This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Accelerator R&D and Production award number DE- SC0021555

A Front-End Framework with Embedded ML Tools for Automating Neutron Scattering Experiments

M. Henderson, J. Edelen, M. Kilpatrick (RadiaSoft LLC)

S. Calder, C. Hoffmann, B. Krishna, G. Taufer, B. Vacaliuc (ORNL)

ICFA ML 2024

Gyeongju, South Korea

March 6, 2024

Outline

- Motivation & background
  - Problem motivation
  - Partner beamlines
  - Image segmentation tasks
- Automation methods
  - Controls software
  - Image recognition
- Results & discussion

Background & motivation

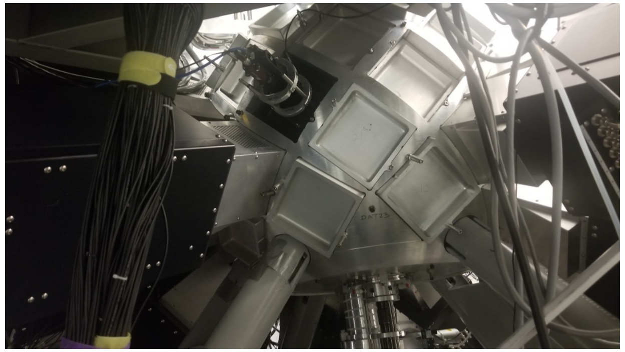
The value of automation on neutron scattering beamlines

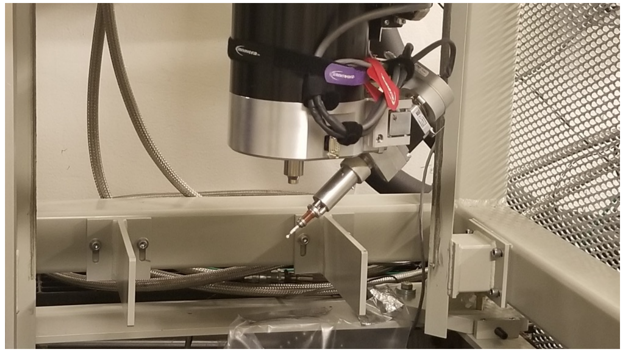
Background & Motivation

- Sample alignment is tedious, but critical
  - Currently requires human image processing
  - Limited neutron production time, schedule constraints
- Machine learning is a key automation tool
  - For computer vision, convolutional neural networks (CNNs)
  - For controls automated, reinforcement learning (RL)
- Alignment protocols are distinct to a beamline
  - Framework must be highly general & robust
  - Opportunity to deploy transfer learning

Partner Beamlines: TOPAZ

- Fed by the Spallation Neutron Source (SNS)
- Sample in a chamber
    - Sample arm with 3 translational, 2 rotational axes
    - Neutron detectors, cameras, & environmental controls
- Highly developed controls system (EPICS)
    - Plenty of control channels (PVs already exist)
    - Pre-existing IOC software
        - Includes point-and-click sample alignment
        - Must consider interactions with our framework





Partner Beamlines: HB-2A

- Fed by the High-Flux Isotope Reactor (HFIR)
- Samples in containers (cans) on a stage
  - 3 translational axes, stage & sample rotation axes
  - Partially encircled by a neutron detector
  - Diagnostics done with a **neutron** camera
- Controls executed with SPICE (no EPICS)
  - Required a "bridge" between software to use EPICS
  - Sample alignment completely manual

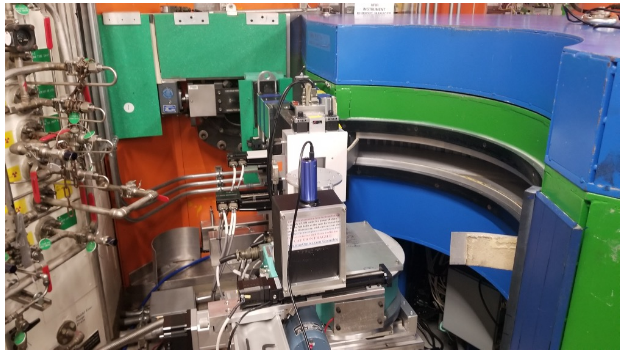
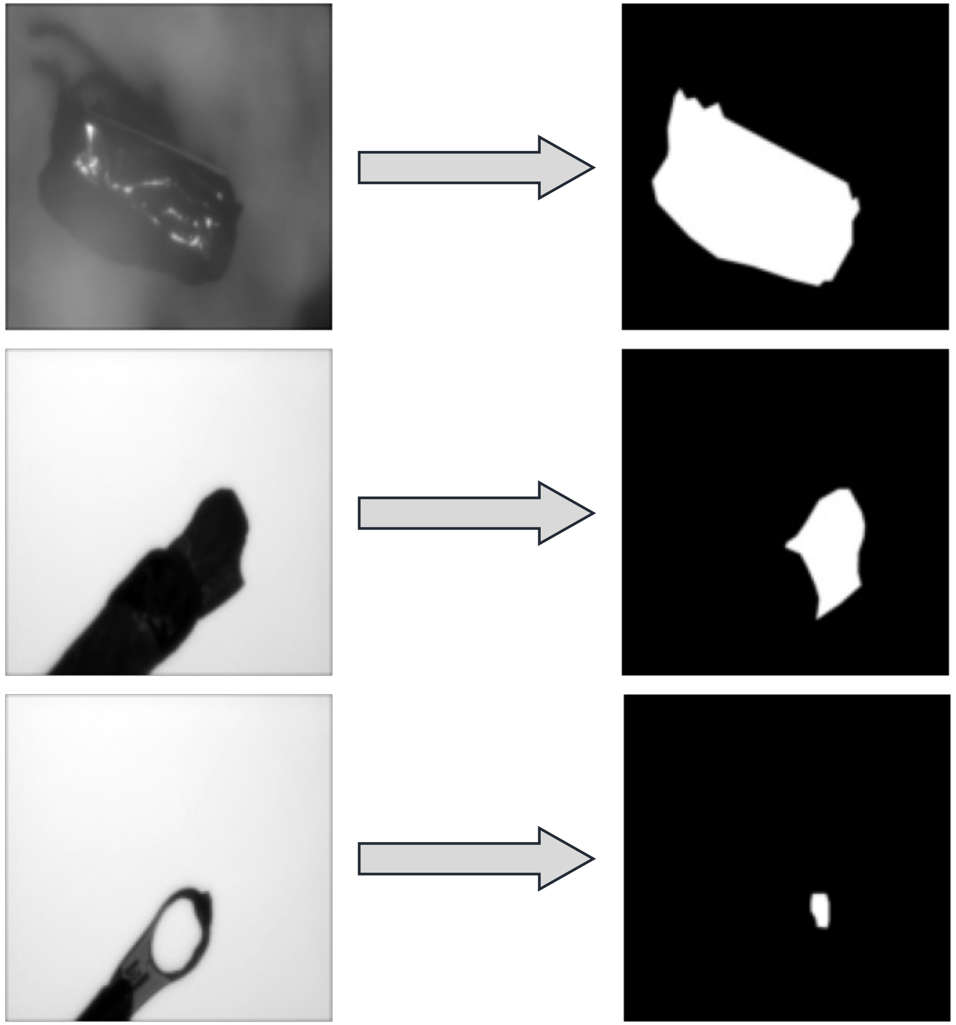
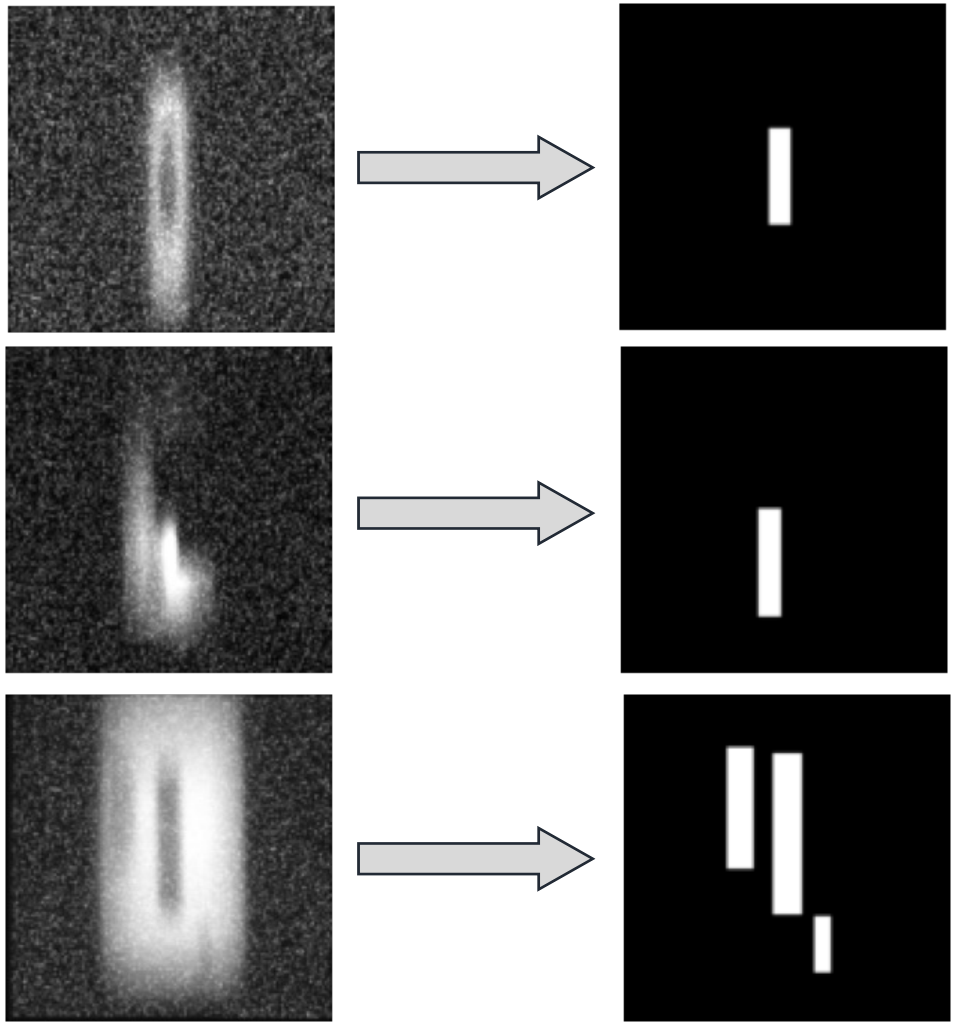


