Laser Focal Position Correction Using FPGA-Based ML Models

Joshua Einstein-Curtis*+, Stephen Coleman*, Nathan Cook*, Jonathan Edelen*,
Samuel Barber**, Curtis Berger**, Jeroen van Tilborg**
+joshec@radiasoft.net

* RadiaSoft LLC, Boulder, CO
**Lawrence Berkeley National Laboratory, Berkeley, CA

ICALEPCS 2023
Cape Town, South Africa
October 10, 2023

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics under Award Number DE-SC 00259037.

Table of Contents

- Abstract
- Project Overview
  - Scientific Basis, Facility Design, Equipment
- Machine Learning
  - Ecosystem, Network architectures, Investigations
- Implementation
  - Data acquisition, processing, and generation
  - Algorithm training, performance, and deployment
  - Hardware
- Further Considerations
- Conclusion

Abstract

High repetition-rate, ultrafast laser systems play a critical role in a host of modern scientific and industrial applications. We present a diagnostic and correction scheme for controlling and determining laser focal position by utilizing fast wavefront sensor measurements from multiple positions to train a focal position predictor. This predictor and additional control algorithms have been integrated into a unified control interface and FPGA-based controller on beamlines at the Bella facility at LBNL. An optics section is adjusted online to provide the desired correction to the focal position on millisecond timescales by determining corrections for an actuator in a telescope section along the beamline. Our initial proof-of-principle demonstrations leveraged pre-compiled data and pre-trained networks operating ex-situ from the laser system. A framework for generating a low-level hardware description of ML-based correction algorithms on FPGA hardware was coupled directly to the beamline using the AMD Xilinx Vitis AI toolchain in conjunction with deployment scripts. Lastly, we consider the use of remote computing resources, such as the Sirepo scientific framework*, to actively update these correction schemes and deploy models to a production environment. Keywords: laser, machine learning, correction, focal position.

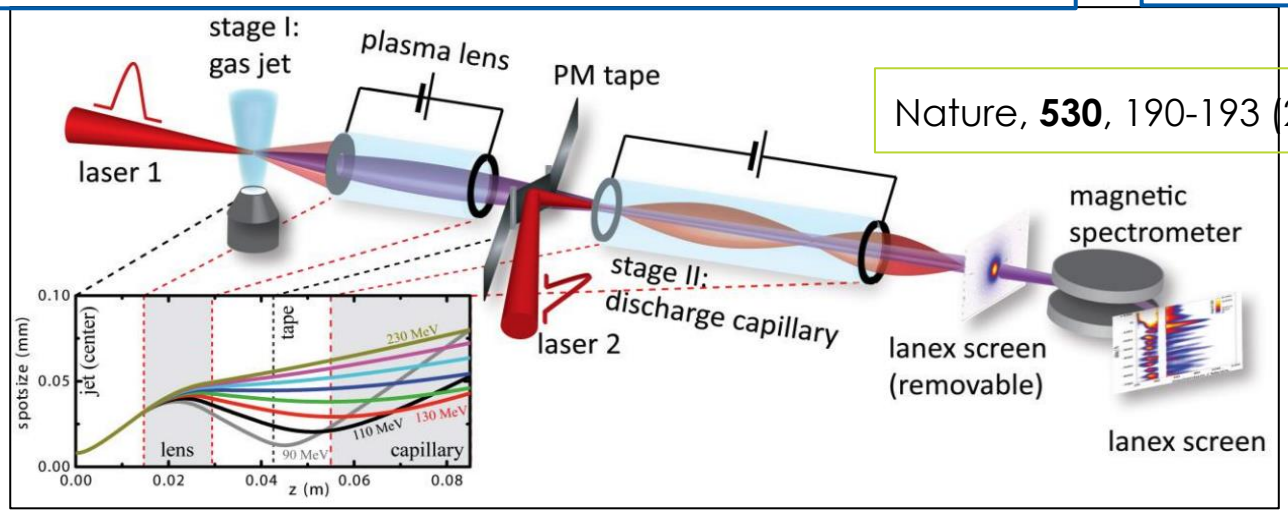
Motivation: High Average Power Lasers for Future Accelerators

High intensity lasers are a critical technology for present-day and future accelerators

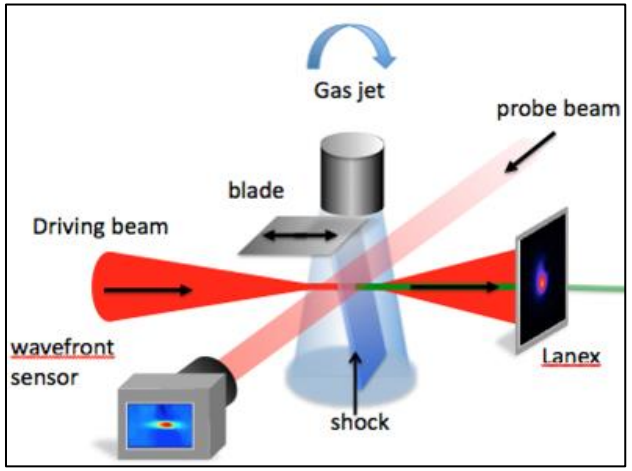
- Electron and proton beam-sources leverage these lasers for ionization, capture, and acceleration
  - Laser plasma accelerators (LPAs) may generate GeV-scale electron beams from cm-scale accelerators
  - Laser-driven ion acceleration schemes rely on careful control of intensity profile and laser/target alignment
- Future applications will require large increases in repetition rate and corresponding stability!

Laser focal position is a critical figure of merit for accelerator applications

- Plasma accelerators are increasingly sensitive to focal position for both injection and acceleration
  - Wavefronts must be matched to plasma for guiding, and focused at the point of injection
- Focal position may vary due to myriad coupled environmental factors along the beamline
  - Vibrations, temperature fluctuations, misalignments
  - Many fluctuations occur at >1 Hz, and require rapid identification and correction



