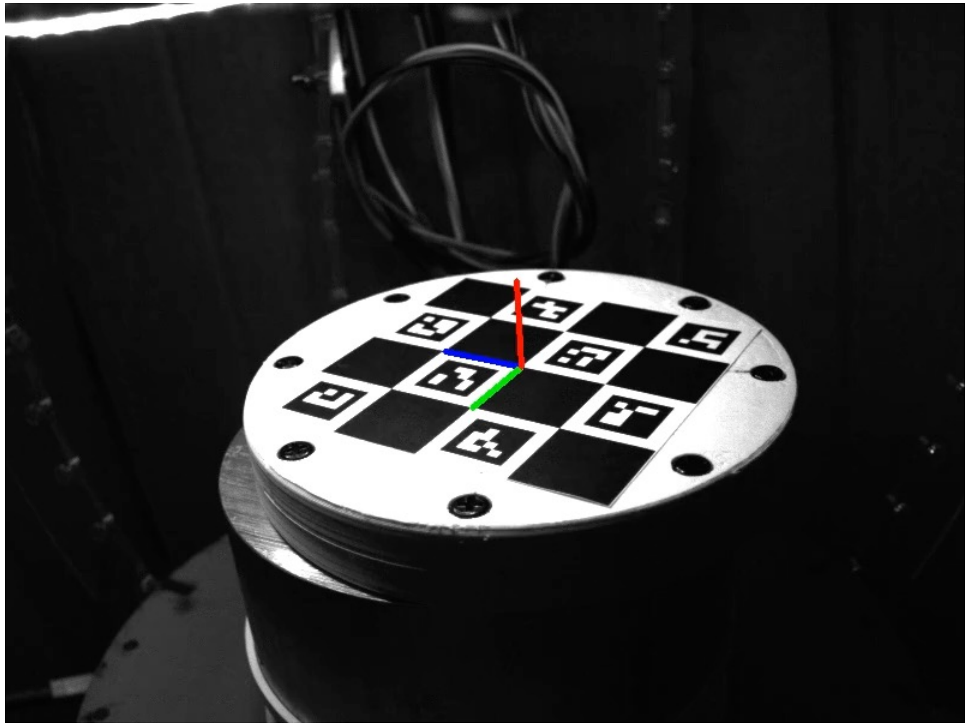
Calibration images

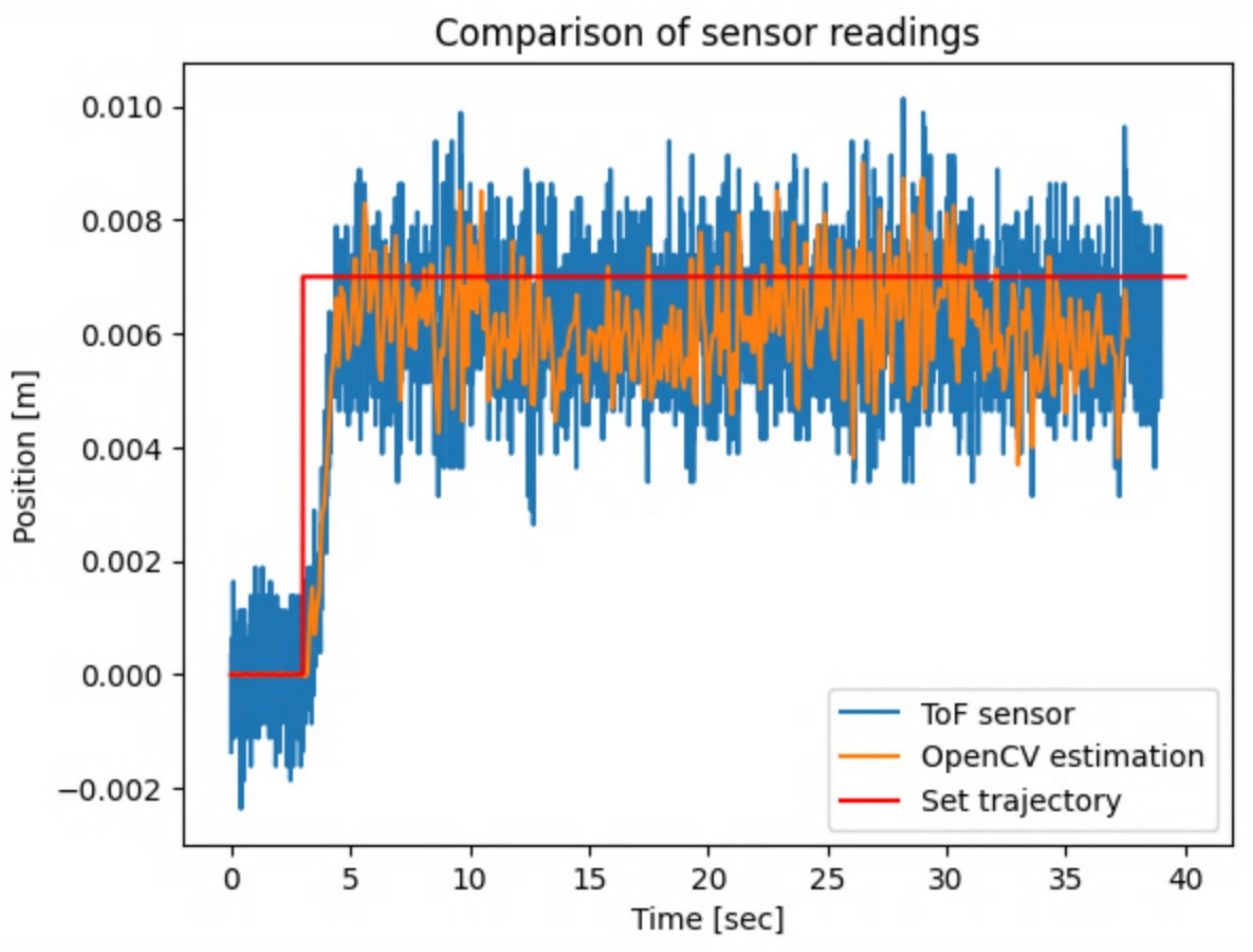


Defining world reference frame

ChArUco-based pose estimation

Frame rate: 20 fps





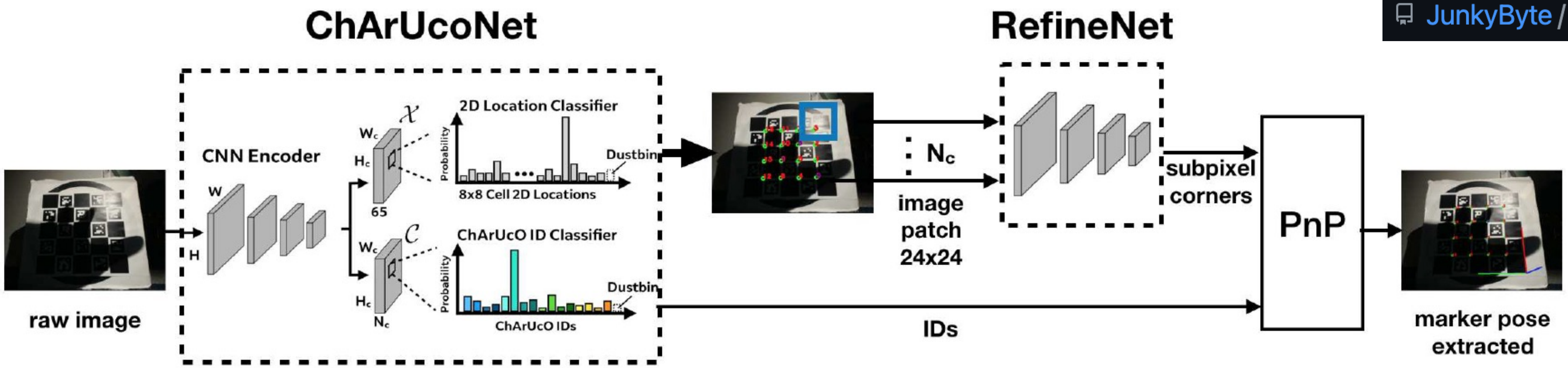
Neural net model

Deep ChArUco: Dark ChArUco Marker Pose Estimation

Danying Hu, Daniel DeTone, and Tomasz Malisiewicz

Magic Leap, Inc.
{dhu,ddetone,tmalisiewicz}@magicleap.com

JunkyByte / deepcharuco Public



1. ChArUcoNet

