Online Correction of Laser Focal Position Via Deployable Machine Learning Models

Nathan Cook*+, Stephen Coleman*, Jonathan Edelen*, Joshua Einstein-Curtis*,
Samuel Barber**, Curtis Berger**, Jeroen van Tilborg**

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

+ncook@radiasoft.net

November 7, 2022

2022 Advanced Accelerator Concepts Workshop

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(s) DE-SC0021680

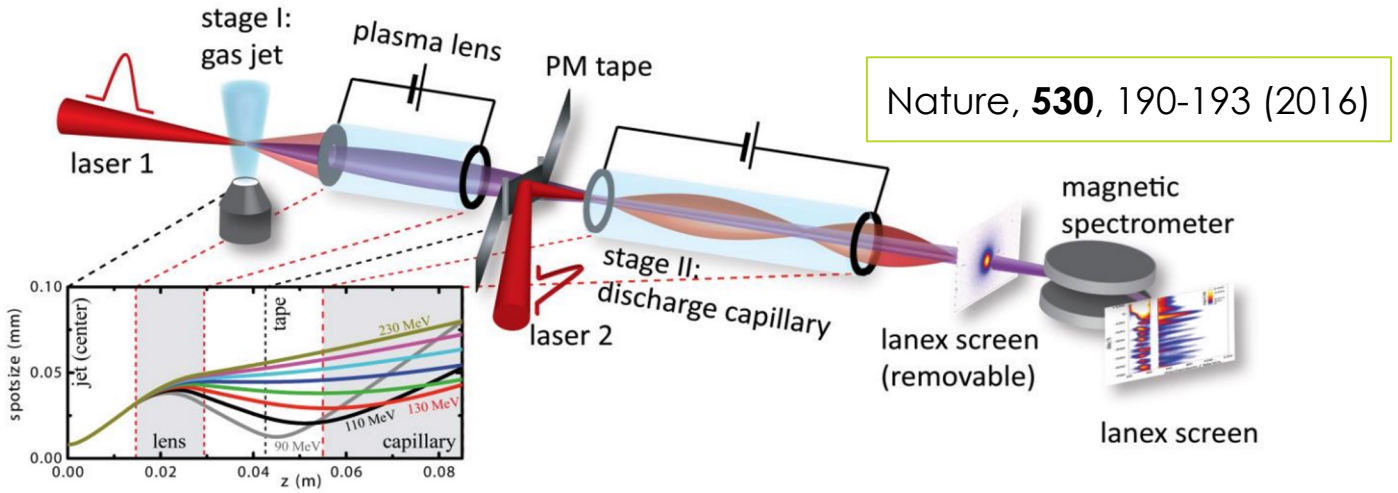
Motivation: High Average Power Lasers for Future Accelerators

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

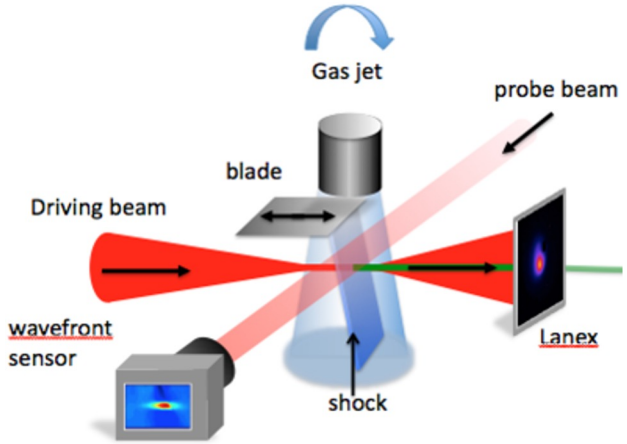
- Electron and proton beam-sources leverage these lasers for ionization, capture, and acceleration
  - Laser plasma accelerators (LPAs) may generate GeV-scale electron beams from cm-scale accelerators
  - Laser-driven ion acceleration schemes rely on careful control of intensity profile and laser/target 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