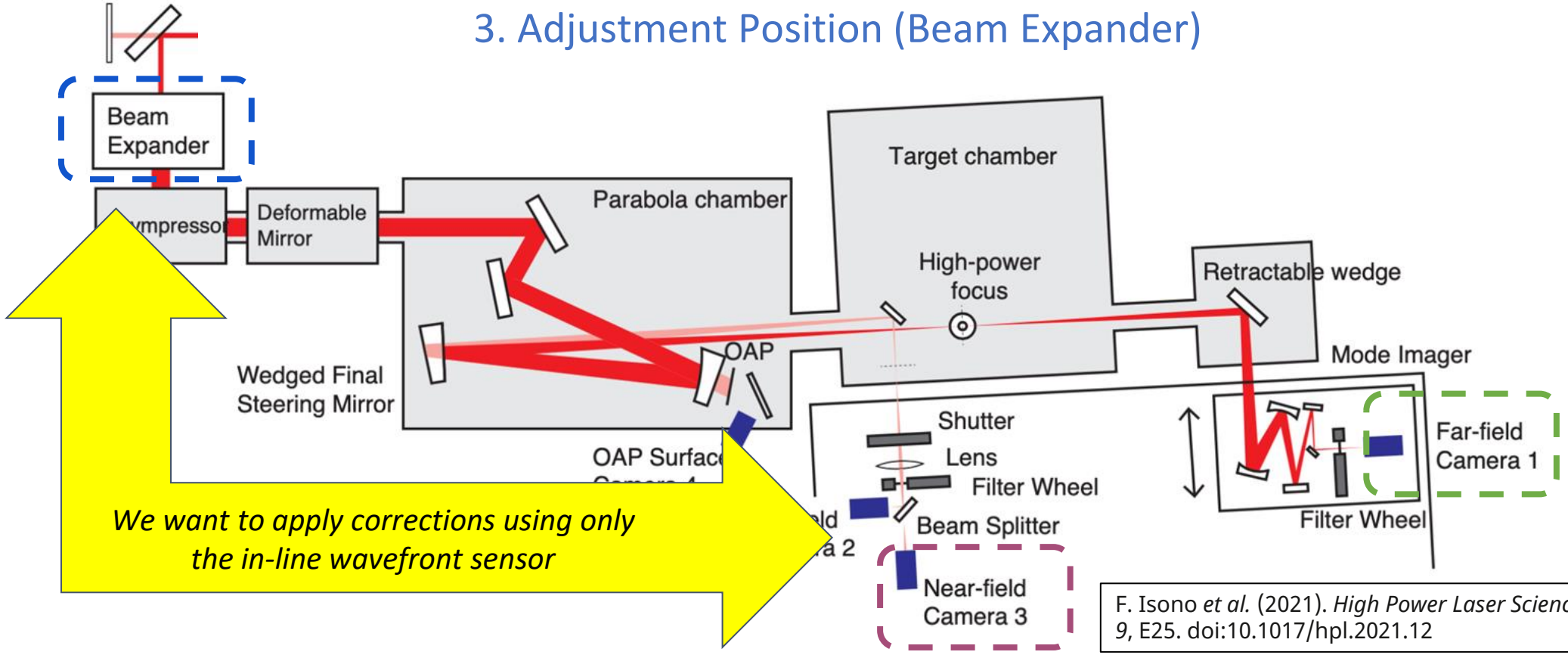
Nature, **530**, 190-193 (2016)



A fast and simple scheme for laser focal position correction

- Objective: Utilize a fast non-perturbative wavefront sensor to predict focal position with high accuracy. A motorized beam expander will permit rapid corrections to the focus.

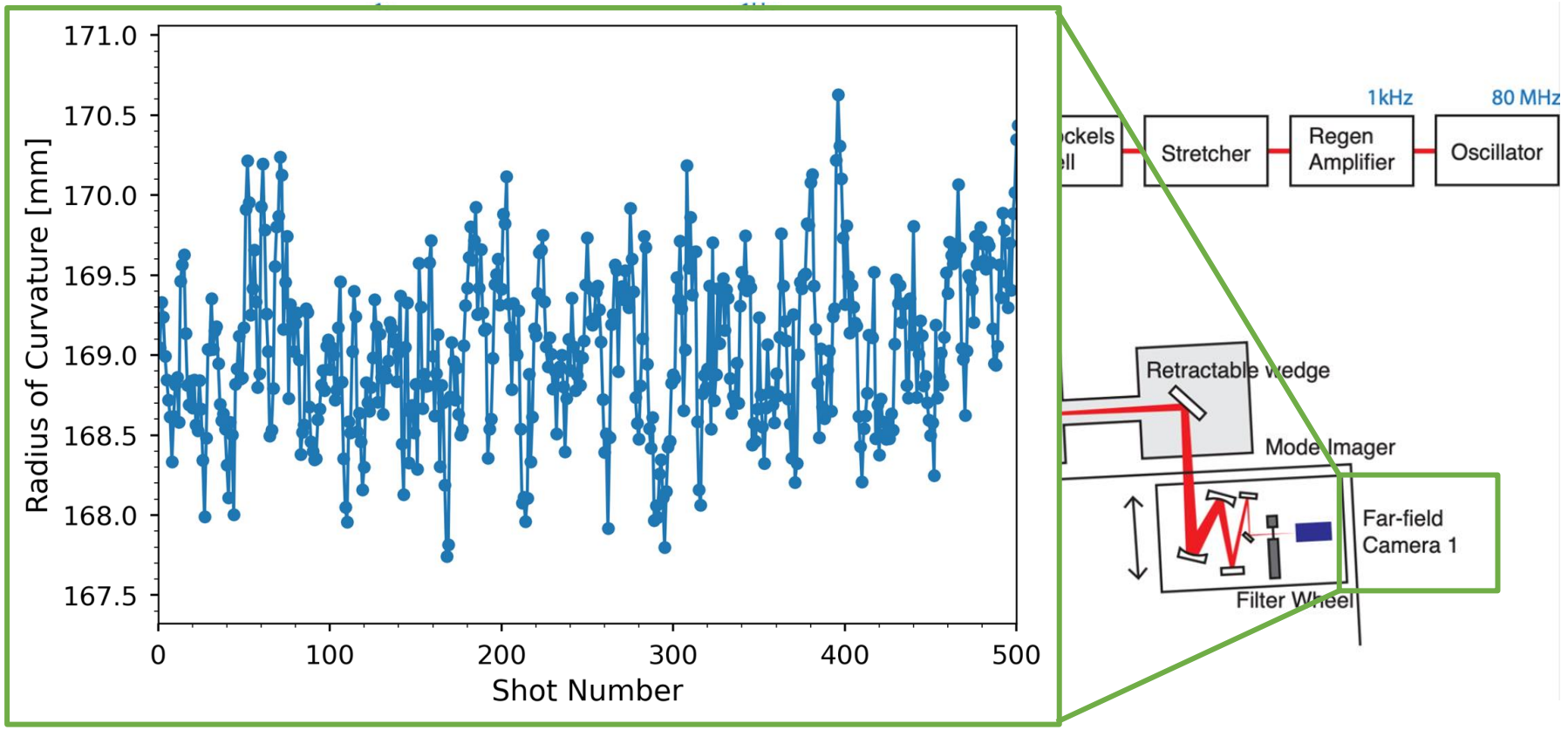
1. Perturbative wavefront sensor (HASO WFS)
2. Non-perturbative wavefront sensor (Thorlabs WFS)
3. Adjustment Position (Beam Expander)



F. Isono *et al.* (2021). *High Power Laser Science and Engineering, 9*, E25. doi:10.1017/hpl.2021.12

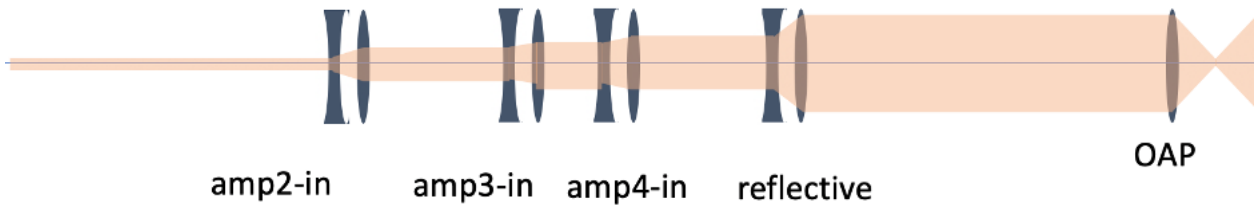
Focal Position Variation is a concern for optimizing interactions