Image Segmentation Tasks

TOPAZ



HB2A



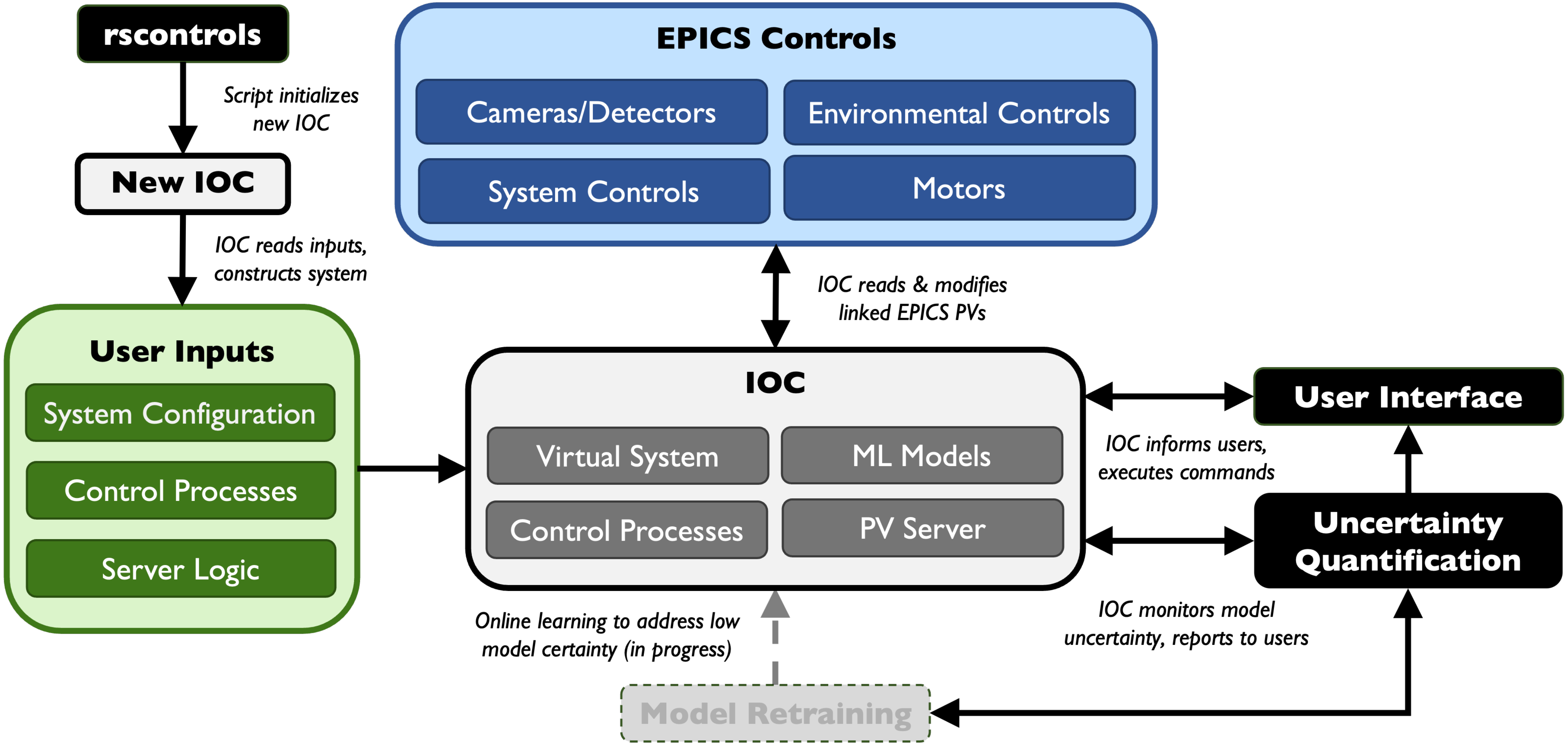
Automation methods

Approaches for removing humans from the loop

Controls Software

- Require a means to deploy developed automation tools
  - Machine learning models, control loops (e.g., PID), etc.
  - Must be flexible to apply broadly to controls environments
- Development & testing resulted in `rscontrols`
  - Written in Python using existing EPICS tools (PyEpics, PCASPy, & P4P)
  - Hardware (detectors, motors, etc.) represented in thin virtualization layer
    - Connected via EPICS process variables (PVs)
  - Handles for user-defined controls processes & served PVs
    - Written & imported as Python modules, on the fly
  - Dedicated embeddings for ML models

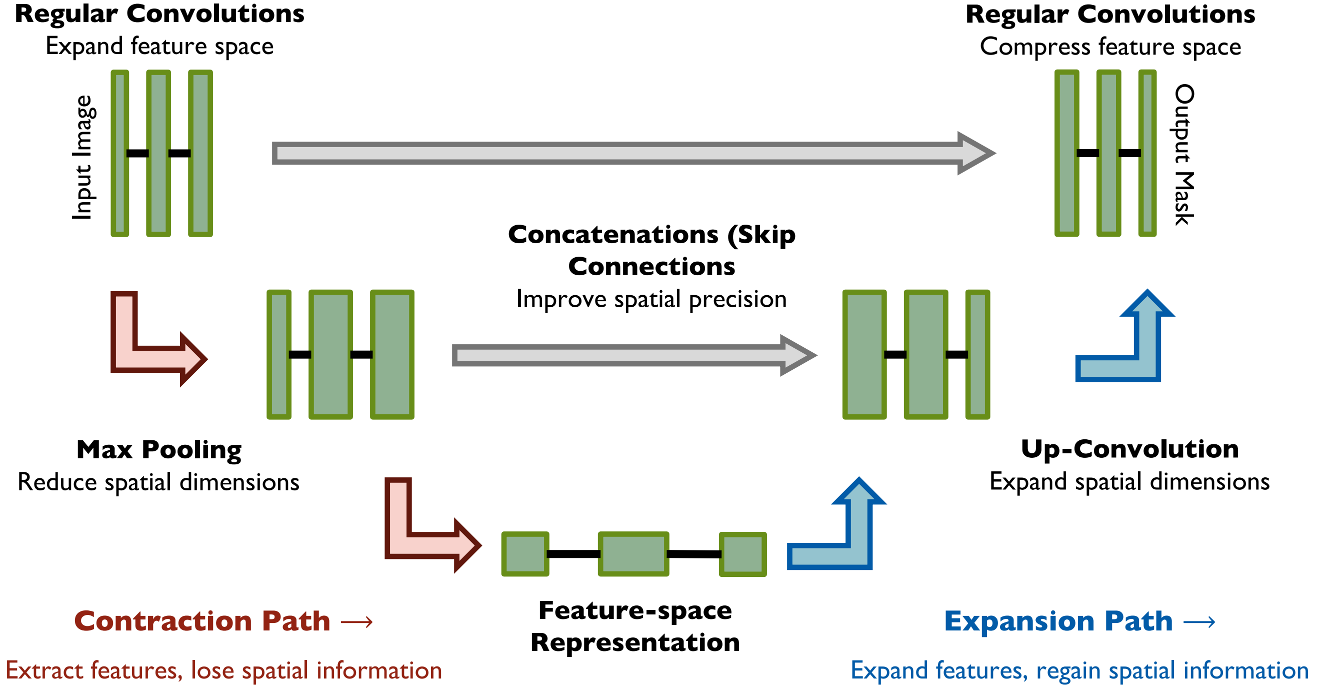
rscontrols Workflow



rscontrols
Script initializes new IOC
New IOC
IOC reads inputs, constructs system
User Inputs
System Configuration
Control Processes
Server Logic
EPICS Controls
Cameras/Detectors
Environmental Controls
System Controls
Motors
IOC reads & modifies linked EPICS PVs
IOC
Virtual System
ML Models
Control Processes
PV Server
User Interface
IOC informs users, executes commands
Uncertainty Quantification
IOC monitors model uncertainty, reports to users
Online learning to address low model certainty (in progress)
Model Retraining

Image Recognition

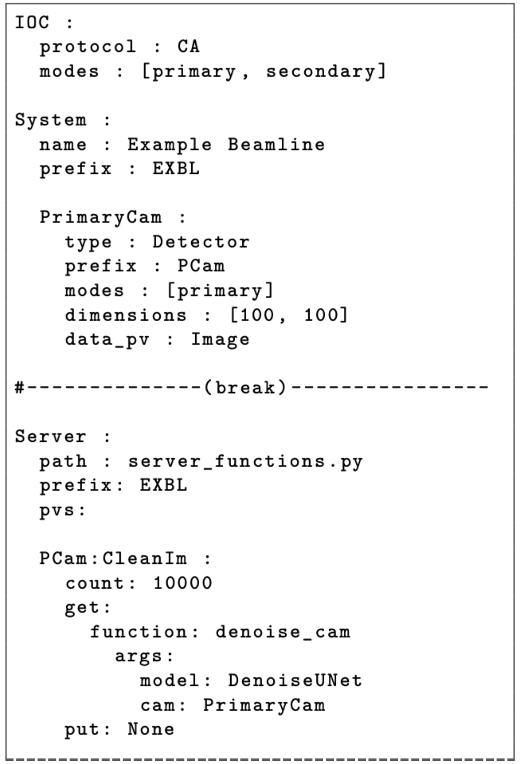
- Classic use-case for convolutional neural networks (CNNs)
  - UNet specifically designed for image & semantic segmentation
  - Labor-intensive supervised learning regimen
- Must be robust to use cases
  - No sample (TOPAZ & HB-2A)
  - Multiple samples (HB-2A)