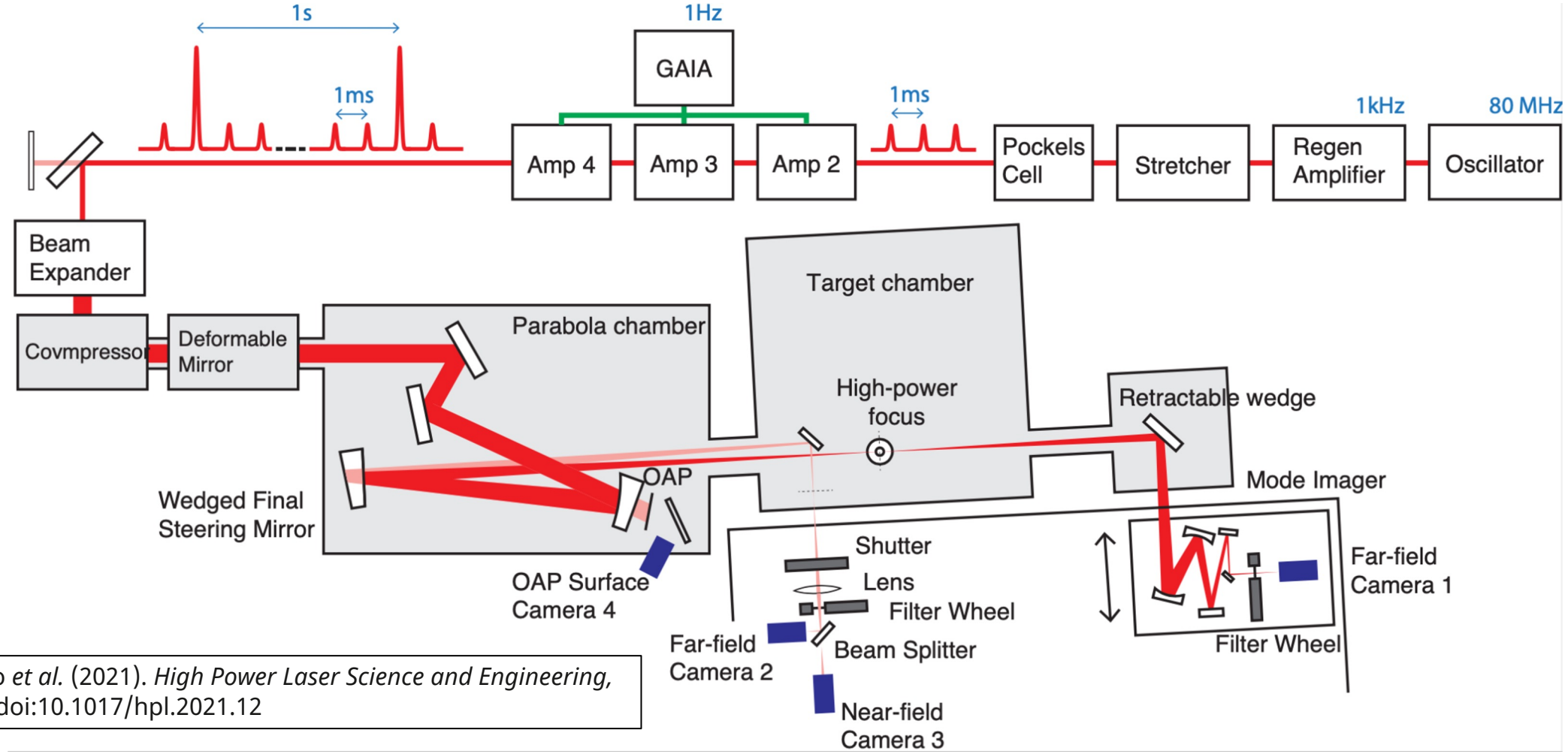


Nature, **530**, 190-193 (2016)



Advances in laser & diagnostic technology promise enhanced stability

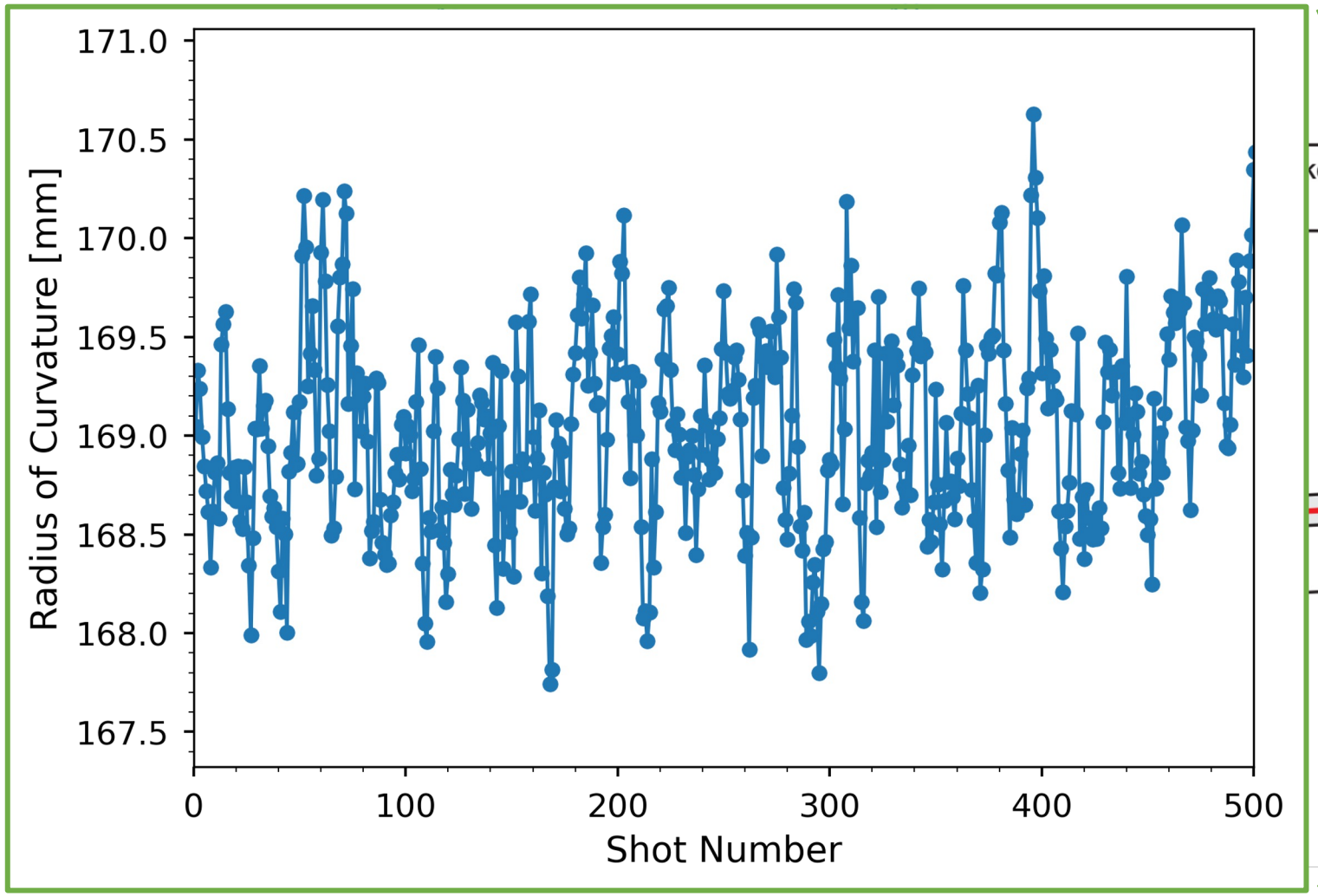
- We consider the HTU laser system at the BELLA Center
  - 100 TW fully amplified pulses at 1 Hz, with non-amplified pulses at 1 kHz, and a set of non-perturbative diagnostics