Identifies location and ID of keypoints (inner corners).

Train on shot data

2. RefineNet

Refines subpixel location of corners from extracted patches.

Implement pre-trained model

3. solvePnP

Geometry-based pose estimation. (Same as OpenCV estimation method)

Algorithm-based operation

ChArUcoNet details PyTorch

SuperPoint: Self-Supervised Interest Point Detection and Description

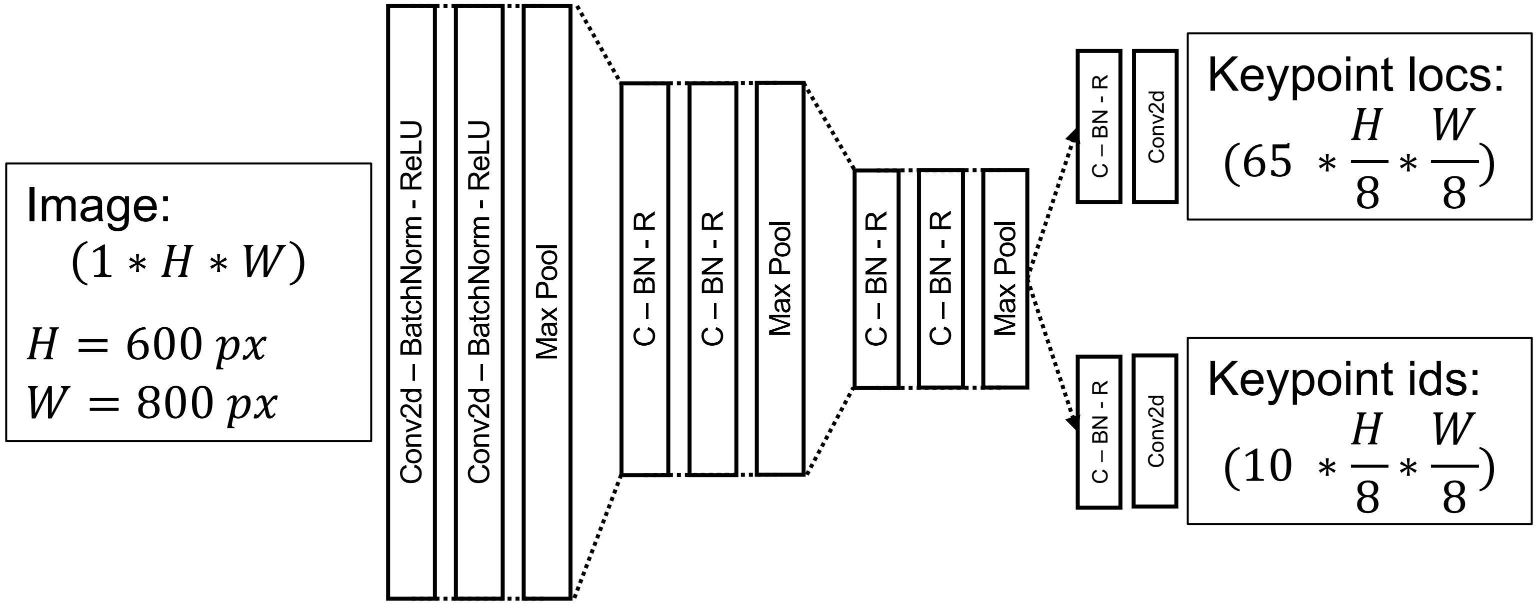
Daniel DeTone
Magic Leap
Sunnyvale, CA
ddetone@magicleap.com