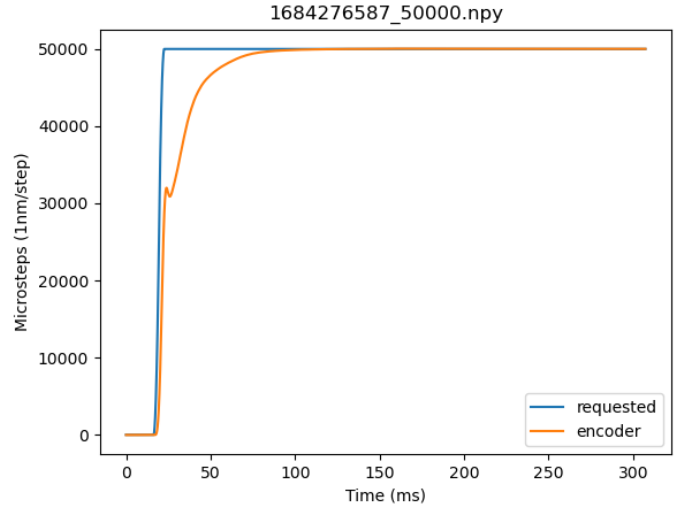
- Systems exhibit significant shot-to-shot fluctuations in focal position, as evidenced by high-quality laser wavefront measurements taken at the beamline
- In any real experiment, there is a tension between what one has and what one wants
  - 1 Hz high-power pulse <-> 1 kHz seed pulse <-> <10 Hz corrector response
  - Camera Resolution <-> Driver-calculated Figures <-> Model Resolution



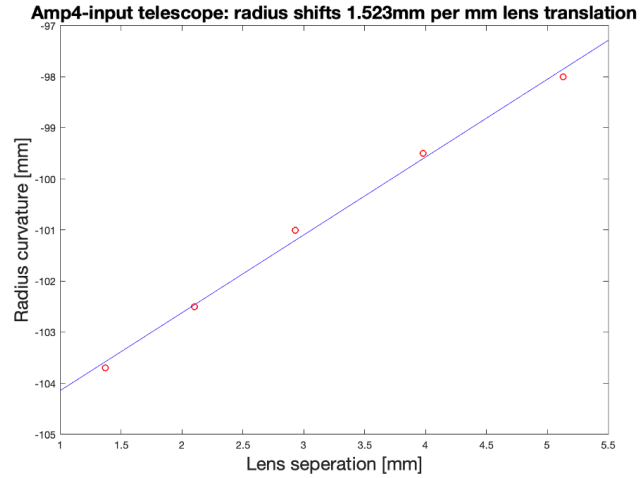
Beam Telescope and Motion Stage

- A transmissive, telescopic beam expander enables flexible focal position adjustment
  - Modifications can be made throughout amplification chain prior to compression and final focus.
- Linear stage is under consideration for fast lens movements
  - We chose a Zaber X-LDA025A-AE53D12 for prototype testing due to its availability and features
  - This brings up operational concerns though – no “park” and need to transparently bridge stage for remote control
  - Built-in PID controller and serial communication baud rate limits control bandwidth