Regular Convolutions
Expand feature space
Input Image
Regular Convolutions
Compress feature space
Output Mask
Concatenations (Skip Connections
Improve spatial precision
Max Pooling
Reduce spatial dimensions
Up-Convolution
Expand spatial dimensions
Contraction Path →
Extract features, lose spatial information
Feature-space Representation
Expansion Path →
Expand features, regain spatial information

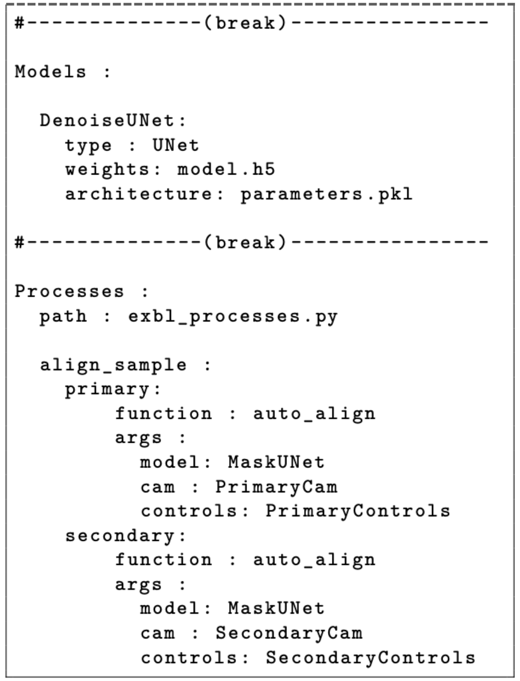
Results

`rscontrols` UI, model uncertainties, and sample alignment

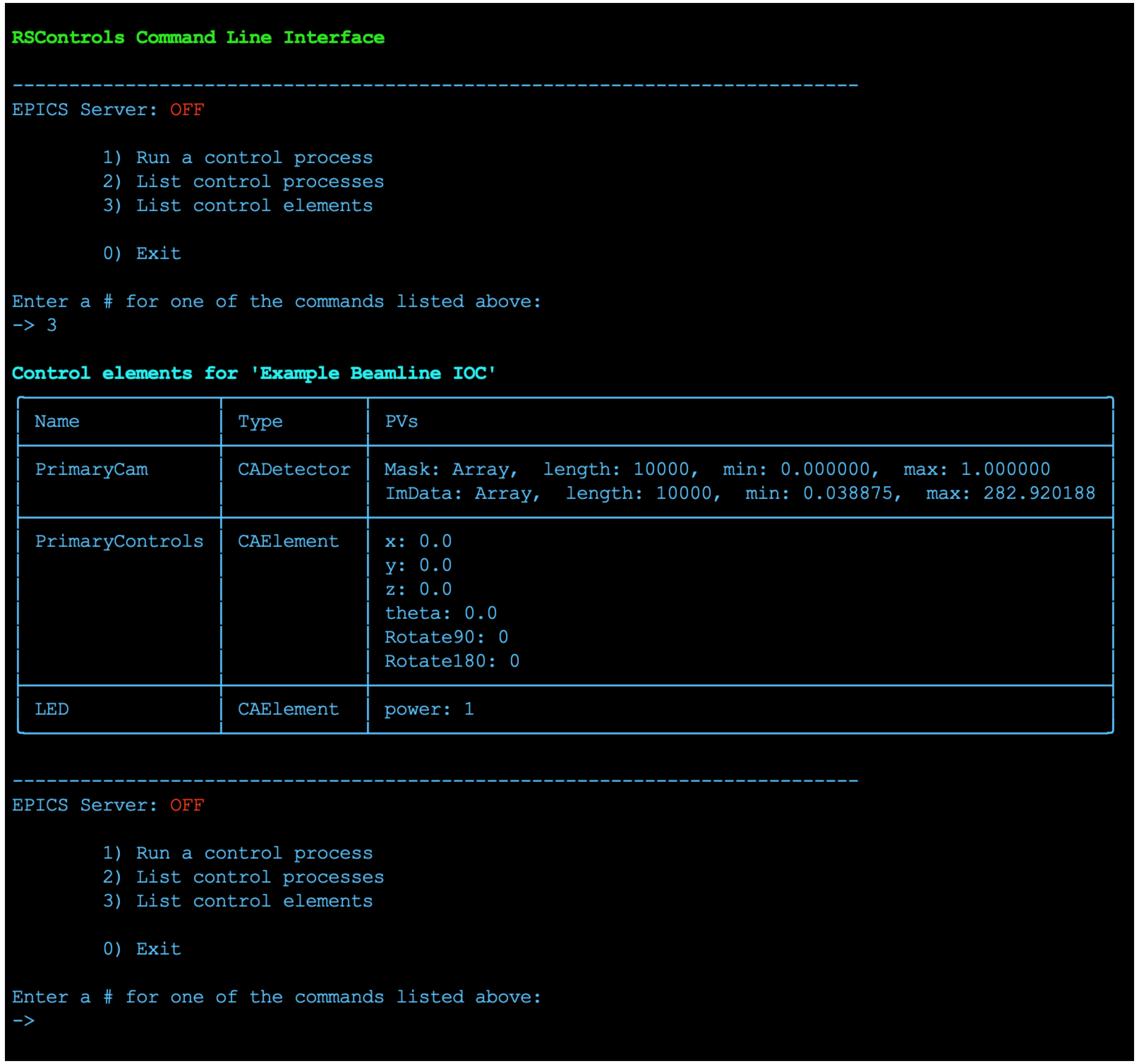
rscontrols, Simulated Beamline Example



```
IOC :
  protocol : CA
  modes : [primary, secondary]

System :
  name : Example Beamline
  prefix : EXBL

  PrimaryCam :
    type : Detector
    prefix : PCam
    modes : [primary]
    dimensions : [100, 100]
    data_pv : Image

#--------------(break)----------------

Server :
  path : server_functions.py
  prefix: EXBL
  pvs:

  PCam:CleanIm :
    count: 10000
    get:
      function: denoise_cam
        args:
          model: DenoiseUNet
          cam: PrimaryCam
    put: None
```

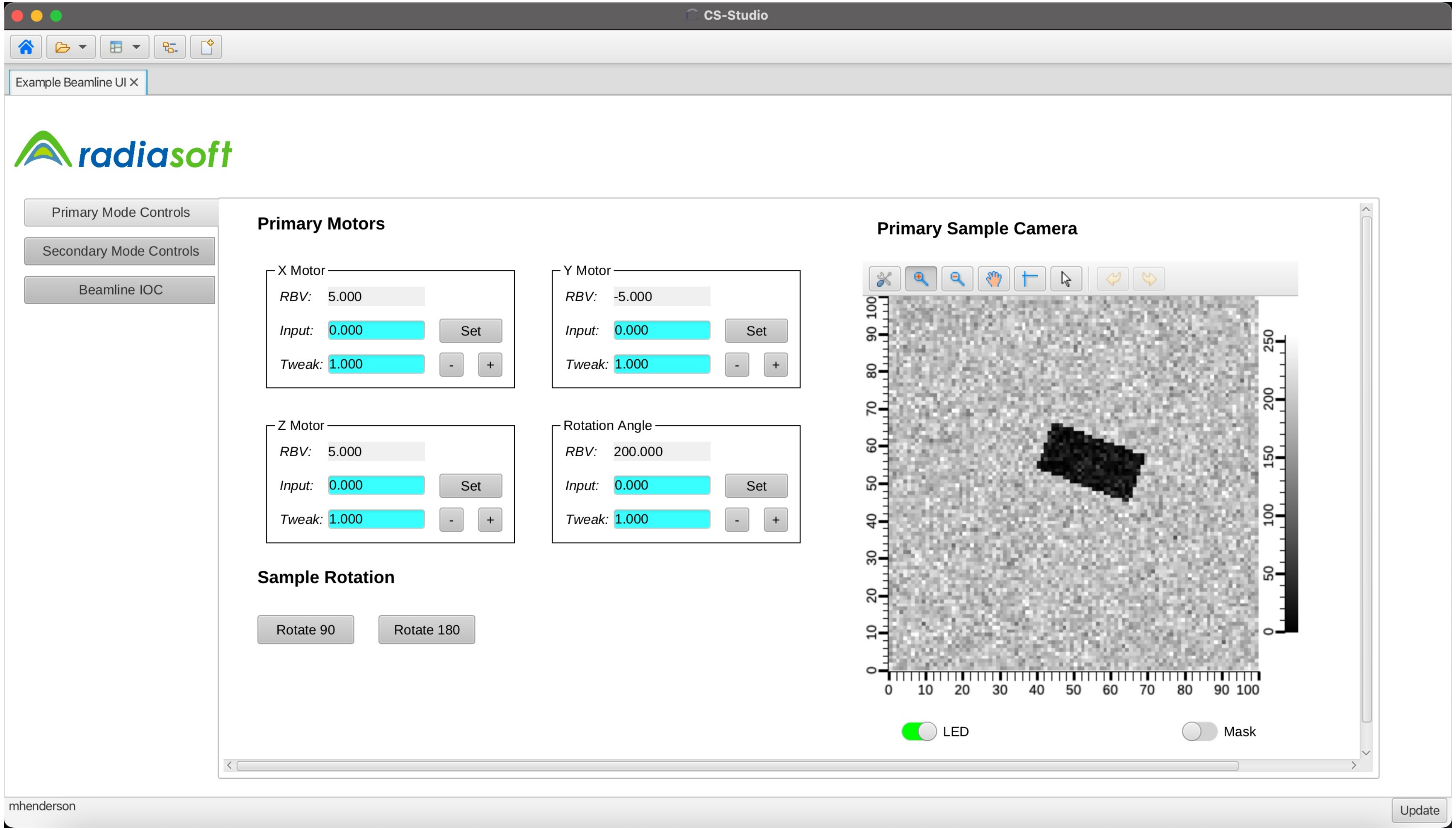


```
#--------------(break)----------------

Models :

  DenoiseUNet:
    type : UNet
    weights: model.h5
    architecture: parameters.pkl

#--------------(break)----------------

Processes :
  path : exbl_processes.py

  align_sample :
    primary:
        function : auto_align
        args :
          model: MaskUNet
          cam : PrimaryCam
          controls: PrimaryControls
    secondary:
        function : auto_align
        args :
          model: MaskUNet
          cam : SecondaryCam
          controls: SecondaryControls
```



```
RSControls Command Line Interface

---------------------------------------------------------------------------
EPICS Server: OFF

        1) Run a control process
        2) List control processes
        3) List control elements

        0) Exit

Enter a # for one of the commands listed above:
-> 3

Control elements for 'Example Beamline IOC'
 ┌─────────────────┬────────────┬──────────────────────────────────────────────────────────────┐
 │ Name            │ Type       │ PVs                                                          │
 ├─────────────────┼────────────┼──────────────────────────────────────────────────────────────┤
 │ PrimaryCam      │ CADetector │ Mask: Array,  length: 10000,  min: 0.000000,  max: 1.000000  │
 │                 │            │ ImData: Array,  length: 10000,  min: 0.038875,  max: 282.920188 │
 ├─────────────────┼────────────┼──────────────────────────────────────────────────────────────┤
 │ PrimaryControls │ CAElement  │ x: 0.0                                                       │
 │                 │            │ y: 0.0                                                       │
 │                 │            │ z: 0.0                                                       │
 │                 │            │ theta: 0.0                                                   │
 │                 │            │ Rotate90: 0                                                  │
 │                 │            │ Rotate180: 0                                                 │
 ├─────────────────┼────────────┼──────────────────────────────────────────────────────────────┤
 │ LED             │ CAElement  │ power: 1                                                     │
 └─────────────────┴────────────┴──────────────────────────────────────────────────────────────┘

---------------------------------------------------------------------------
EPICS Server: OFF

        1) Run a control process
        2) List control processes
        3) List control elements

        0) Exit

Enter a # for one of the commands listed above:
->
```