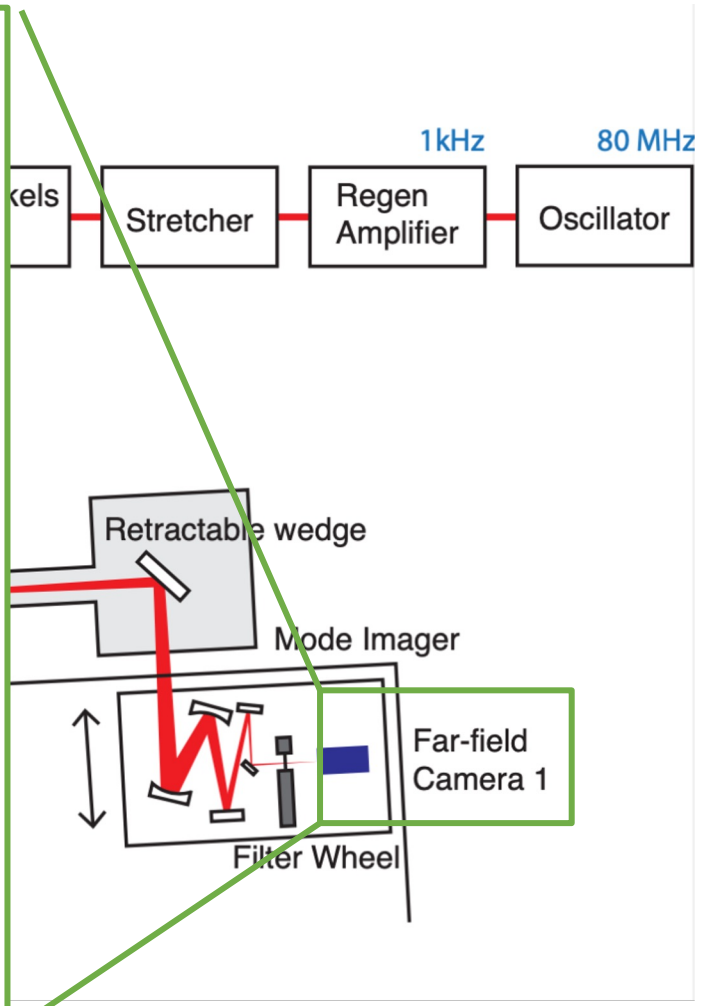


F. Isono *et al.* (2021). *High Power Laser Science and Engineering, 9*, E25. doi:10.1017/hpl.2021.12

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

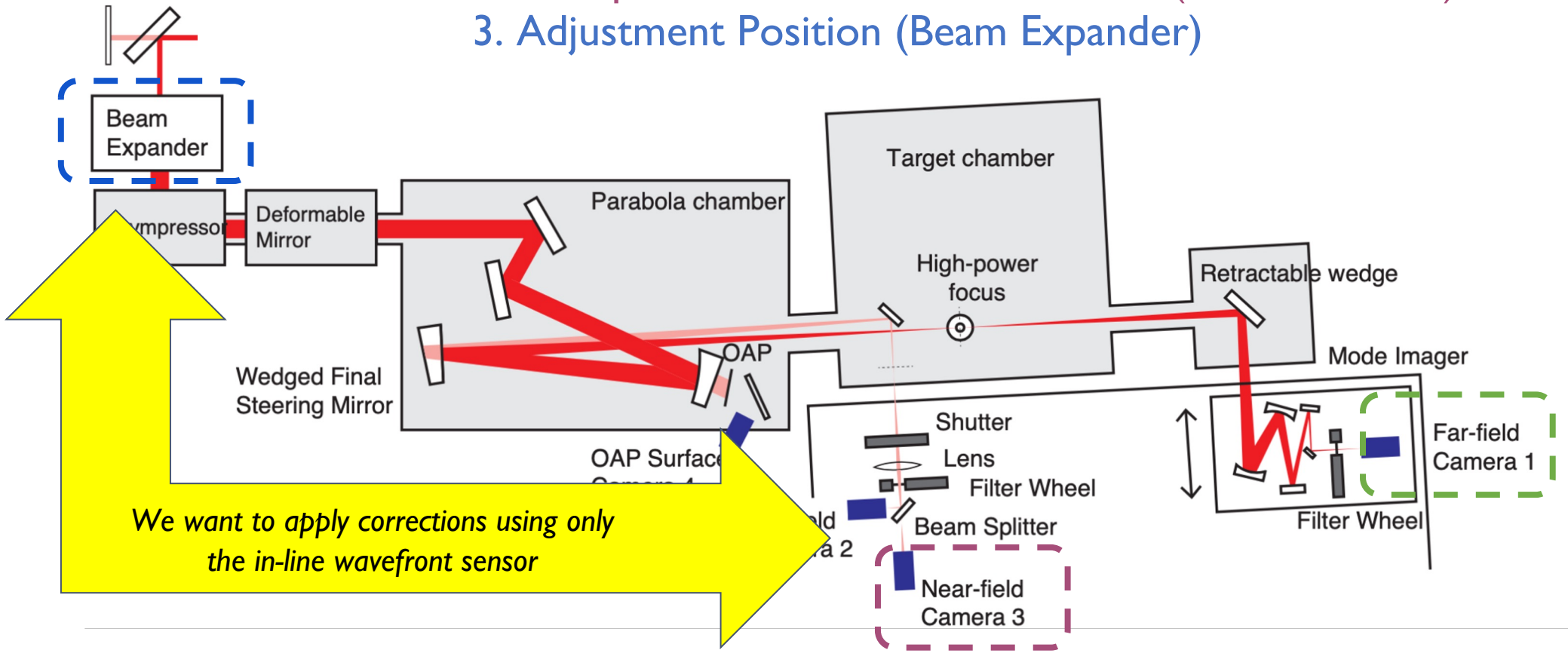




A fast and simple scheme for laser focal position correction

- 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)



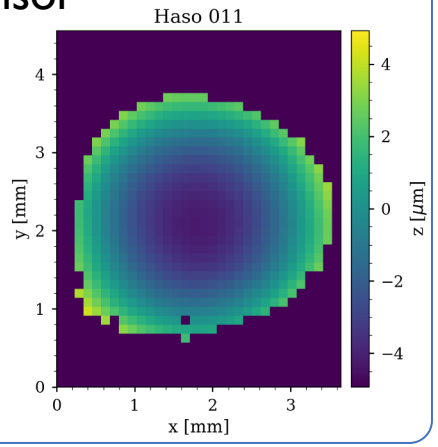
Two classes of challenges for implementing this scheme

- **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)
  - Evaluate methods for rapidly adjusting focal position
- **Deployment:** Demonstrate the feasibility of deploying such algorithms on FPGA/accelerator systems in conjunction with corrective optics
  - Explore different hardware configurations
  - Identify and address relevant data pipeline considerations
  - Determine toolchain for flexible deployment of corrective model
  - Consider optical configuration and actuator needs
- We first discuss processing followed by deployment