Tomasz Malisiewicz
Magic Leap
Sunnyvale, CA
tmalisiewicz@magicleap.com

Andrew Rabinovich
Magic Leap
Sunnyvale, CA
arabinovich@magicleap.com

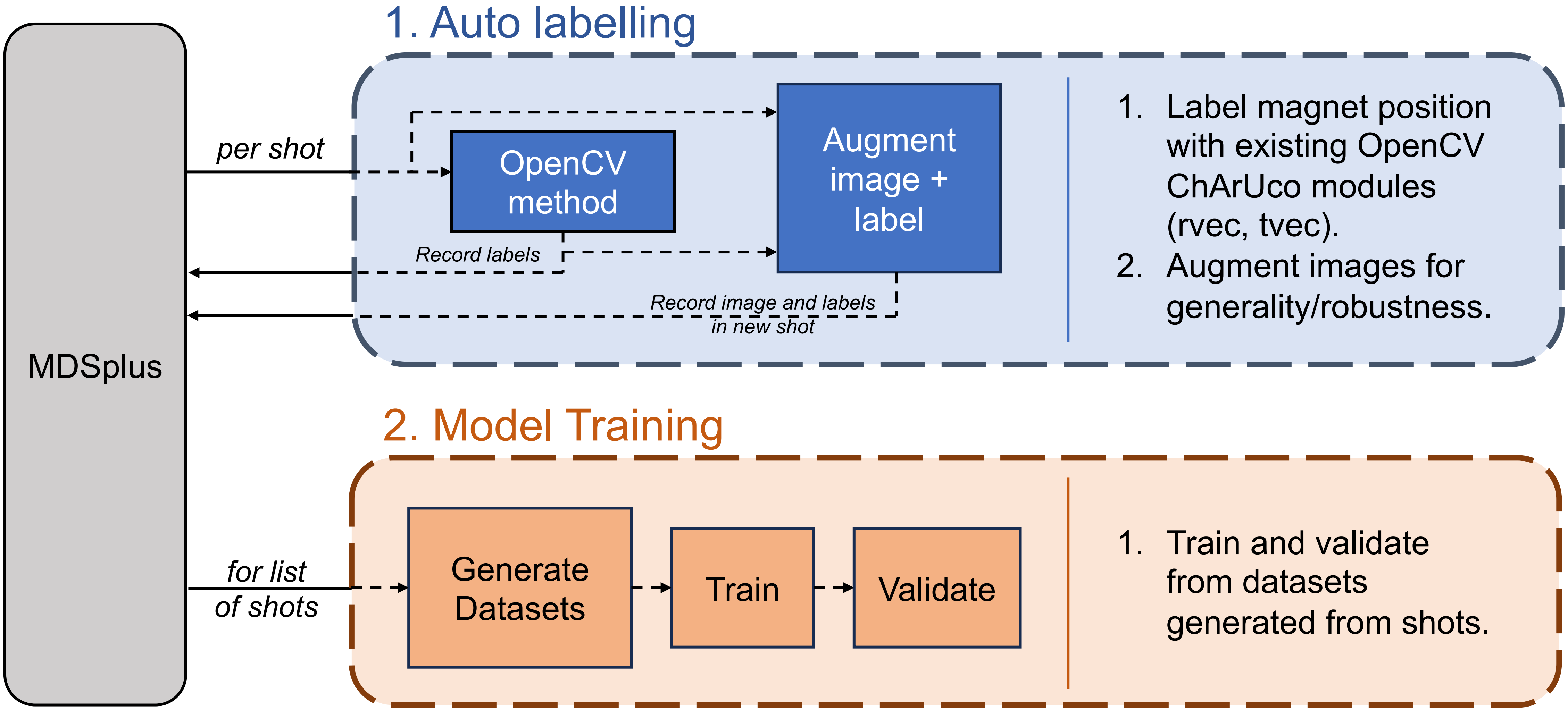
Model:

Based on SuperPoint VGG-style encoder with detector heads changed for purpose.