rscontrols, CS Studio UI



Primary Motors

X Motor

RBV: 5.000

Input: 0.000

Tweak: 1.000

Y Motor

RBV: -5.000

Input: 0.000

Tweak: 1.000

Z Motor

RBV: 5.000

Input: 0.000

Tweak: 1.000

Rotation Angle

RBV: 200.000

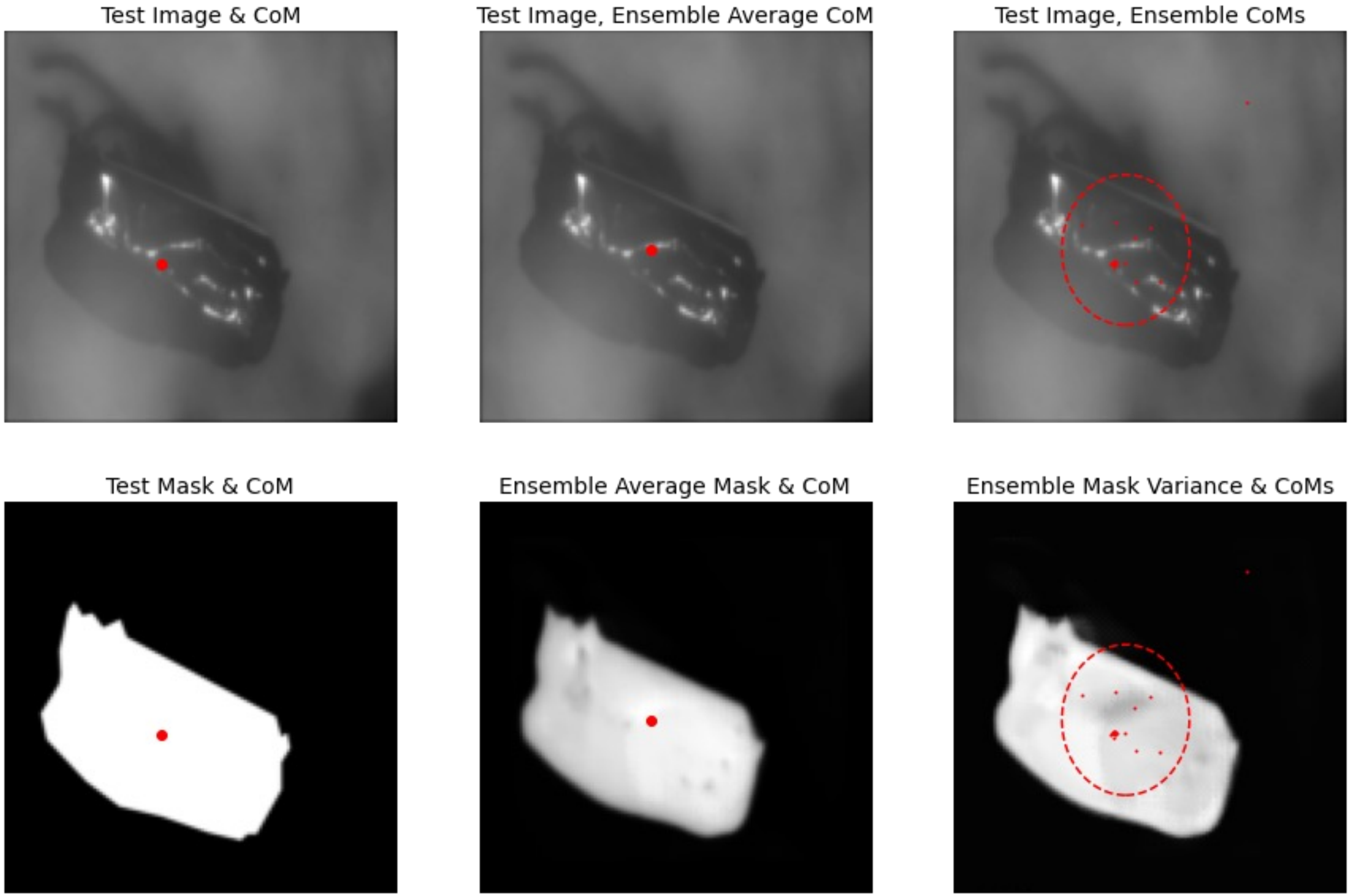
Input: 0.000

Tweak: 1.000

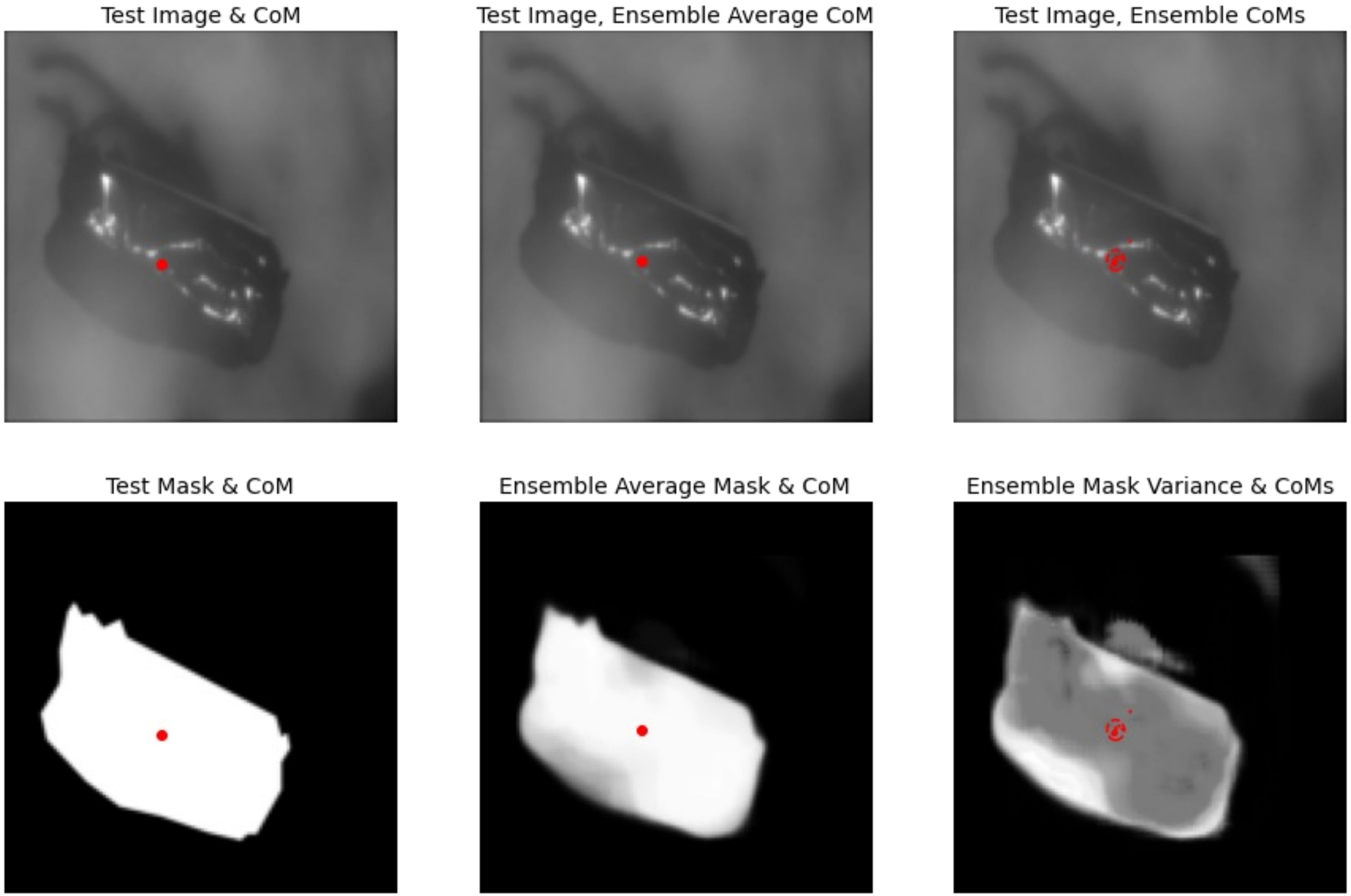
Sample Rotation

Primary Sample Camera

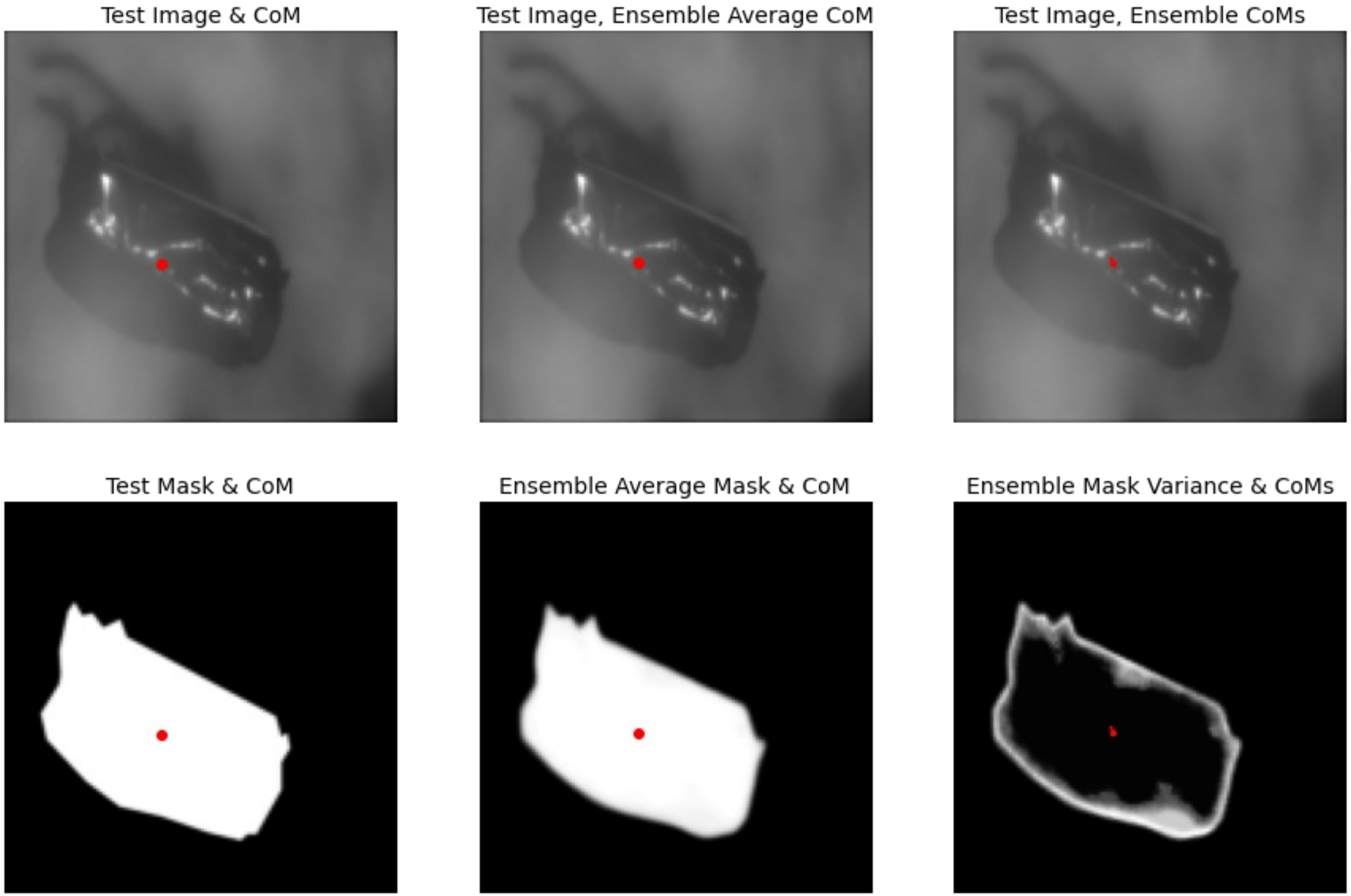
Unoptimized Ensemble Predictions


Test Image & CoM
Test Image, Ensemble Average CoM
Test Image, Ensemble CoMs