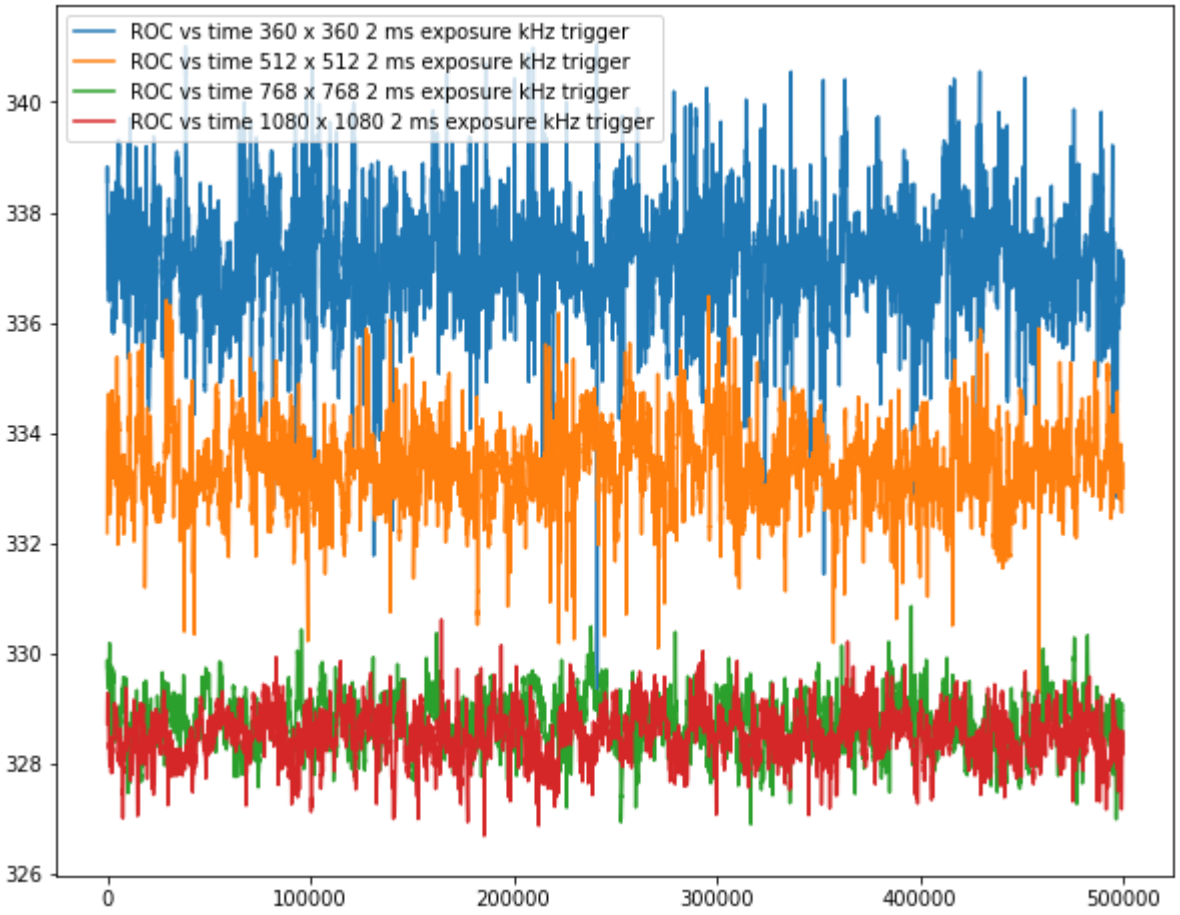


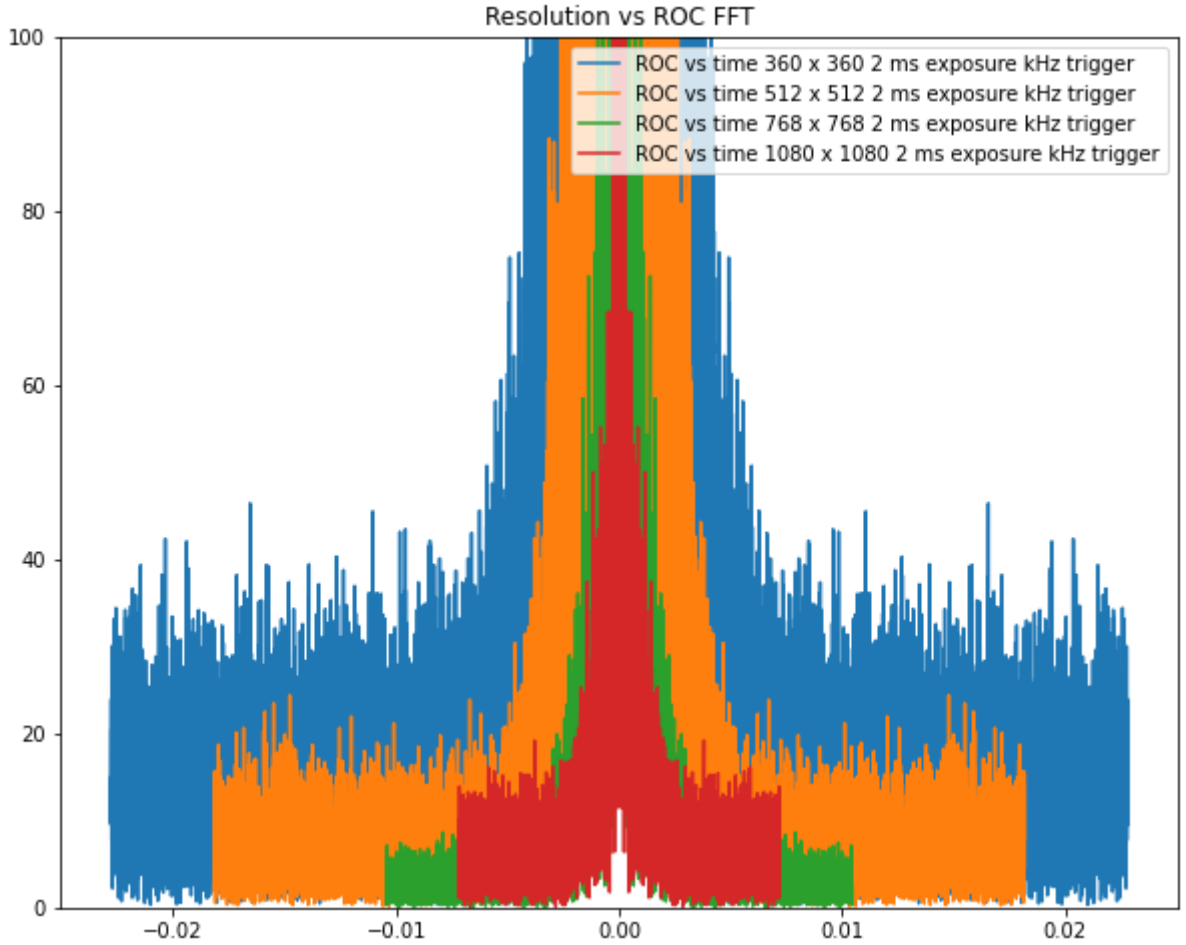
Linear stage response



Focal position sensitivity to lens movement

Systematic Measurement Effects



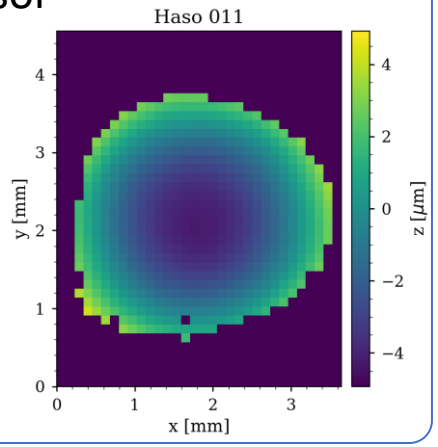


Thorlabs WFS20 image resolution setting vs calculated Radius of Curvature
Left: Calculated ROC in time domain, Right: FFT of calculated ROC

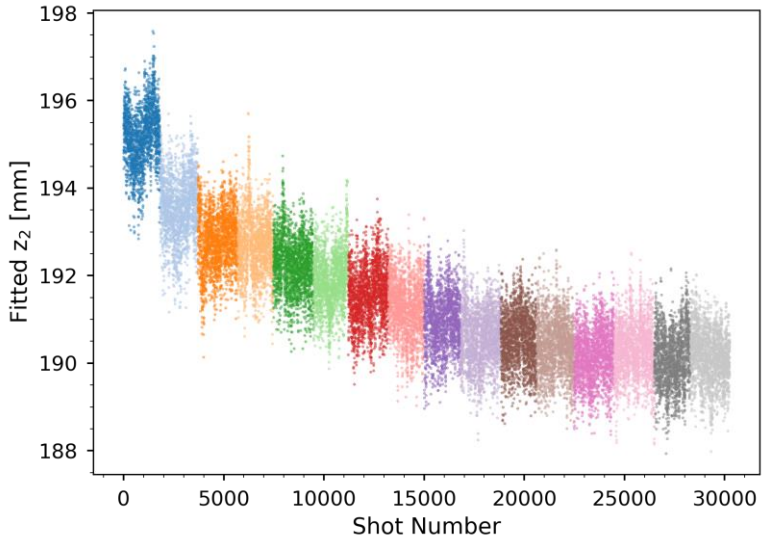
Speed and fidelity tradeoffs motivate processing pipeline

HASO4 Wavefront Sensor

- Positioned at end of beamline
- 110µm pixel pitch
- 100Hz Read Frequency
- Used as ground truth for sample data
- Provides tools for generating fits to wavefront data

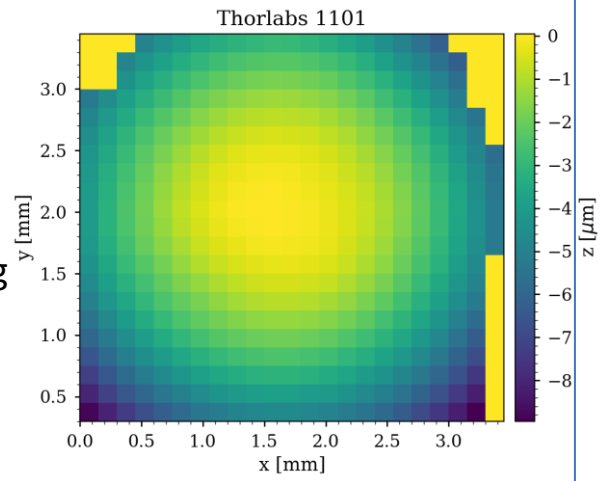


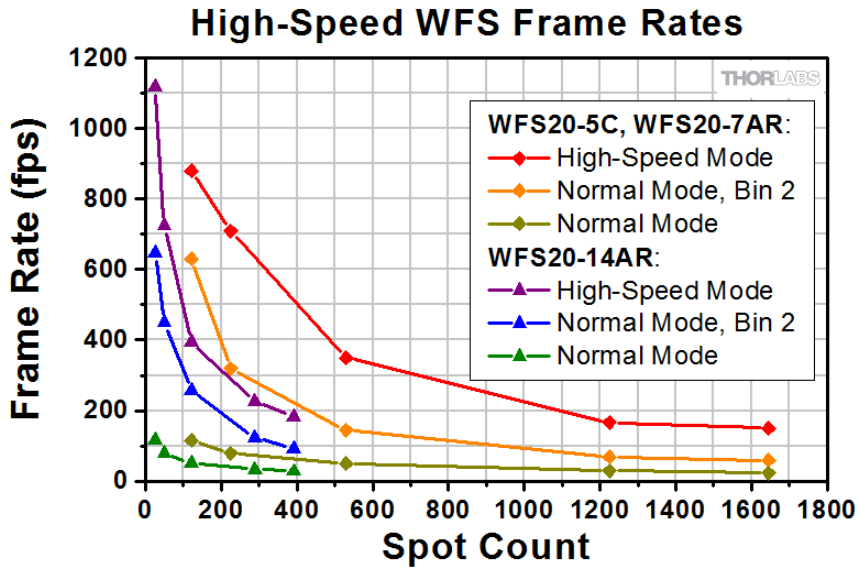
Dataset includes 30k shots across separate runs



Thorlabs WFS20-7AR Wavefront Sensor

- Non-perturbative, parasitic measurement upstream of laser focus
- 150µm lenslet pitch
- Provides tools for generating fits to wavefront data
- Up to 1 kHz Read Frequency