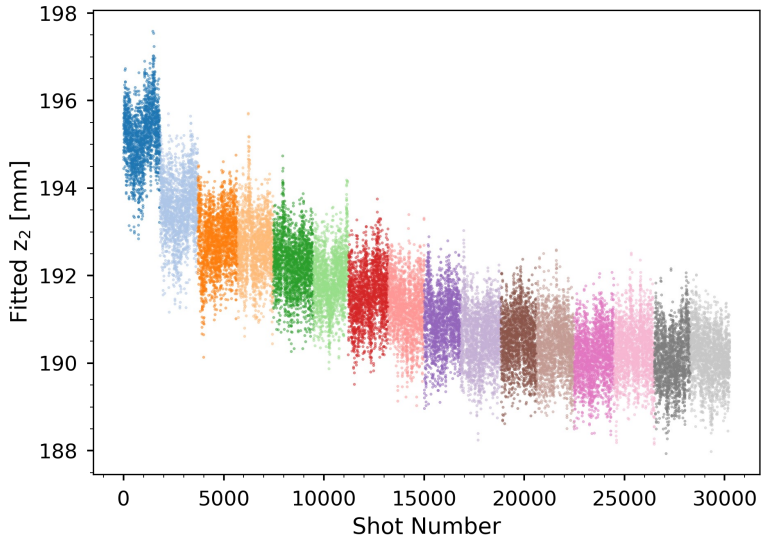
Camera and dataset considerations for fast capture

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

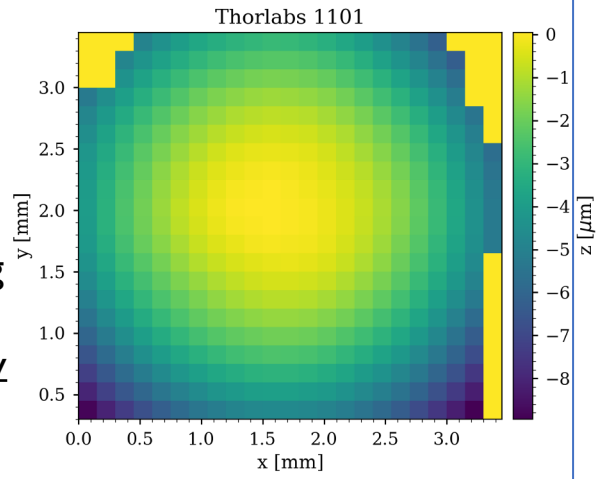


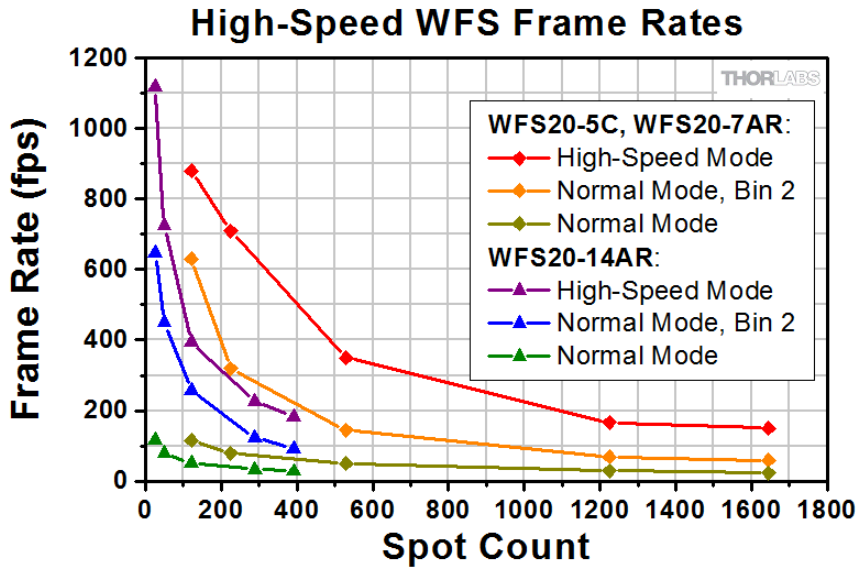
Dataset includes 30k shots across separate runs



Thorlabs WFS20-7AR Wavefront Sensor

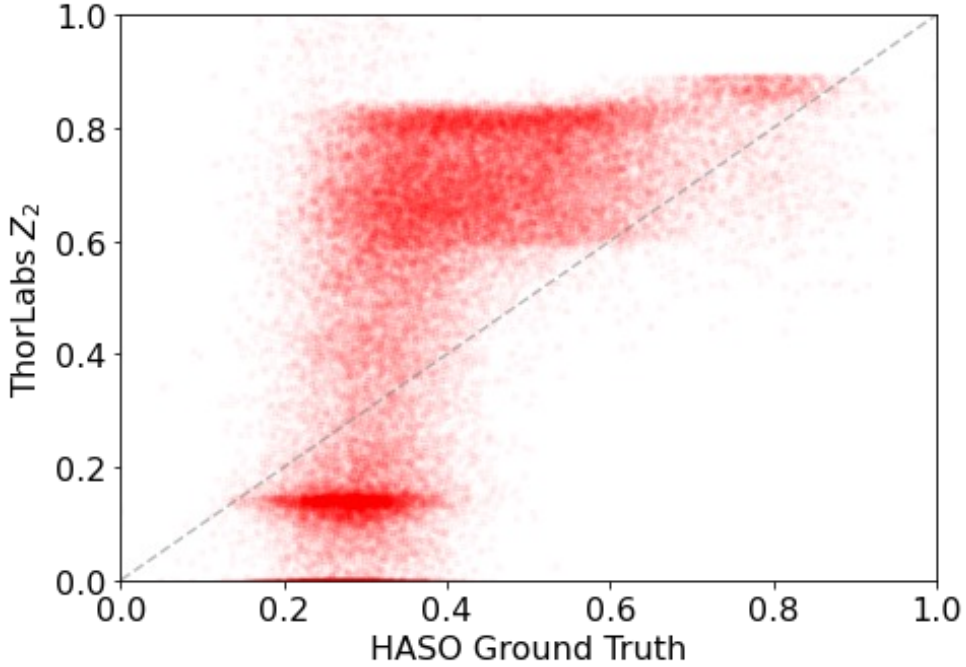
- Non-perturbative, parasitic measurement upstream of laser focus
- 150µm lenslet pitch
- Provides tools for generating fits to wavefront data
- Up to 1 kHz Read Frequency