Test Mask & CoM
Ensemble Average Mask & CoM
Ensemble Mask Variance & CoMs

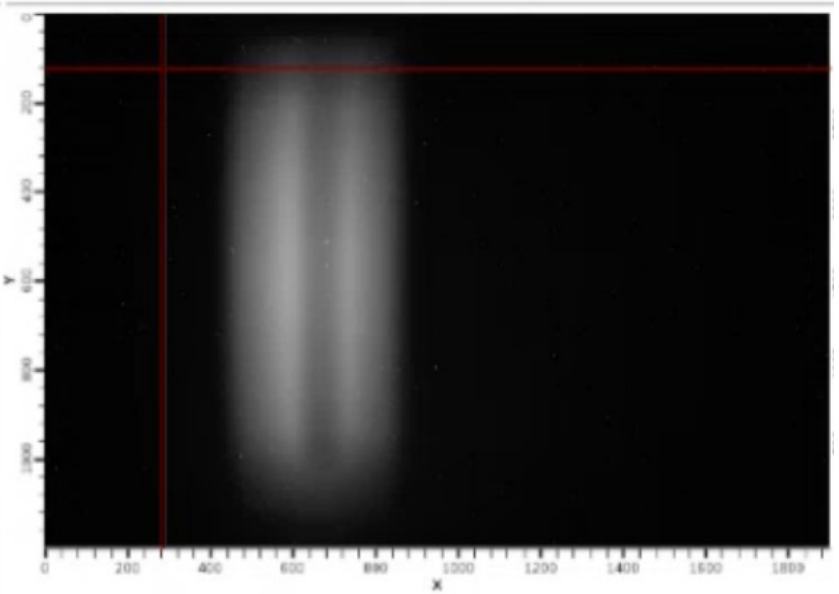
Optimized Ensemble Predictions


Test Image & CoM
Test Image, Ensemble Average CoM
Test Image, Ensemble CoMs
Test Mask & CoM
Ensemble Average Mask & CoM
Ensemble Mask Variance & CoMs

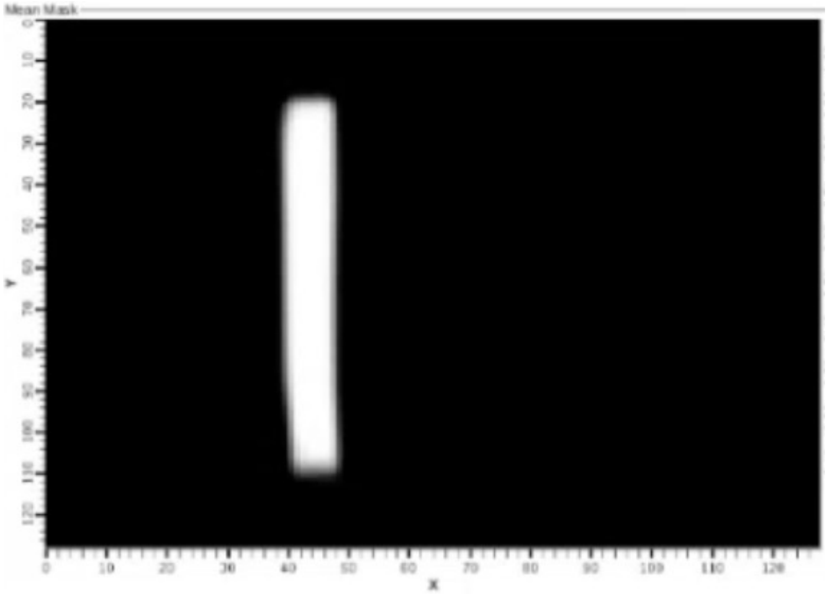
Optimized & Transferred Ensemble Predictions


Test Image & CoM
Test Image, Ensemble Average CoM
Test Image, Ensemble CoMs
Test Mask & CoM
Ensemble Average Mask & CoM
Ensemble Mask Variance & CoMs