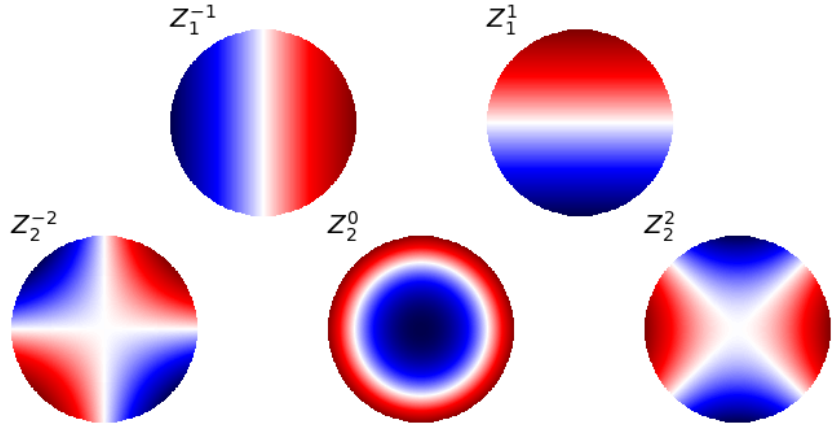




Two classes of challenges for implementing this scheme

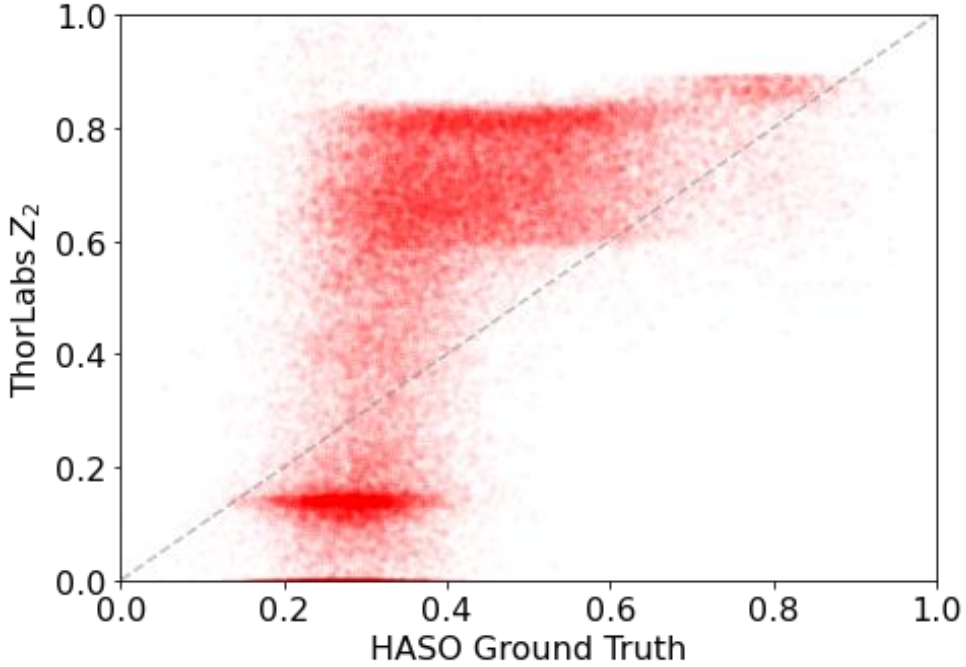
- **Processing:** Develop an algorithm that can identify and correct for variations in laser focal position using fast, online measurements
  - Generate a representative dataset of wavefront images
  - Train an ML-based model to correlate non-perturbative (e.g. online) measurements with perturbative (e.g. offline) ones and inform correction
  - Consider controls schemes for predicting correction (e.g. PID, feed forward, model predictive)
  - What figure of merit? e.g., Zernike coefficients, radius of curvature

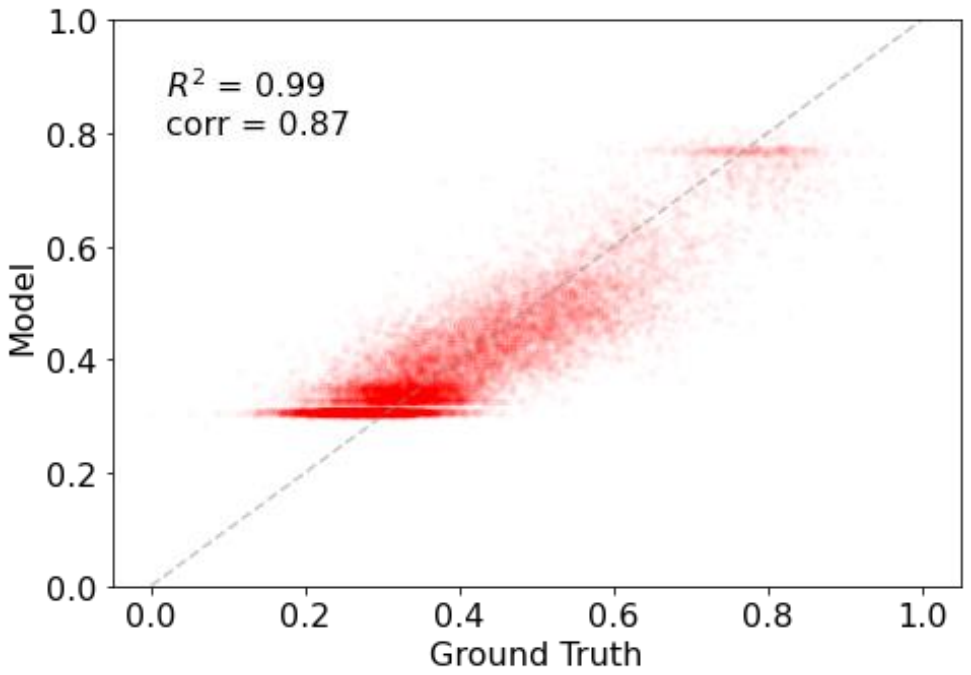
- **Deployment:** Demonstrate the feasibility of deploying such algorithms on FPGA/accelerator systems in conjunction with corrective optics
  - Identify and address relevant data pipeline considerations
  - Determine toolchain for flexible deployment of corrective model
  - Consider optical configuration and actuator needs