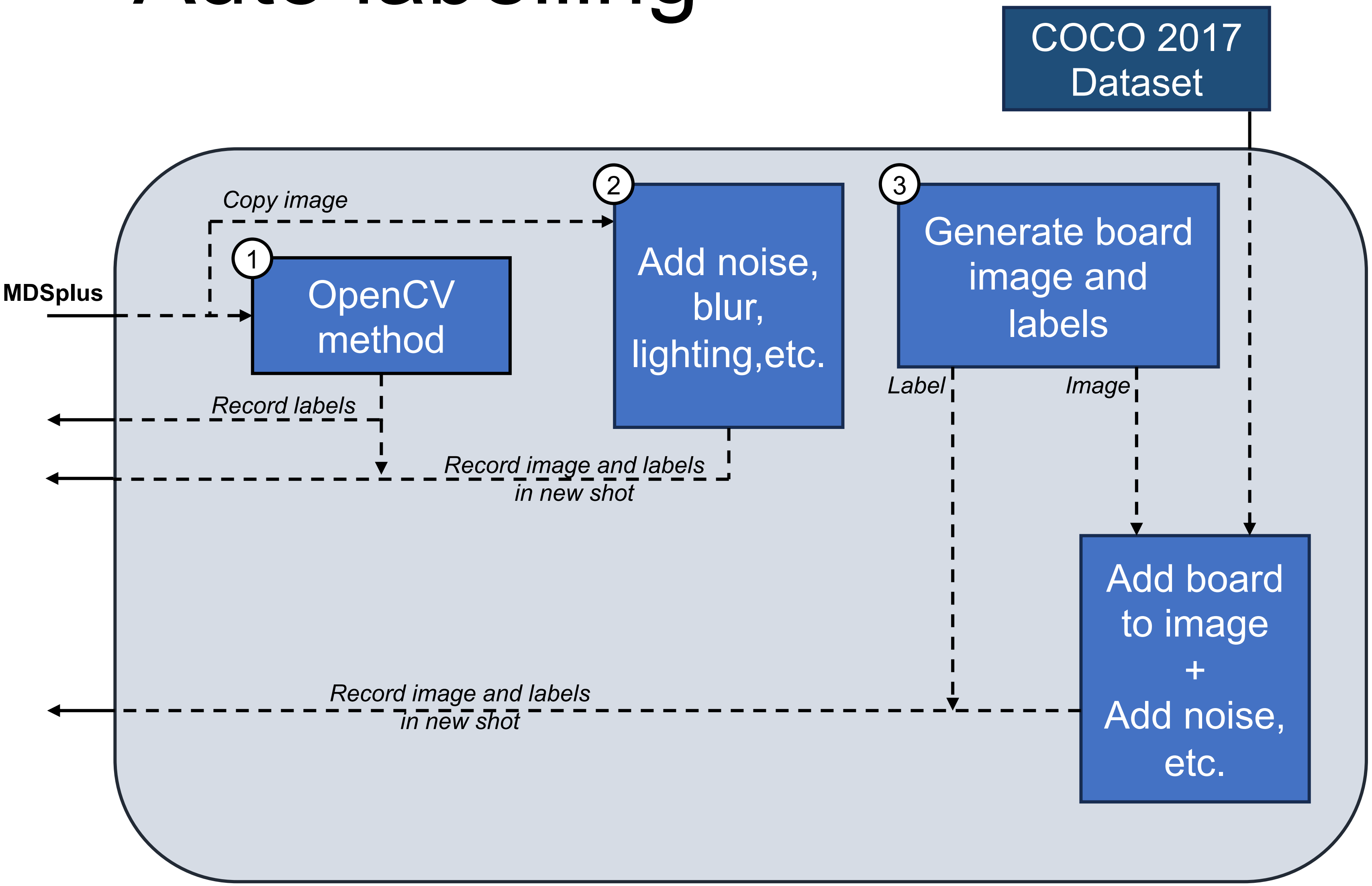


Network architecture

Training pipeline

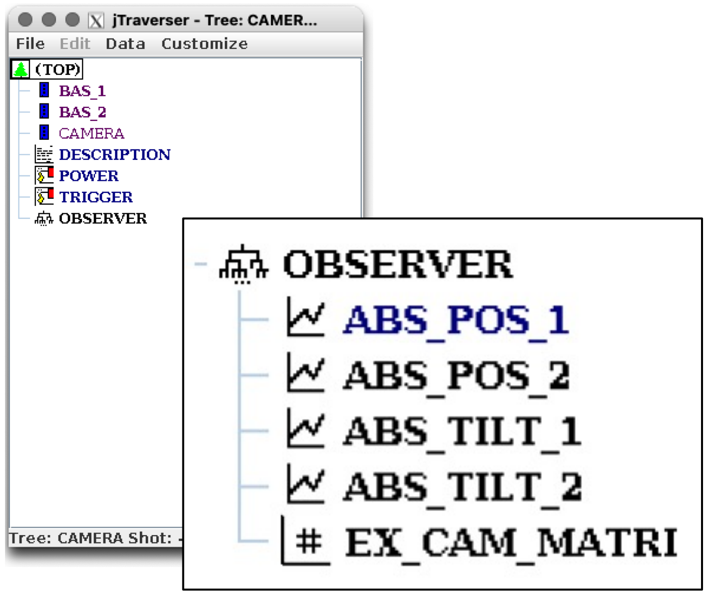


Auto labelling



Auto labelled datasets

1. Data image
   + OpenCV estimated label
2. Augmented data image
   + OpenCV estimated label
3. Generated image
   + Generated label