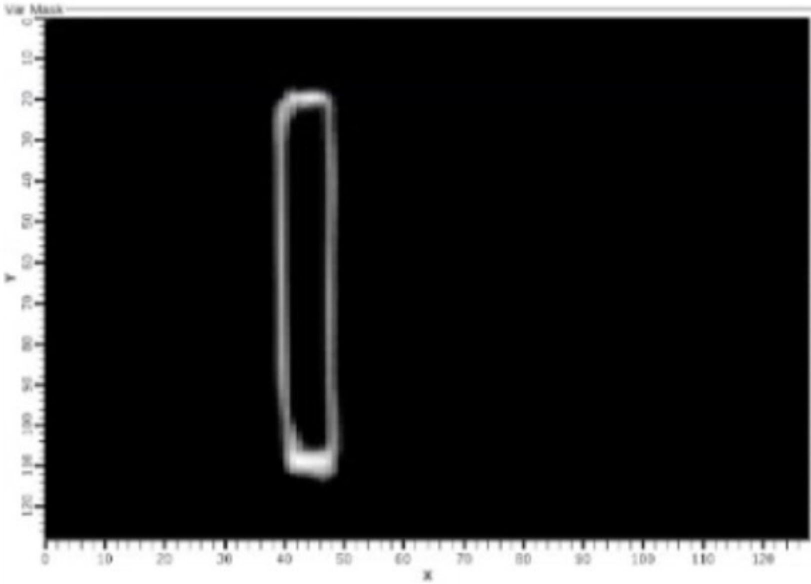
Live Uncertainty Reporting, HB-2A



Live Sample Image

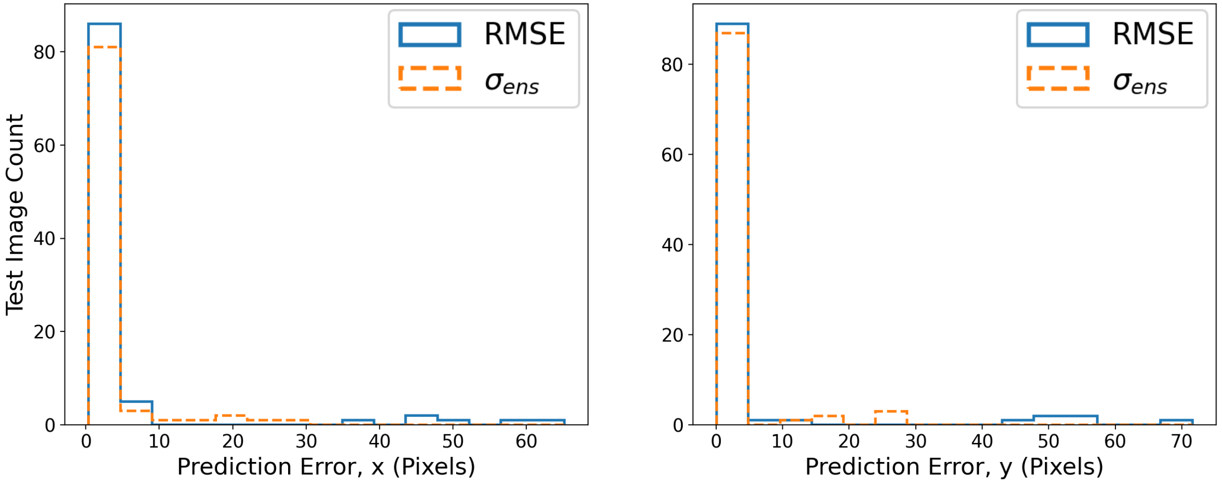


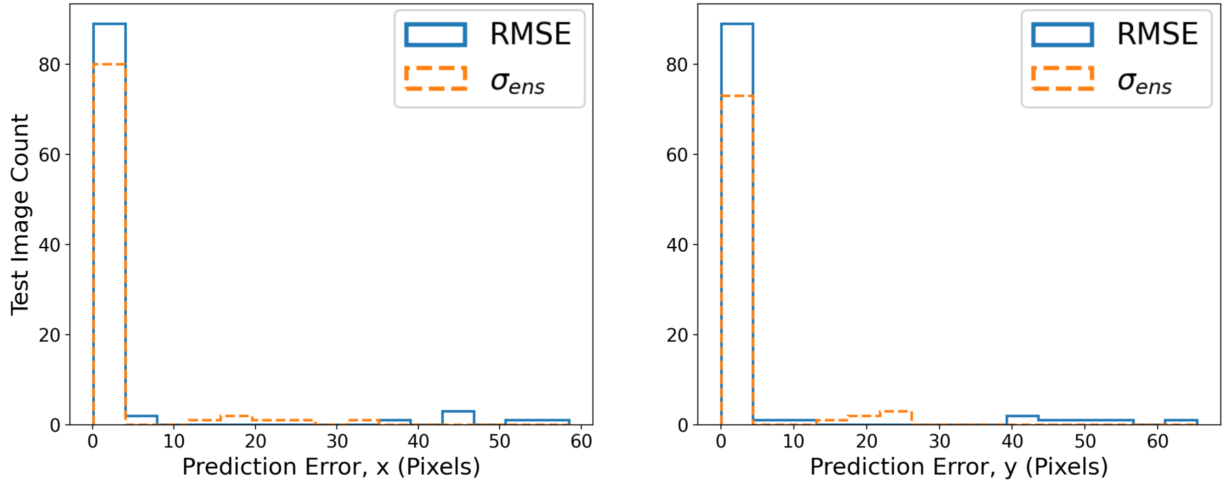
Ensemble Mask Average

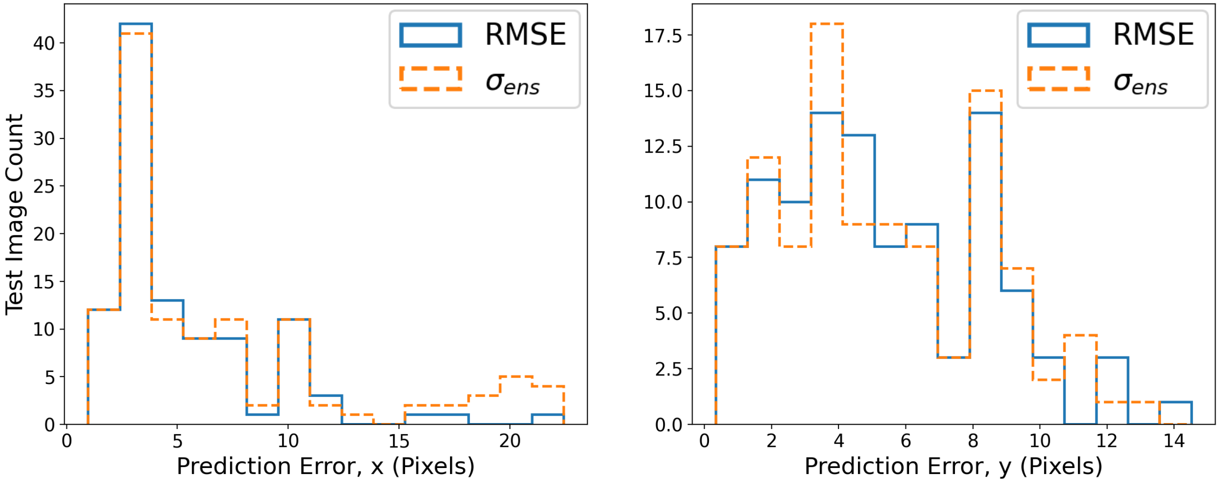


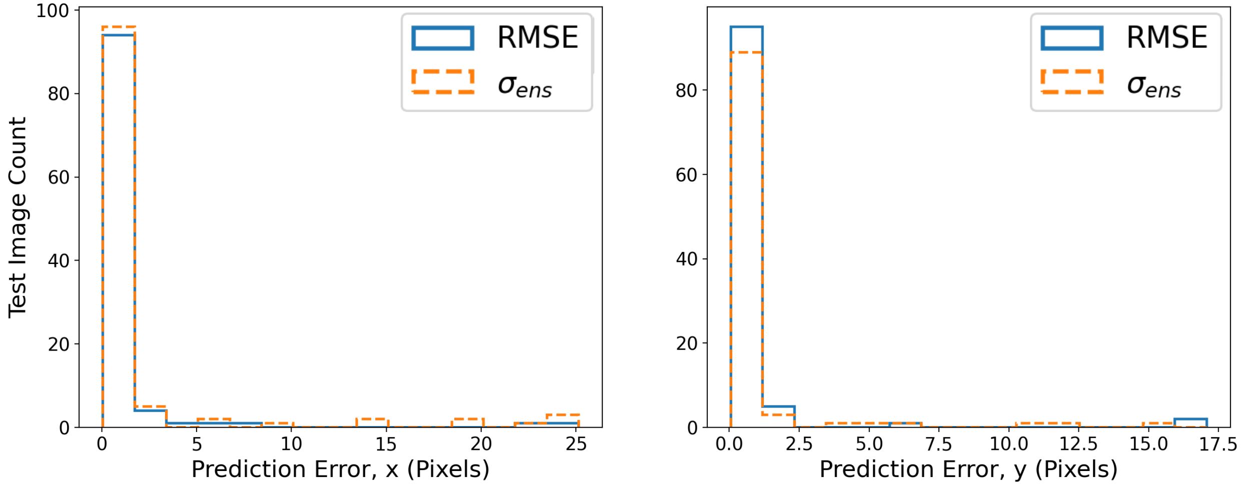
Ensemble Mask Variance

UNet Prediction Uncertainties

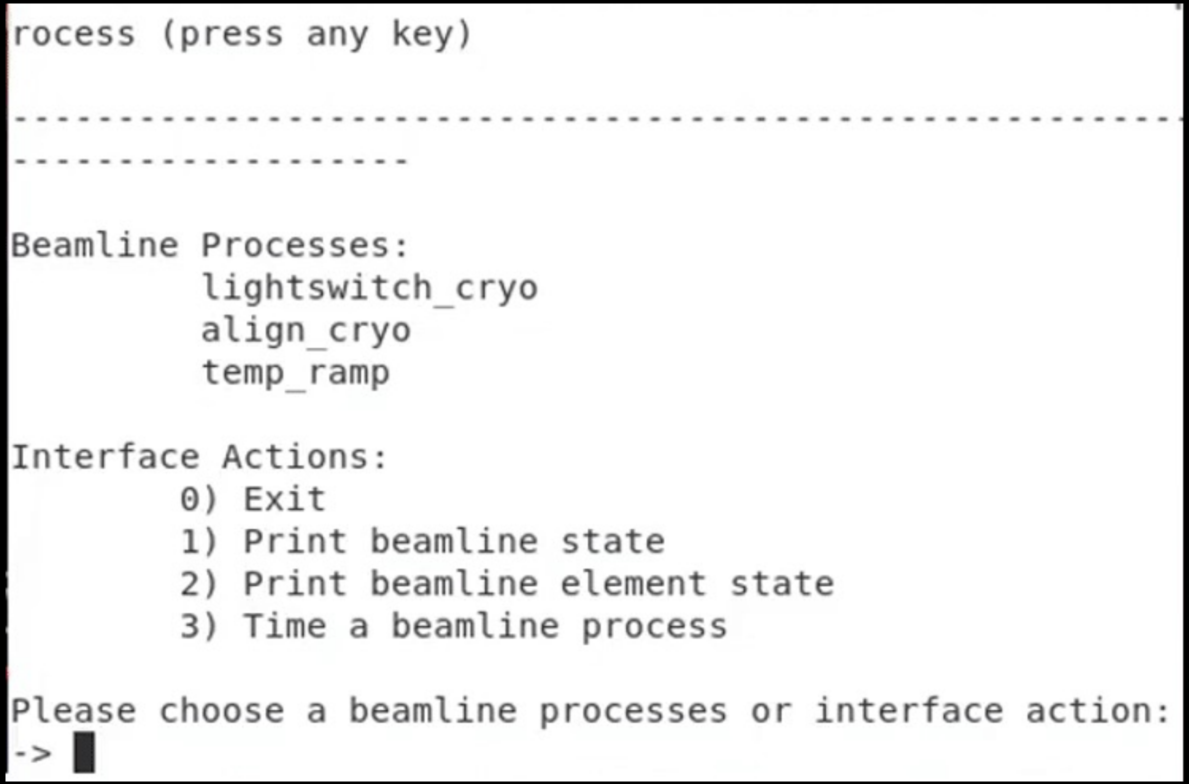

Prediction Uncertainties, Ensemble hh


Prediction Uncertainties, Ensemble hhh

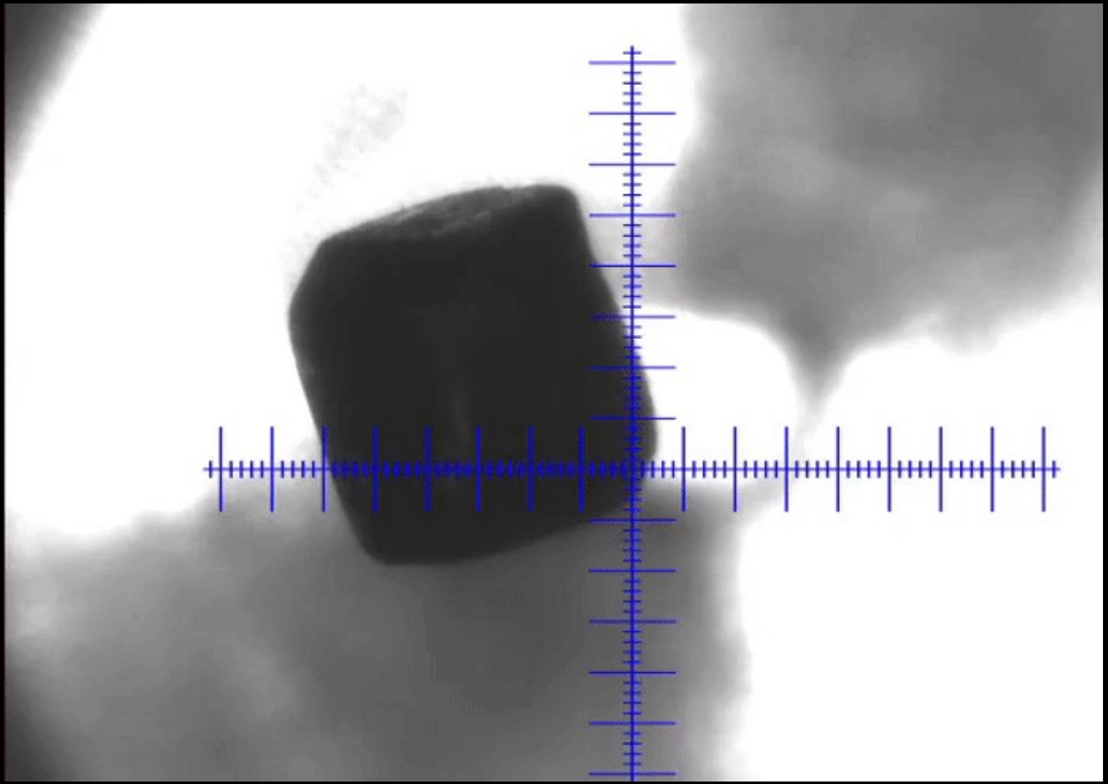

Prediction Uncertainties, Ensemble ht


Prediction Uncertainties, Ensemble hht

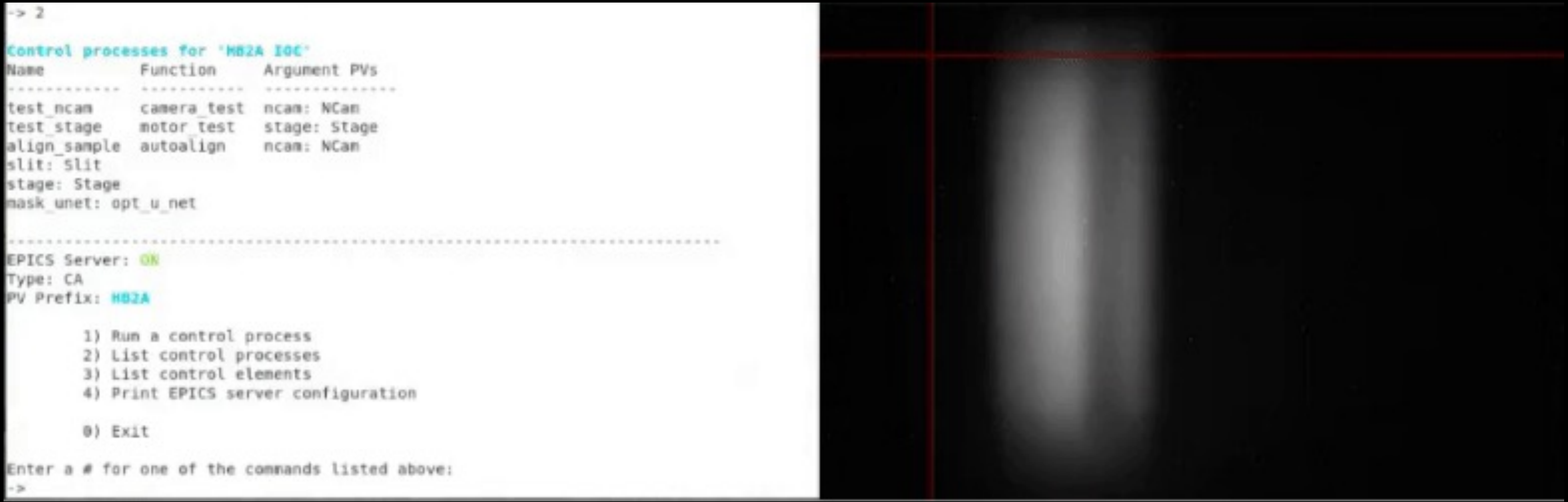
Automated Alignment, TOPAZ



```
rocess (press any key)

-------------------------------------------------------
-------------------

Beamline Processes:
         lightswitch_cryo
         align_cryo
         temp_ramp

Interface Actions:
        0) Exit
        1) Print beamline state
        2) Print beamline element state
        3) Time a beamline process

Please choose a beamline processes or interface action:
-> █
```



Automated Alignment, HB-2A



```
-> 2

Control processes for 'HB2A IOC'
Name          Function     Argument PVs
------------  -----------  --------------
test_ncam     camera_test  ncam: NCam
test_stage    motor_test   stage: Stage
align_sample  autoalign    ncam: NCam
slit: Slit
stage: Stage
mask_unet: opt_u_net

------------------------------------------------------------------------------
EPICS Server: ON
Type: CA
PV Prefix: HB2A

        1) Run a control process
        2) List control processes
        3) List control elements
        4) Print EPICS server configuration

        0) Exit

Enter a # for one of the commands listed above:
->
```

Ongoing & Future Efforts

- Further improve `rscontrols` UI
- Automate complete/more complex procedures
  - Temperature ramping with constant realignment at TOPAZ
  - Beam refinement at HB-2A
- Develop reinforcement learning solutions
  - Fuller automation at TOPAZ (less reliance on existing software)
  - Detector-based alignment at TOPAZ & HB-2A
- Synergize with RS 3D visualizer, also deployed at ORNL
- Deploy at more partner beamlines