Neural Networks and Data Correlation

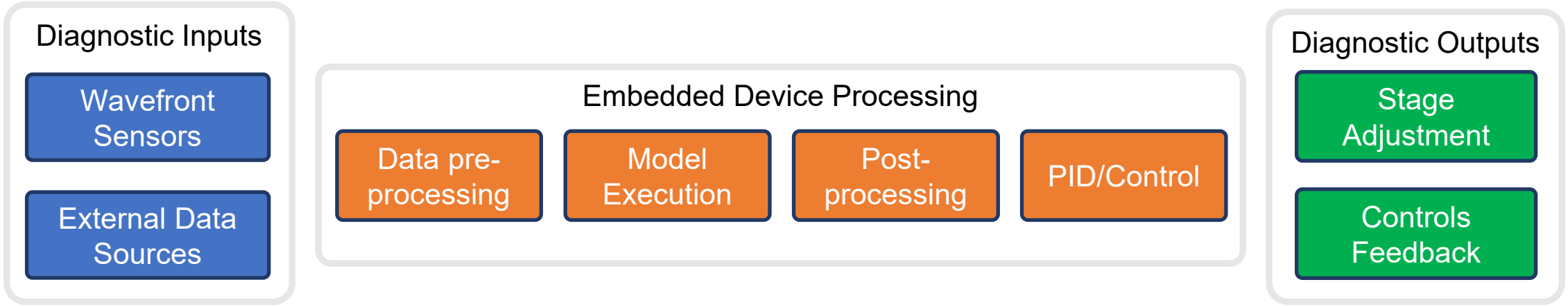
- Initial strategy leverages investigated both CNN and fully-connected feed-forward networks
  - Networks feature 2-4 hidden layers of varying size, ReLU activation, robust scaler on inputs/outputs
  - Since focal point errors may not be correlated between cameras, FFNN may be better at learning relationships
  - FFNN should be easier to augment with additional input, with meaning, without highly custom architectures
- Balance between data cleaning, augmentation, and network size
  - Raw correlation between radius of curvature from Zernike fits alone is poor (0.45 - *left*)
  - Using *only* pixel data (144-pixel vector) requires largest network size to enhance correlation (0.82)
  - Augmenting pixel data with Zernike fits achieves higher correlation (0.87) with fewer hidden layers (*right*)
    - Thorlabs camera firmware provides native Zernicke fits to 5th order (16 terms)





Data pipeline and controls integration are required

- System-level control requires coordination between many components
  - Wavefront sensors, data analysis and model prediction, telescopic lens adjustments, and broader controls system
  - System should have flexibility to incorporate external data sources independent of the wavefront sensor
    - Environmental inputs may include auxiliary measurements (temperature, humidity, etc.)
    - Controls system feedback may be required to account for global operational needs (machine protection, interlocks, etc.)

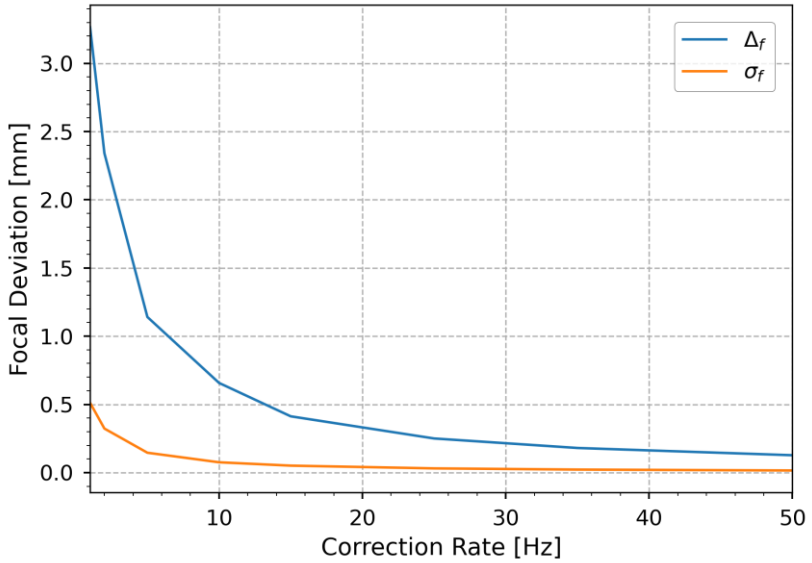


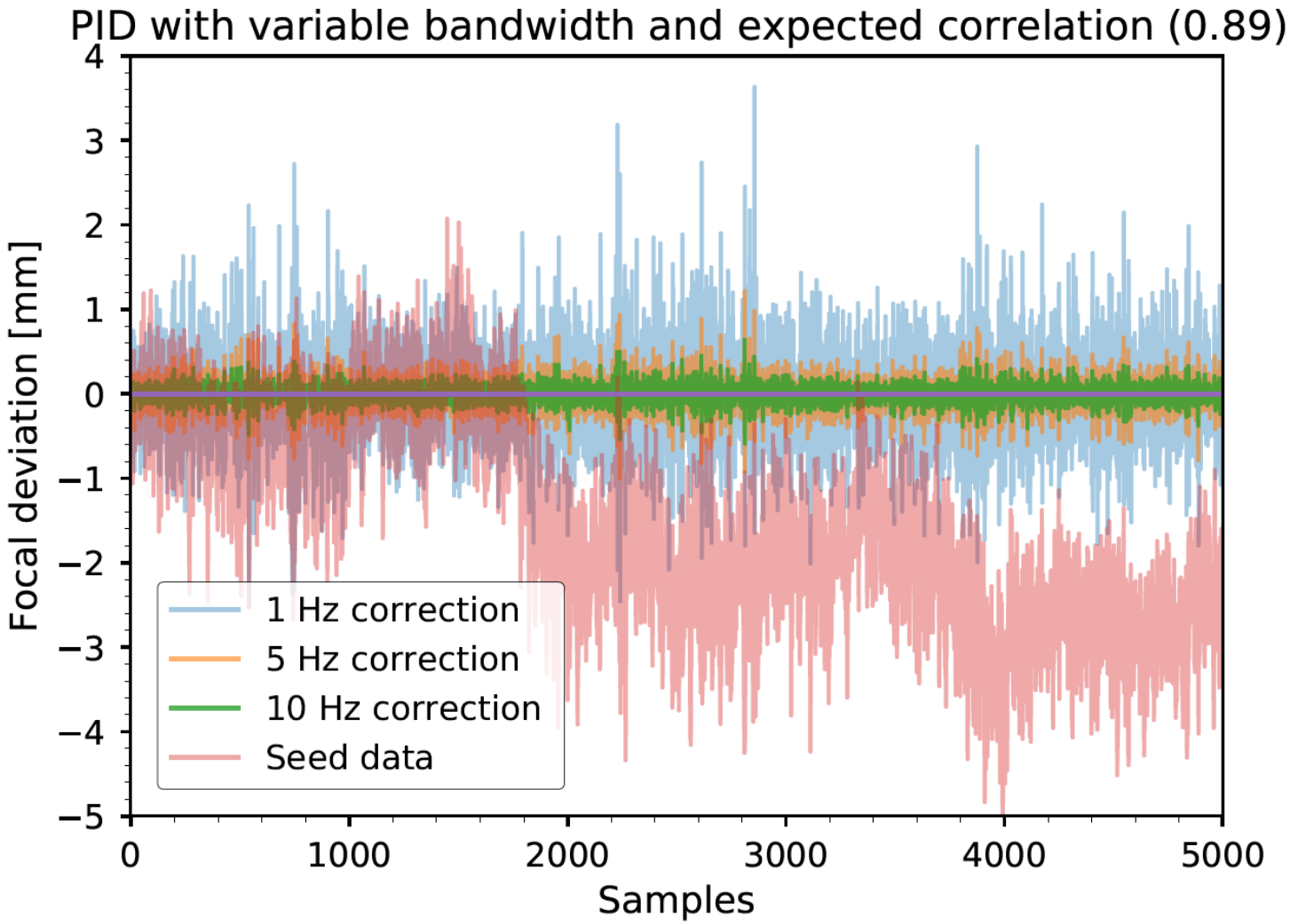
- Processing pipeline requires data processing and communication across systems
  - Multiple inputs and outputs requiring multiple levels of processing

- Our strategy is to employ a fast, embedded device to enact processing and control algorithms

Estimating Online Performance

- Simulated PID controller provides an initial correction estimate
  - Existing data sampled and interpolated
- Correction scales with update rate
  - 1 Hz sampling is sufficient to remove slow excursions
  - 25 Hz correction reduces maximum deviation to ~0.25 mm and one sigma deviation to 0.031 mm