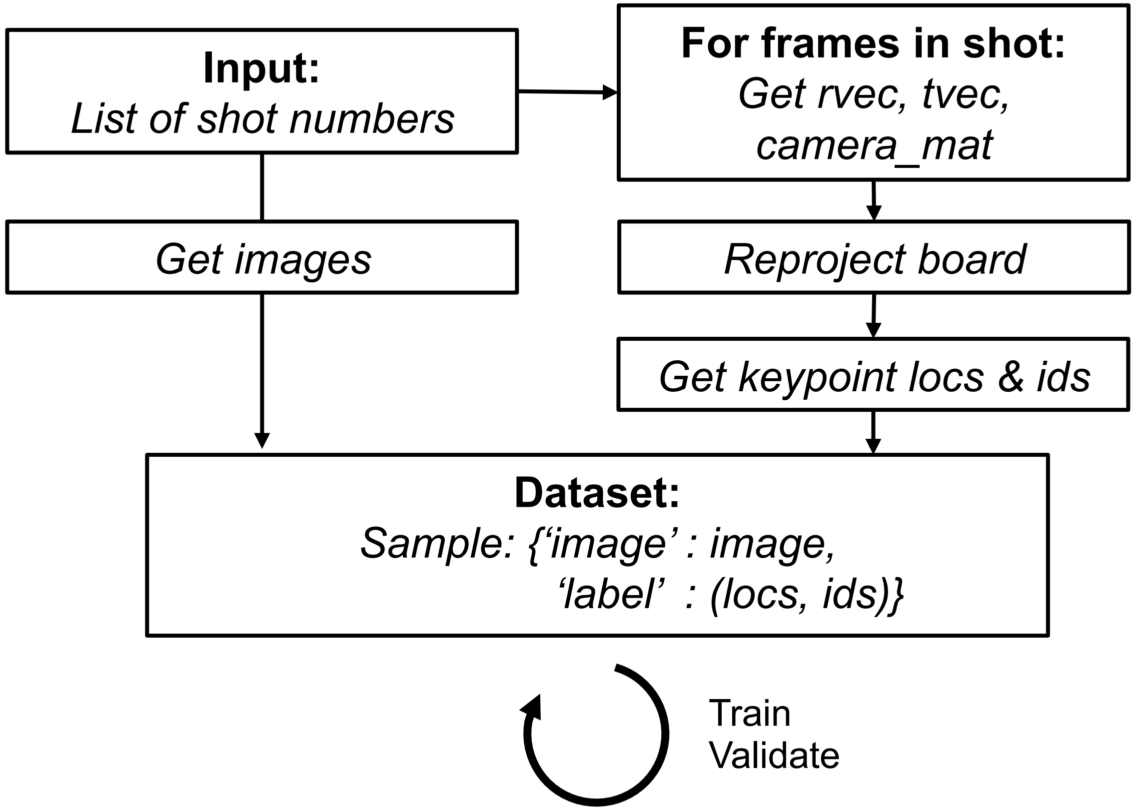


Camera OBSERVER branch

ChArUcoNet Training

Dataset:

Compute keypoint locations and ids from rotation and translation vectors and add as sample.



Model Training Details	
Trained for	50 epochs (~40hrs)
Trained on	NVIDIA A100
Image size	800 x 600
Training set - (i) (Original shots)	15,129 (21 shots)
Training set – (ii) (Noise-added shots)	15,129 (21 shots)
Training set - (iii) (Augmented images)	16,000 (20 shots)
Training set	46,258 (62 shots)
Validation set	1,400 (2 shots)
Test set	734 (1 shot)

ChArUcoNet performance

Key Metrics

1. Percentage of Detected Keypoints (PDK):
Ratio of identified keypoints within 3 pixel distance from label.

$$PDK = \frac{\sum_{i=1}^{9} bool(d_i < 3\,px)}{9} * 100$$

2. Mean L2 error of keypoints (L2):
Average Euclidean distance between identified kepoints and label.

$$L2 = \frac{\sum_{i=1}^{9} \|\boldsymbol{x_i} - \hat{\boldsymbol{x}}_i\|_2}{9}$$

3. Percentage of Estimated Frames (PEF):
Ratio of frames with more than 3 identified keypoints.

$$PEF = \frac{\sum_{j=1}^{n_f} bool(PDK_j > 0.33)}{n_f} * 100$$

Inference Speed	
CPU	0.7278 sec/frame **(x 5.7)**
	11:49 (Total test set) **(x1.7)**
GPU	0.1275 sec/frame (~8Hz)
	6:55 (Total test set)

Metric	ChArUcoNet	OpenCV
PDK	95.44%	
L2	0.32 px	
PEF	**98.50%** **(723/734)**	95.32% (700/734)

Deep ChArUco implemented with MARTe2

MARTe2 implementation:

- Model inference becomes compiled to and implemented in C/C++
    - Pytorch C++ API
- MARTe2 pyGAM exists but switching context (python ↔ cuda) for GPU needs to be resolved with Python GIL.

→ MARTe2 GAM to generalize deployment of neural net models with GPU inference

Data transfer and hardware implementation:

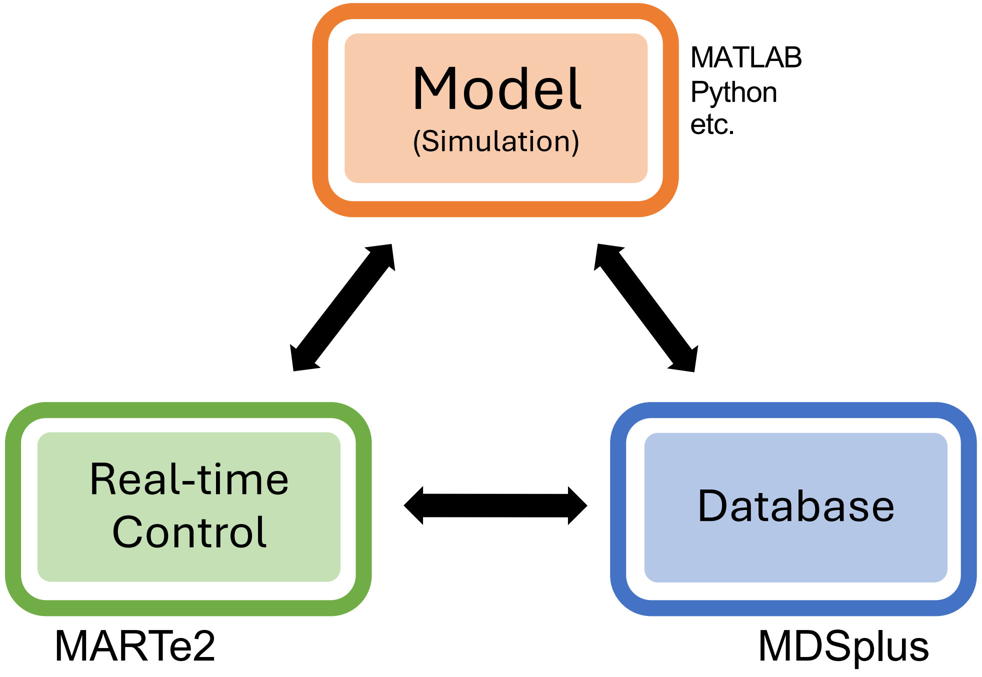
- Need to handle large data packets in real-time
    - Crop the image for faster rate around region of interest
    - Lower frame rate

→ Multi-rate implementation with camera as slow, high accuracy observer and ToF sensors as fast, low accuracy observer

Key Remarks

Developing a high-fidelity model shouldn't be an independent task but rather built on top of low-fidelity models with fast iteration.

1. **Robust management of data and configurations.**
   - High 'yield' of quality data used for training.
   - Seamless use of data for code-based model development.
2. **Efficient implementation from development**
   - Training is essentially the same as inference.
   - Switching Pytorch models like the Simulink models.
3. **Modular, distributed system components**
   - New components can be developed in parallel.
   - Data from testing one model can be used for training another.



Future Steps

Continue to build data-driven models and explore applications in modern control theory such as:

- Integrate vision based observer (MIMO multi-rate Kalman Filter, sensor fusion)
- Nonlinear control (adaptive, sliding, etc.)
- Learning-based physics simulator
- Data-driven, modern control methods
- etc…

References

[1] Simonyan, K., and Zisserman, A., 2015, “Very Deep Convolutional Networks for Large-Scale Image Recognition.”

[2] DeTone, D., Malisiewicz, T., and Rabinovich, A., 2018, “SuperPoint: Self-Supervised Interest Point Detection and Description.”

[3] Manduchi, G., Rigonii, A., Fredian, T. W., Stillerman, J. A., Neto, A., and Sartori, F., 2020, “MARTe2 and MDSplus Integration for a Comprehensive Fast Control and Data Acquisition System,” Fusion Engineering and Design, 161, p. 111892.

[4] Manduchi, G., Luchetta, A., Taliercio, C., Neto, A., Sartori, F., and De Tommasi, G., 2015, “Integration of Simulink, MARTe and MDSplus for Rapid Development of Real-Time Applications,” Fusion Engineering and Design, 96–97, pp. 645–648.

[5] Garnier, J. S. D., and Ferron, G. M. N., “First Applications of MARTe2/MDSplus/Simulink Framework for Real-Time Control Applications.”

[6] D, A., 2024, “JunkyByte/Deepcharuco,” GitHub [Online]. Available: https://github.com/JunkyByte/deepcharuco.git.

[7] Hu, D., DeTone, D., and Malisiewicz, T., 2019, “Deep ChArUco: Dark ChArUco Marker Pose Estimation,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Long Beach, CA, USA, pp. 8428–8436.

Q&A

Thank you!

We gratefully acknowledge Eni S.p.A. and MITEI for funding this research

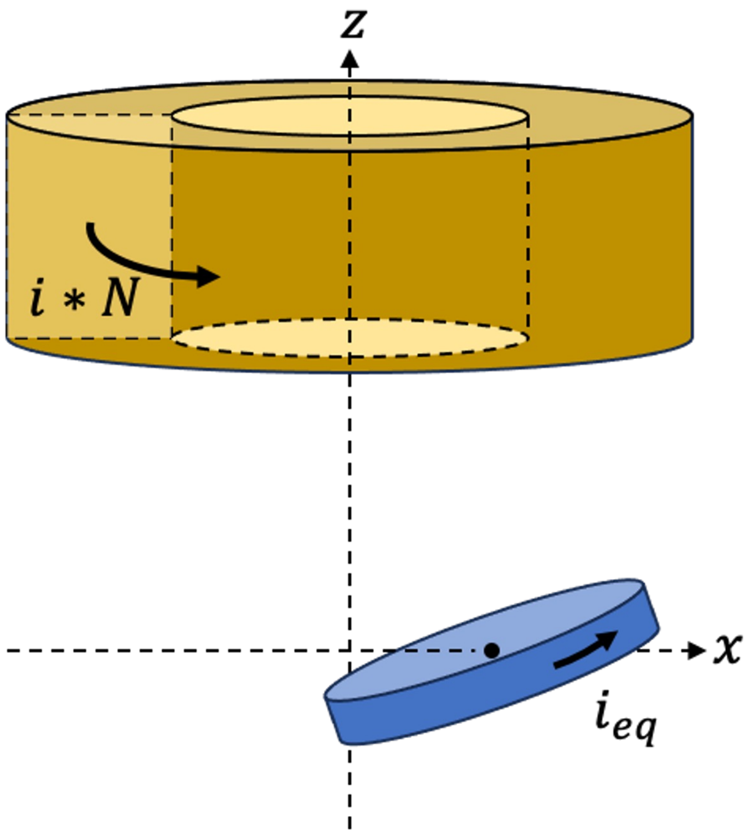
Massachusetts Institute of Technology PSFC Plasma Science and Fusion Center