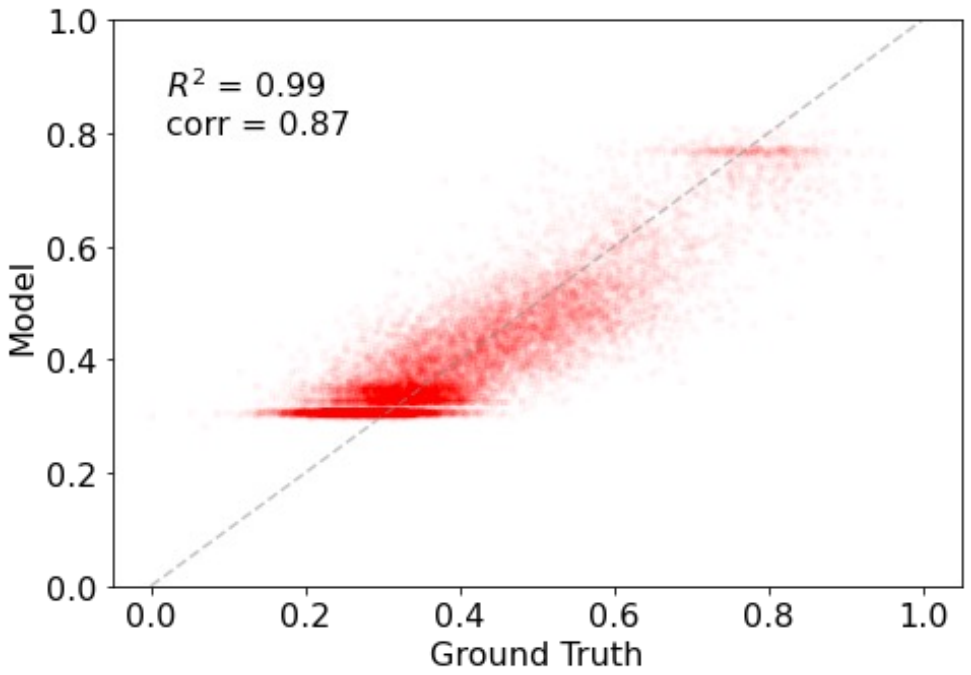




Neural networks show promise for correlating diagnostic outputs

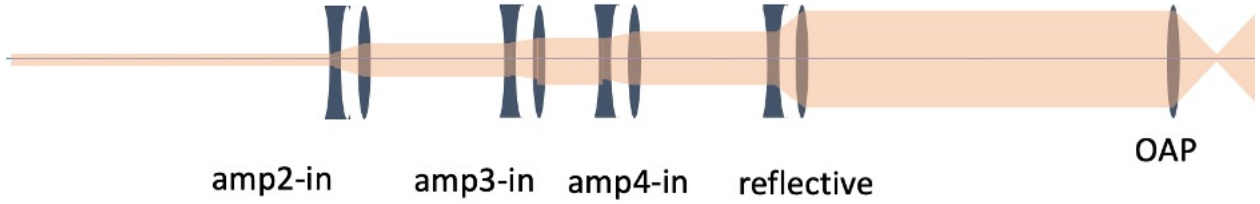
- Initial strategy leverages vanilla feed-forward neural networks for mapping datasets
  - Networks feature 2-4 hidden layers of varying size, ReLU activation, robust scaler on inputs/outputs
- Balance between data cleaning, augmentation, and network size
  - Raw correlation between radius of curvature 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 fits achieves higher correlation (0.87) with fewer hidden layers (*right*)
    - Thorlabs camera firmware provides native Zernicke fits to 5th order (16 terms)



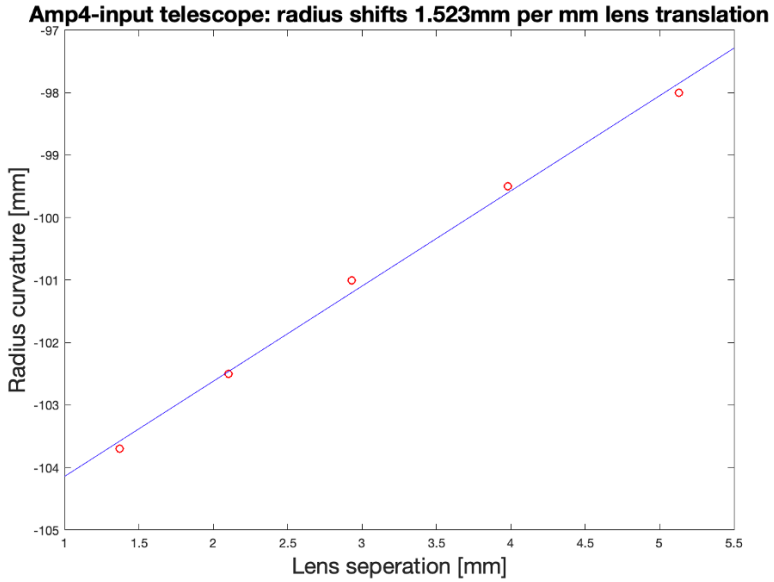


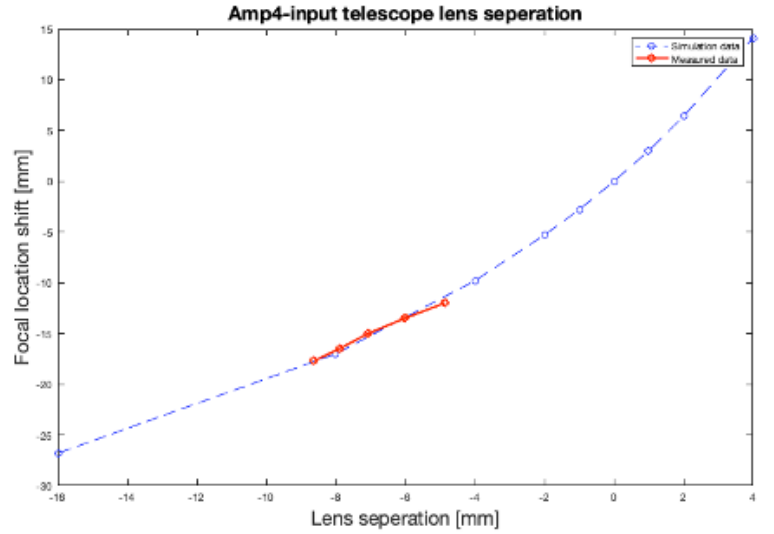
Telescopic Lens Translation Provides Focal Adjustment