Data Transport Example

```
(image) xilinx-zcu104-2021_1:~/khzwave_python/src/khzwave$ python
receiver.py -d roentgen
Connecting to host: roentgen
INFO:main:[[129  28 169  69  25 255 226 251  27  56   8]
 [ 87 114 177 211  22  65  43  75 164  14 142]
 [236 139 230  23   9 180   9  76  31 139 199]
 [252 229   7 215  59   4  81  71  43  32 224]
 [241   1 176 228 123  51  29  34  47 174  57]
 [121 138 199 103 110 116 162 175 211 154  18]
 [215  48  80  30  52 191  50 113  35 229  82]
 [ 46 210 210  83  51 168 252  13 188  31 109]
 [ 98  59 215 159 194 167  44  32 181 249 237]
 [ 17 101 233  60 130  90 219 155 238 120 174]
 [ 80  85  30 138 176 121 237  69 154 198 125]]
```

```
(dev) C:\Users\jeins\source\repos\khzwave\src\khzwave>python server.py
INFO:producer:Producing data...
…
INFO:main:+1 subscriber (1/1)
INFO:main:Awaiting data...
INFO:producer:Producing data...
INFO:main:Send took 0 ns
INFO:main:Awaiting data...
INFO:producer:Producing data...
INFO:main:Send took 0 ns
INFO:main:Awaiting data...
INFO:producer:Producing data...
INFO:main:Send took 0 ns
INFO:main:Awaiting data...
INFO:producer:Producing data...
INFO:main:Send took 0 ns
INFO:main:Awaiting data...
INFO:producer:Producing data...
INFO:main:Send took 0 ns
INFO:main:Awaiting data...
```

NOTE: This does not really take 0 ns, it is just that the system time reporting granularity is too large to measure. For example some systems have 100 us or 1 ms operating system ticks

Deploying corrections via FPGA systems

- FPGAs enable lightweight performant embedded systems for real time controls
  - Reprogrammable logic permits streamlined updates and operational feedback
- Performance necessitates use hardware description language (HDL)
  - High Level Synthesis (HLS) provides a means for consistently transferring operations designed using high level languages (C++, Python) into low-level representations required by hardware
- Deployment Process involves multiple steps:
  1. *Algorithm Design* – Parametrize and train neural network
  2. *Optimize and Quantize* – Reduce size of network features (pruning) or precision of operations (quantization) to enhance performance
  3. *Compile* – Build HDL representation of network for device execution
  4. *Deploy* – Configure device runtime to load and execute network in response to inputs and job requests
- Choice of device architecture, manufacturer, and firmware constrains development environment
  - Manufacturers impose toolchain requirements which limit high level framework choice
    - E.g. Choice of OS (Windows, Linux), API (Python, C), and Architecture (ARM, X86)
  - Ongoing efforts to integrate disparate platforms and architectures – see ONNX (https://onnxruntime.ai/)

Xilinx Vitis AI DPU Workflow

- Develop and save model
- Build python script to run in Vitis AI docker container to quantize model and save the model
  - This script should also check performance for a production deployment as performance IS lost in the quantization process
  - Quantization is necessary to run a model using integer types instead of float types, due to accelerator data format requirements
- Compile the model in to the Xilinx DPU .xmodel format
  - The Xilinx DPU pipeline involves several steps, including the use of the Vitis AI docker container (docker.io/xilinx/vitis-ai or built manually)
  - Highly version dependent as to if the saved model can run through the pipeline properly
  - Model format fights: onnx, tf, tf2, torch, tvm
- A series of scripts had to be developed or modified to handle the build environments and tooling that we are using (TANSTAAFL)
  - Deploy the xmodel, run script, and dataset to the device
  - Run the performance test