CONSORZIO RFX Ricerca Formazione Innovazione

MITei

eni

Appendix 1 - State space model of the Bagel



System model

1. Energy equation

$$L = T_{mech} + T_{mag} - U_g$$
$$T_{mech} = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + \frac{1}{2}I(\dot{\alpha}^2 + \dot{\beta}^2)$$
$$U_g = -mgz$$
$$T_{mag} = \frac{1}{2}L_1 i_1^2 + \frac{1}{2}L_2 i_2^2 + M_{12} i_1 i_2$$

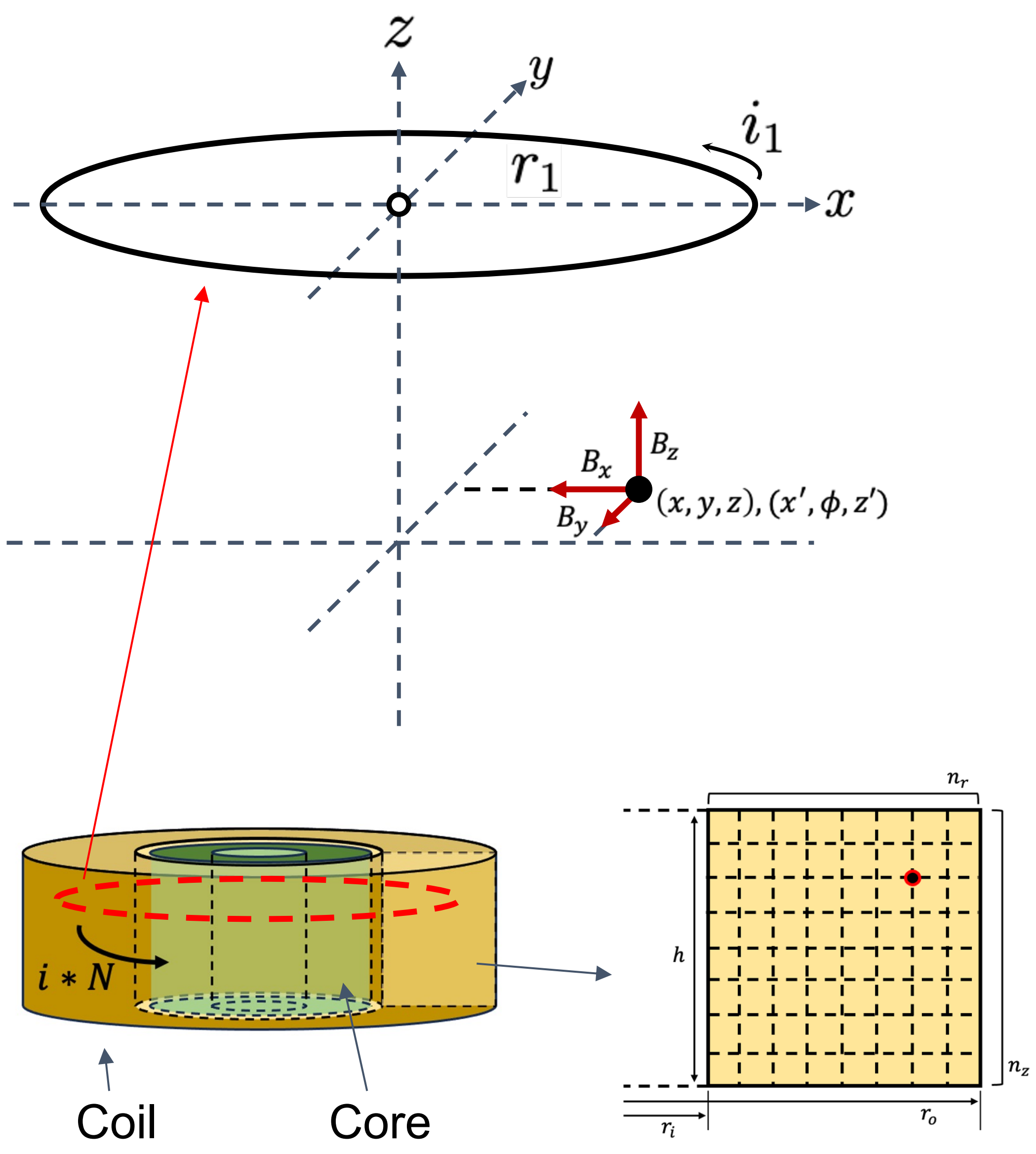
2. Magnetic Energy

$$M_{12} = \frac{\Phi_{12}}{i_1}, \quad B = \nabla \times A$$

$$\Phi_{12} = \oiint_{S_2} \vec{B}_1 \cdot \hat{n}\, dS = \oiint_{S_2} (\nabla \times \vec{A}_1) \cdot \hat{n}\, dS = \oint_{C_2} \vec{A}_1 \cdot \vec{dl}$$

Apply linearization about initial position to Lagrangian.