- A transmissive, telescopic beam expander enables flexible focal position adjustment
  - Modifications can be made throughout amplification chain prior to compression and final focus.
  - Evaluations were made to select between amp3-in, amp4-in transmissive lenses, and a reflective mirror following amp4-out
    - Amp4-in lens was chosen for large focal shift per mm change in lens separation
    - Additional re-tuning required if reference lens separation is changed



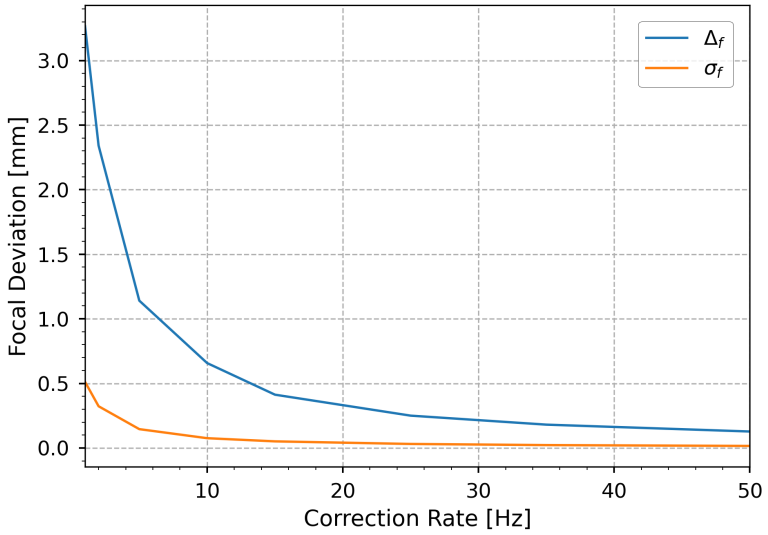
- Linear stage is under consideration for fast lens movements
  - PI precision linear stage is leading candidate for implementation
    - 20 mm travel range, 250 mm/s change, load capacity of 100 N, and 0.02 micron tolerances for incremental motion
  - 2 channel motion controller with USB and SPI interface