Government Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Unused Slides

Denoising Results

Image denoising, regular filter

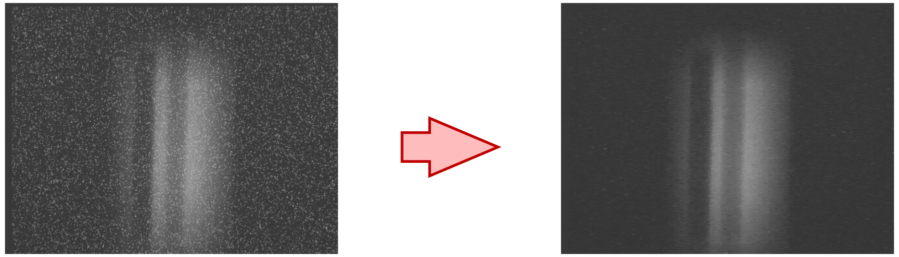
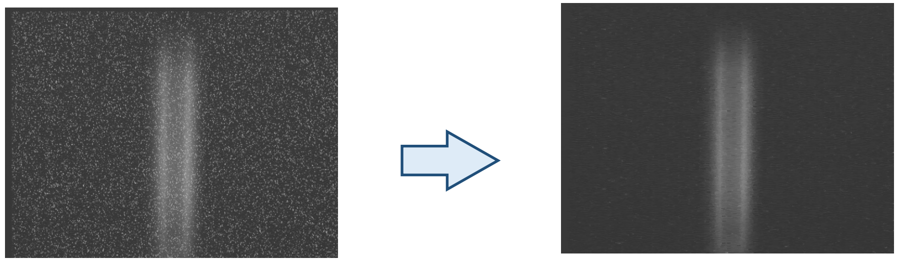
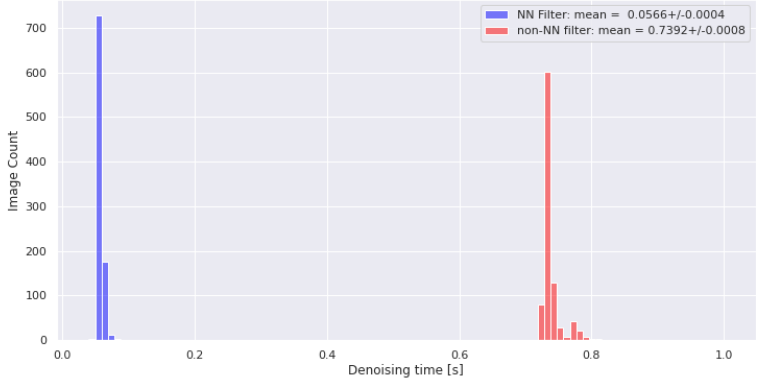
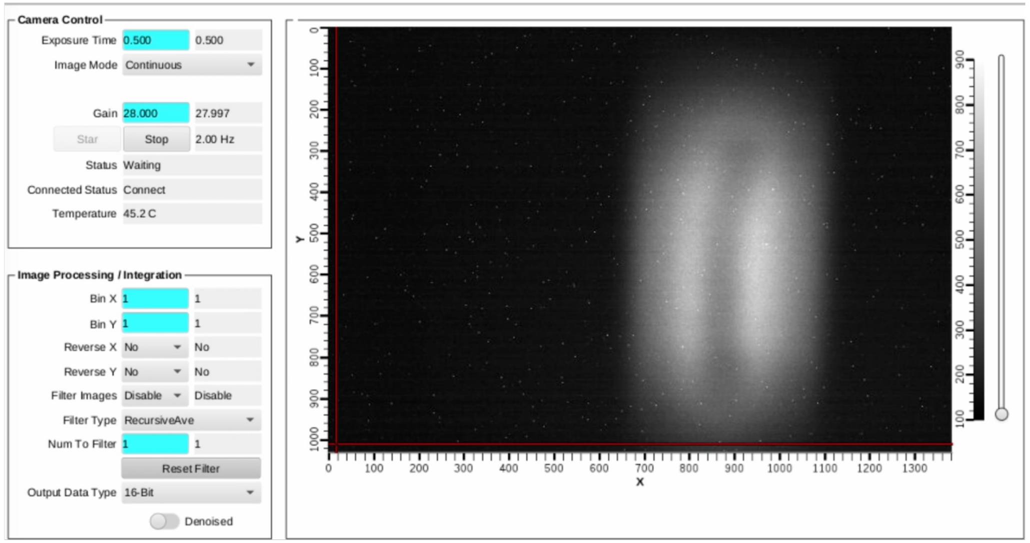


Image denoising, CNN filter



Execution time, regular vs. CNN filter





Camera Control

Exposure Time 0.500 0.500

Image Mode Continuous

Gain 28.000 27.997

Star Stop 2.00 Hz

Status Waiting

Connected Status Connect

Temperature 45.2 C

Image Processing / Integration

Bin X 1 1

Bin Y 1 1

Reverse X No No

Reverse Y No No

Filter Images Disable Disable

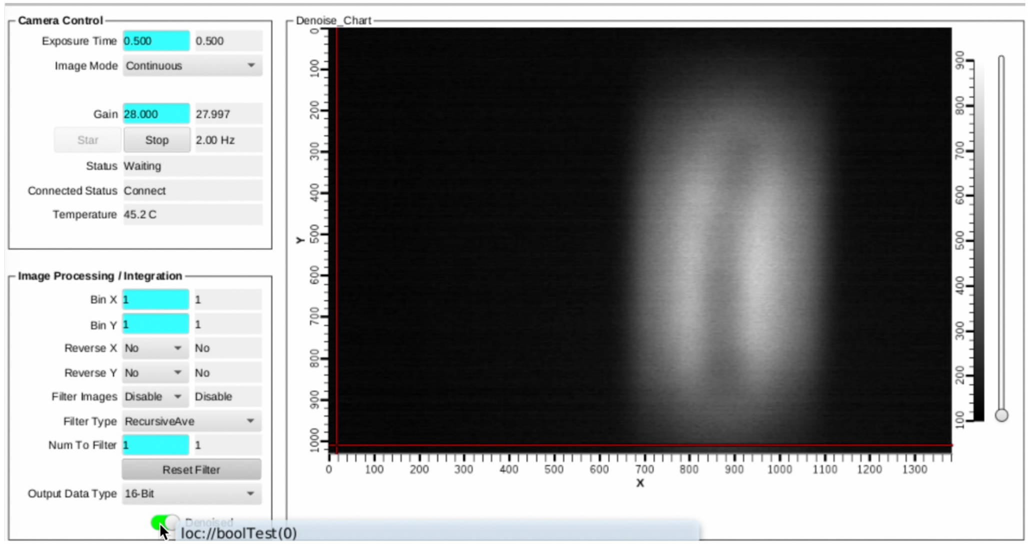
Filter Type RecursiveAve

Num To Filter 1 1

Reset Filter

Output Data Type 16-Bit

Denoised



Camera Control

Exposure Time 0.500 0.500

Image Mode Continuous

Gain 28.000 27.997

Star Stop 2.00 Hz

Status Waiting

Connected Status Connect

Temperature 45.2 C

Image Processing / Integration

Bin X 1 1

Bin Y 1 1

Reverse X No No

Reverse Y No No

Filter Images Disable Disable

Filter Type RecursiveAve

Num To Filter 1 1

Reset Filter

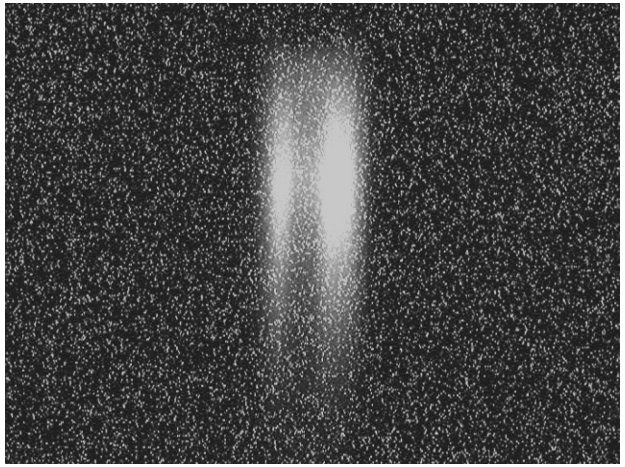
Output Data Type 16-Bit

Denoised

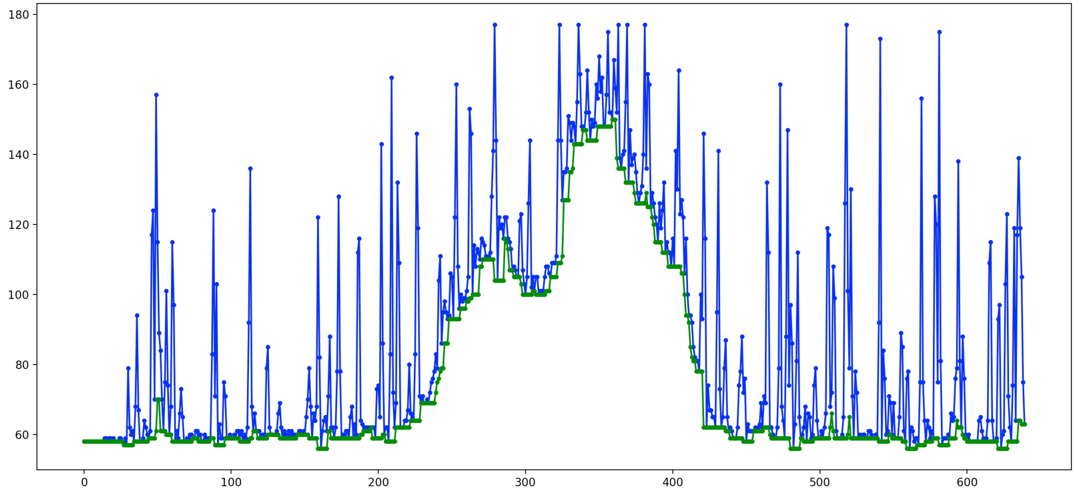
loc://boolTest(0)

Neutron Camera Denoising

- Neutron cameras are highly sensitive to noise
  - Characteristic "salt-and-pepper" speckle pattern
  - Signal condition degrades over time
- Image recognition models are sensitive to noise patterns
  - Trained on original (noisy)
- Multiple possible solutions
  - Simple analytic denoising algorithm
    - e.g. "inscribed envelope"
  - ML-based denoising solution
    - Potentially faster execution



Typical raw image from HB-2A



Horizontal pixel trace & inscribed envelope