Appendix 1



Governing law (Static Maxwell's equation)

$$\nabla^2 \vec{\mathbf{A}} = \vec{\mathbf{J}} \qquad \vec{\mathbf{B}} = \nabla \times \vec{\mathbf{A}}$$

Magnetic vector potential Current Magnetic field

1. Magnetic vector potential of ring current

$$k^2 = \frac{4 r_1 x'}{(r_1 + x')^2 + z'^2}$$

$$\vec{A}_\phi(r_1, 0, x', z') = \frac{\mu_o i_1}{4\pi} \frac{r_1}{\sqrt{(r_1 + x')^2 + z'^2}} \left[\frac{(2 - k^2)K(k^2) - 2E(k^2)}{k^2}\right] \cdot \hat{\phi}$$

2. Magnetic field of ring current

$$\vec{B} = \nabla \times \vec{A} = \frac{\partial \vec{A}_\phi}{\partial z} \hat{r} + \frac{1}{r} \frac{\partial \vec{A}_\phi}{\partial r} \hat{z}$$

$$\boldsymbol{B_{ring}(x, y, z) = f(r_1, i_1, x, y, z)}$$

$$\longrightarrow \boldsymbol{B_{ring} = \overline{B}_{ring}(x, y, z) * i}$$

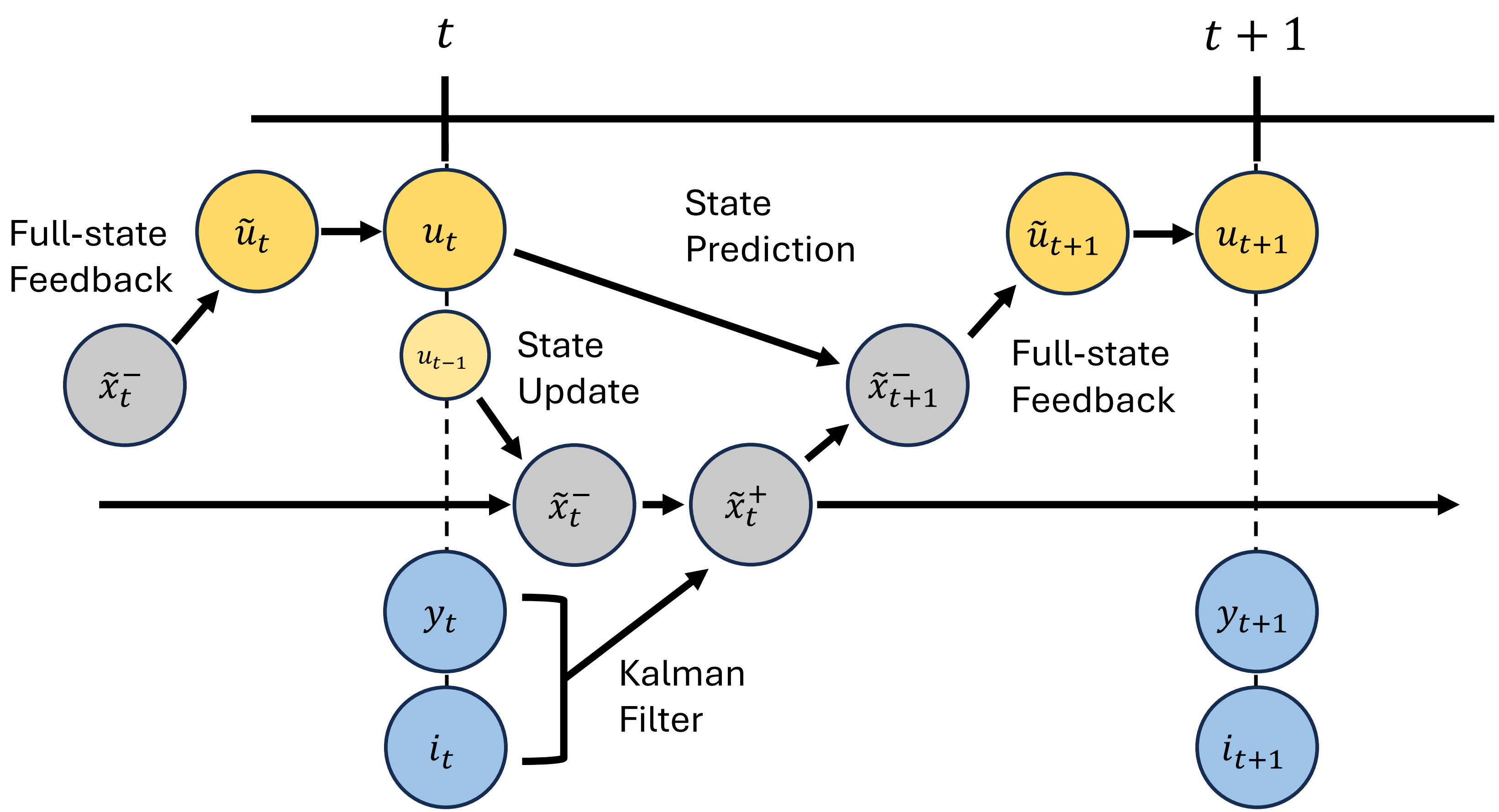
3. Discretized model for coil

Apply function to Euler-Lagrange equation

n_r, n_z: *Number of coil discretization*

$$B_{coil}(x_0, y_0, z_0)$$
$$= \sum_i^{nz} \sum_j^{nx} \overline{B}_{ring}(r[j], x_{,0}\ y_0, z_0 - z[i]) * i$$

Appendix 2 - Prediction-based control within MARTe2



Controller computation sequence