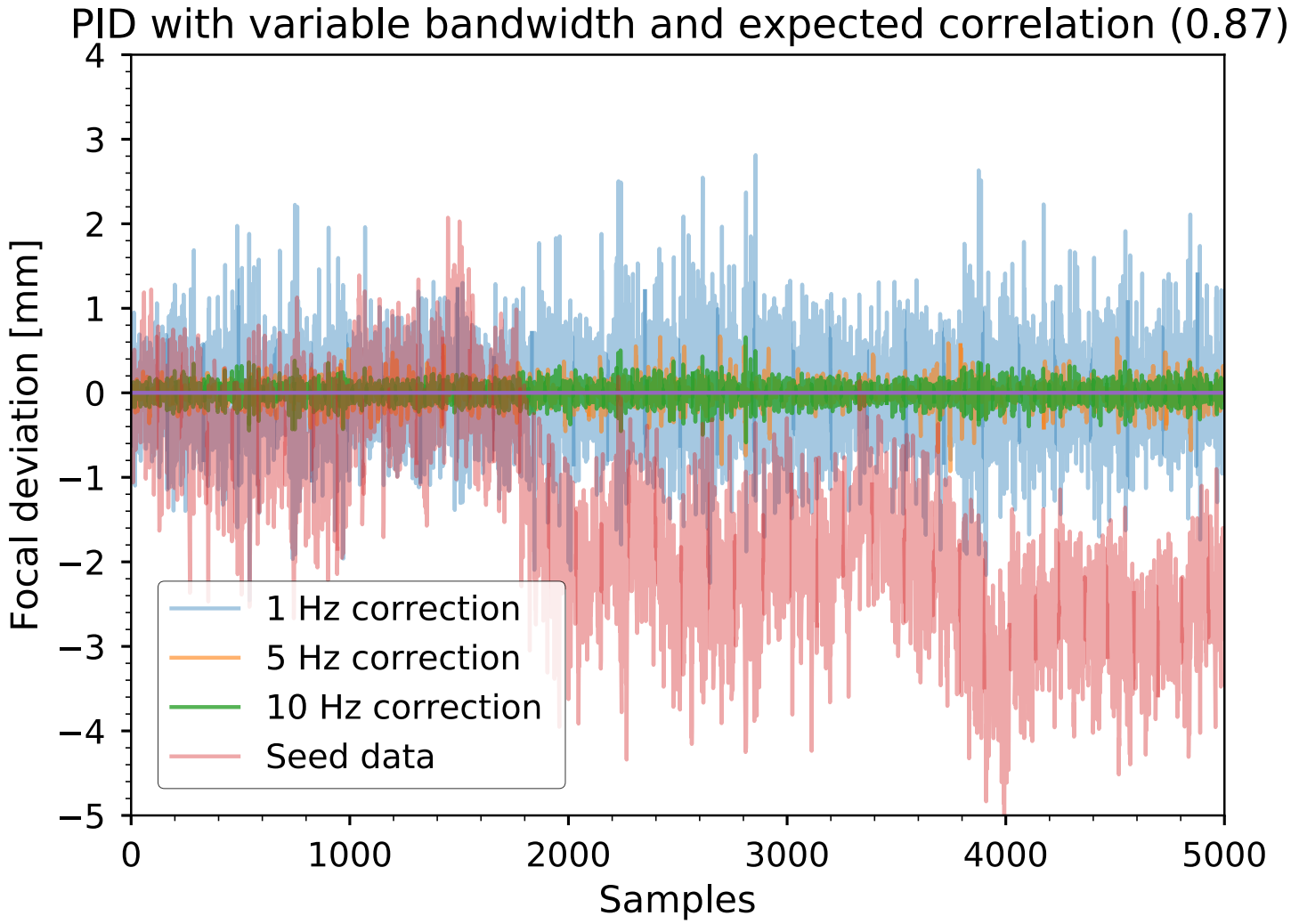




Estimating online performance operating on pre-existing data

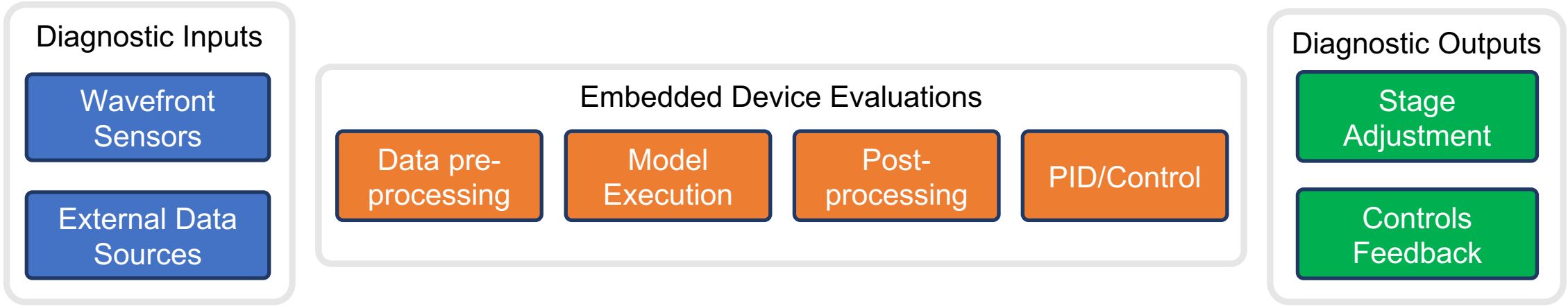
- Simulated PID controller provides an initial correction estimate
  - Existing data sampled and interpolated
- Correction scales with bandwidth
  - 1 Hz sampling is sufficient to remove slow excursions
  - 25 Hz correction reduces maximum deviation to ~0.25 mm and one sigma deviation to 0.031 mm





Data 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

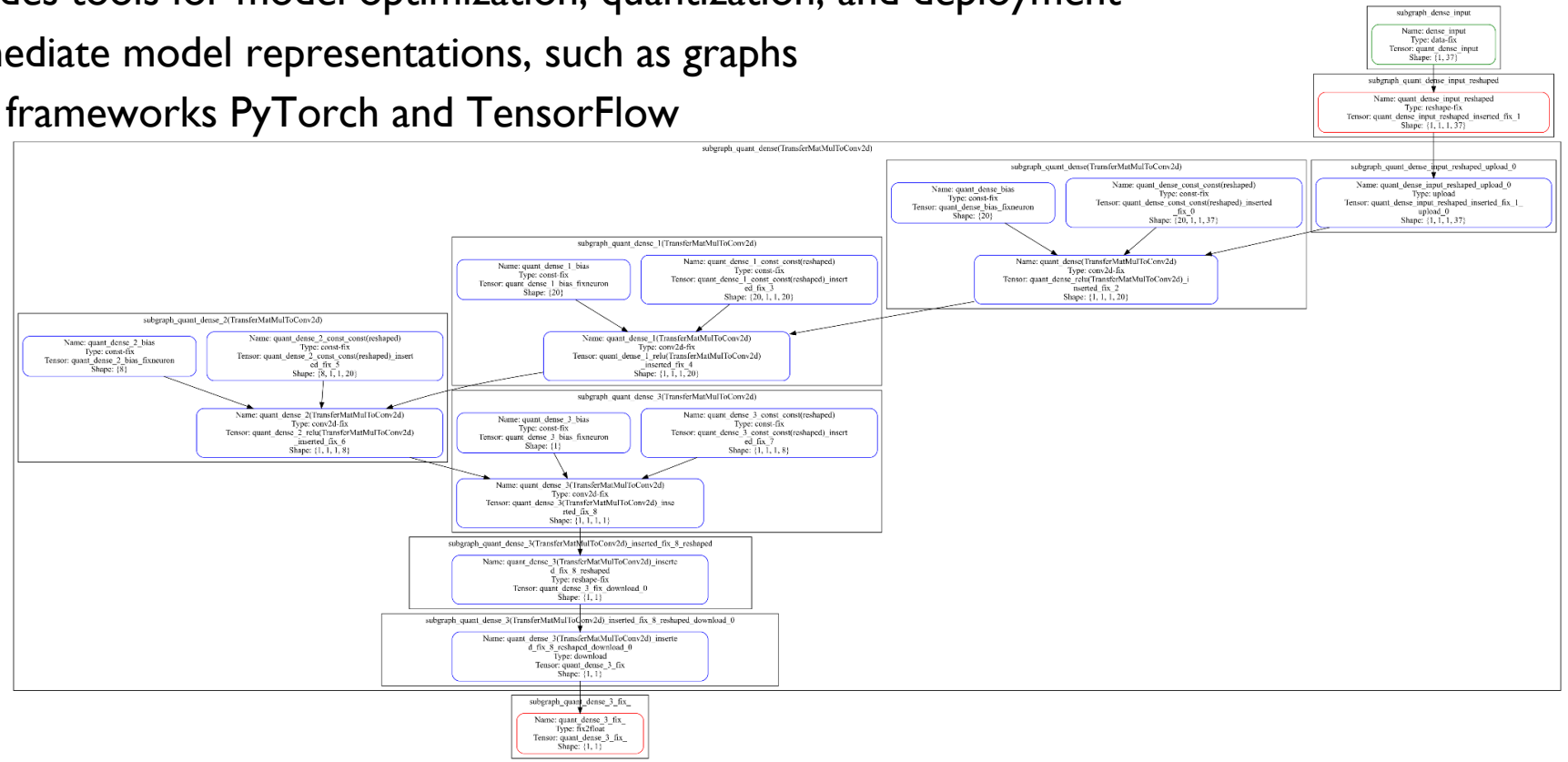
- Our strategy is to employ a fast, embedded device to enact processing and control algorithms

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/)