Stage 1

ffnn.py → docker_run.sh → runme.sh → runme_tf2.py → compile_for_zcu104.sh

Stage 2

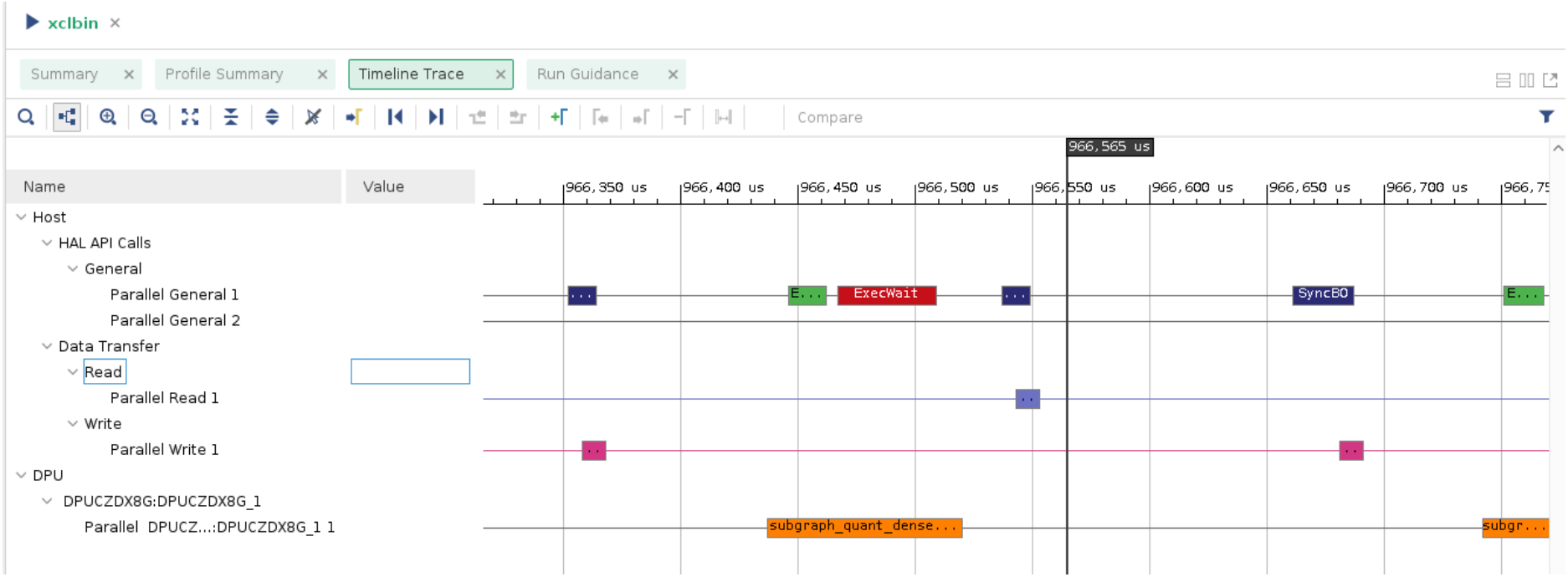
deploy.sh → radiasoft.py

Xilinx Vitis AI DPU Trace

Performance estimates are encouraging for high repetition rates

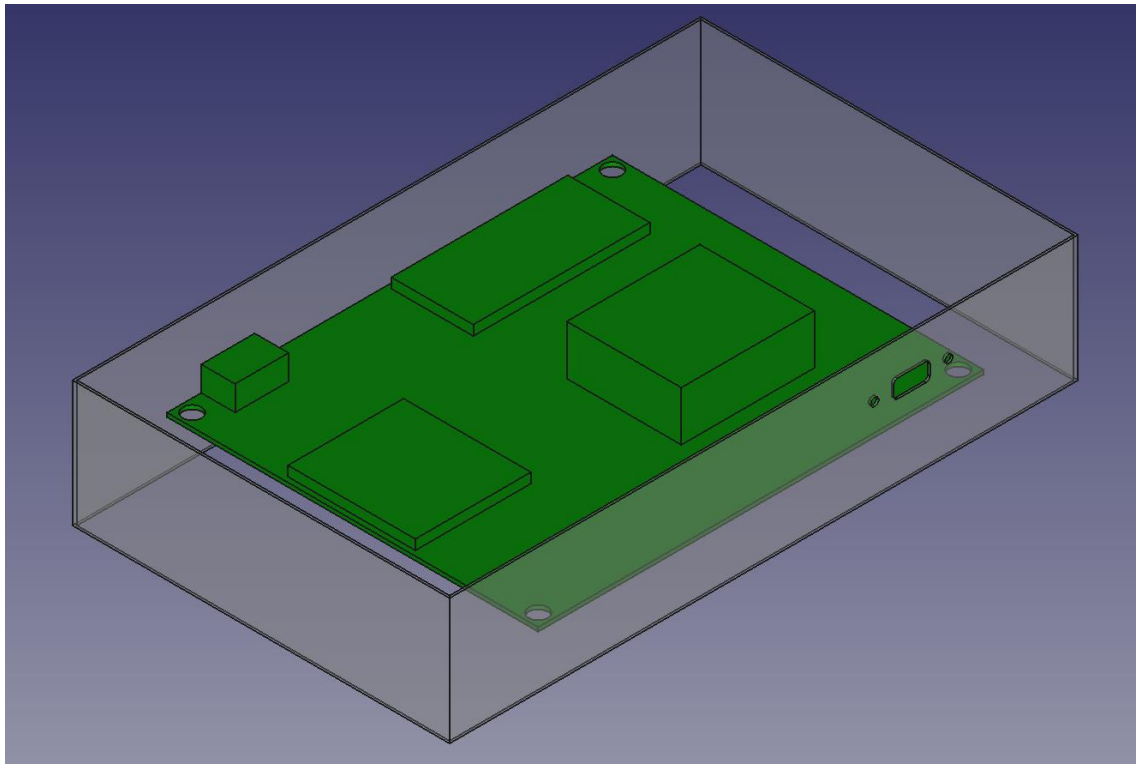
- Prototype implementation shows consistent performance at 5k SPS
  - Room to expand network size or couple multiple interactions while still achieving kHz operation

```
(image) xilinx-zcu104-2021_1:~# vaitrace -p python /tmp/radiasoft.py -d /tmp/data.hdf5 -m /tmp/radiasoft
      .xmodel
input_fixpos=7 input_scale=128
SPS=5047.43, total samples = 360.00 , time=0.071323 seconds
Closing remaining open files:/tmp/data.hdf5...done
```



Physical Deployment

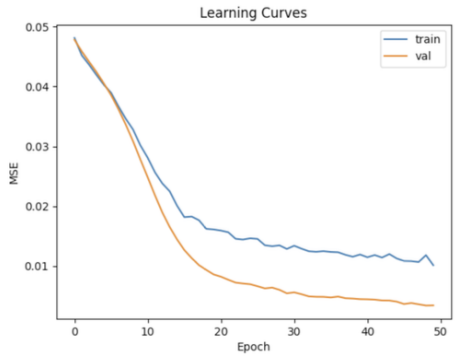
- Several reasons exist to develop a solution that is packaged and looks good
  - Shifting timelines and schedule make it a challenge to source components when collaborating
- Schedule and resources
  - Custom machining vs kit-bashing something that fits
  - Component selection is not straightforward when working with evaluation kits



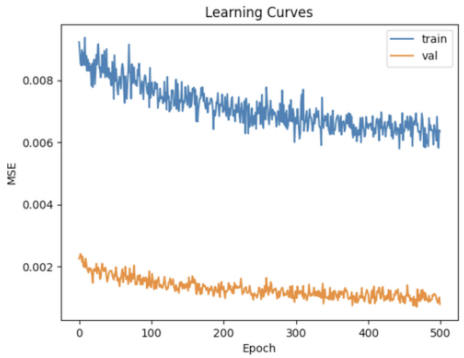
Sirepo Activait

Neural Network Layers

Layer	Dimensionality	Kernel Size	Strides	Padding	Activation
Conv2D Layer	32	3	1	Pad with zeros (sar	
Layer	**Momentum**				
Batch Normalization	0.99				
Conv2D Layer	32	3	1	Pad with zeros (sar	
Batch Normalization	0.99				
Layer	**Pool Size**	**Strides**	**Padding**		
Max Pooling	2	2	Pad with zeros (same)		
Layer	**Rate**				
Dropout	0.3				
Conv2D Layer	64	3	1	Pad with zeros (sar	
Batch Normalization	0.99				
Conv2D Layer	64	3	1	Pad with zeros (same)	Rectified Linear Unit (relu)
Batch Normalization	0.99				
Layer	**Pool Size**	**Strides**	**Padding**		
Max Pooling	2	2	No padding (valid)		
Layer	**Rate**				
Dropout	0.5				



(a) The first 50 epochs.



(b) Additional 500 epochs.

Figure 25: Training curves for the prototype convolutional neural network trained to predict real-valued labels using single-channel 2D images as input (see text).

We train neural networks on simulation or measurement data to create a surrogate model of beamlines.

We have automated SRW simulations to create input to be read into Activait for training data, for example.

https://www.sirepo.com/activait

Conclusions

- We are developing a scheme for correcting laser focal position at high repetition rate
  - Our approach integrates fast, non-perturbative measurements with machine learning models to predict the focal position and determine a correction.
  - We aim to develop an in-hardware solution for flexibly deploying and updating the correction model
- We have trained a model to quickly correlate upstream and downstream diagnostics
  - Feed-forward neural network, augmented by fits, can significantly increase correlation between sensors
    - Identified trade-offs between sample data size, pre-processing, input space, network size, and correlation quality
    - We are continuing to evaluate different models to address different use-cases and deployment strategies
  - PID-based controller, alone, promises significant improvements at > 1 Hz repetition rates
- We have identified a deployment strategy leveraging a Xilinx DPU co-accelerator
  - Programmable device with large FPGA fabric for expansion and/or parallelization
  - Deployment pipeline leverages PyTorch + digitization and export via Xilinx development platform (Vitis AI)
    - Future efforts will explore the use of ONNX runtime for packaging the entire deployment process
  - Initial testing illustrates execution speeds of 5K SPS
- Experimental demonstration is scheduled shortly
  - We have tested each component of the pipeline independently and on the testbench

Thank you for your attention!

Disclaimer

Sirepo development has been supported by the U.S. Department of Energy Office of Science under multiple awards:

- by the Office of High Energy Physics under Award Nos. DE-SC0011340, DE-SC0015897 and DE-SC0018719;
- by the Office of Basic Energy Sciences under Award Nos. DE-SC0011237, DE-SC0015209, DE-SC0018556, DE-SC0020593 and DE-SC0018571;
- by the office of Nuclear Physics under Award Nos. DE-SC0015212 and DE-SC0017181;
- by the Office of Advanced Scientific Computing Research under Award Nos. DE-SC0017162, DE-SC0021553, DE-SC0019682, DE-SC0022386 and DE-SC0017057.

References

- Mlexchange (https://arxiv.org/pdf/2208.09751)
- https://indico.bnl.gov/event/16158/contributions/69576/attachments/44293/74735/20221102_Edelen_ICFA.pdf
- NERSC Nvidia Nsight training https://www.nersc.gov/users/training/past-training-events/2022/profiling-deep-learning-applications-with-nvidia-nsight/