Exploring a prototype testbed implementation

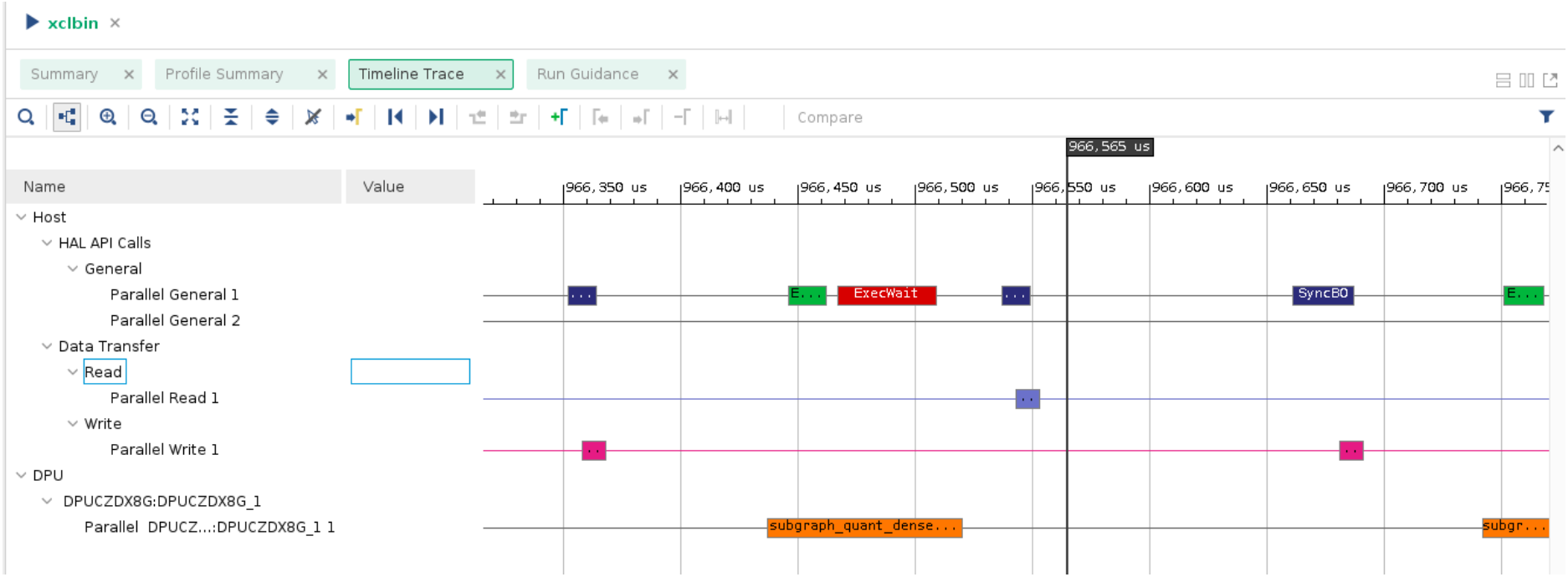
- Testing our implementation with a Xilinx ZCU104
  - Multi-processor system on a chip – embedded CPU, GPU, and programmable logic (e.g. DPU)
  - USB and network interfaces, relatively large memory and I/O bandwidth
- Deployment support via Xilinx utilities
  - Vitis AI interface provides tools for model optimization, quantization, and deployment
  - Utilities include intermediate model representations, such as graphs
  - Supports common ML frameworks PyTorch and TensorFlow



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
```



Conclusions

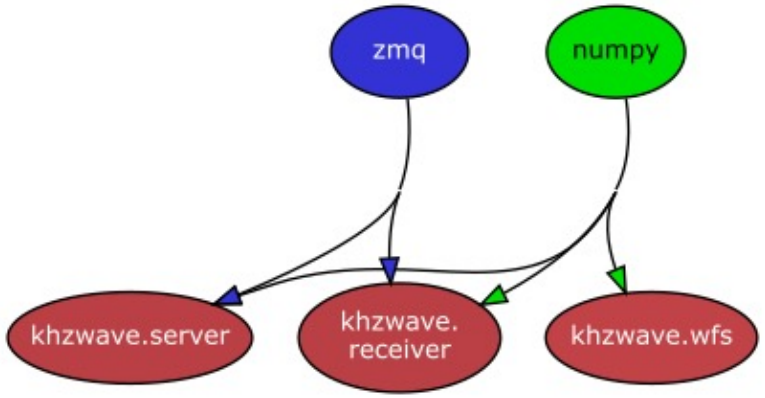
- We are developing a scheme for correcting laser focal position at high repetition rate
  - Our approach integrates fast, non-perturbative measurements with machine learning models to predict the focal position and determine a correction.
  - We aim to develop an in-hardware solution for flexibly deploying and updating the correction model
  - Adjustments will be made using transmissive lenses mounted to a fast linear stage
- 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 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
- Planning for an experimental demonstration is underway

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.

Improving Capture Control and Integration

- Developed a new Python-based API for camera control
  - Python wrappers provide a direct interface to the native camera driver
  - Instrument management and configuration are accessible
  - Streamlines aperture/acquisition parameterization



Index

Super-module

khzwave

Classes

Wfs20Instrument

DEVOFFSET
MAXLINE
STATUS
buildSpotfieldImageArray
calcBeamCentroidDia
calcBeamInformation
calcSpotToReferenceDeviations
calcZernikeLSF
close
configureCamera
displaySpotDeviations
displaySpotIntensities
getDriverRevision
getHighSpeedWindows
getInstrumentCount
getInstrumentInfo
getInstrumentListInfo
getLine
getLineView
getMLAData
getMlaCount
getSpotCentroids
getSpotDeviations
getSpotIntensities
getSpotfieldImage
getStatus
handleError
initInstrument
printStatus
reset
selfTest
setExposureGain
setHighSpeedMode
setReferencePupils
takeSpotfieldImage

Module **khzwave.wfs**

Interface to WFS driver based on sample applications (Windows-only)

This code includes several major classes, of which two are used only by the WfsInstrument class for data storage (WfsImage, WfsInfo)

WfsInfo : data required for managing instrument settings and parameters WfsImage : data buffers and image information

The other classes in this file are used to provide a direct interface to the Thorlabs NI CVI camera driver and to wrap the functions themselves with Python methods

WfsInterface : interface to driver Wfs20Instrument : Instrument management and interaction methods

▸ EXPAND SOURCE CODE

Classes

```
class Wfs20Instrument (dll, *args, instrIdx=0, mlaIdx=0, doInit=True, ignoreErrors=False,
                       **kwargs)
```

Instrument management class

Management methods and interface to get data and configure instrument

Initializes Wfs20Instrument

Args

dll : WfsInterface
interface to driver

instrIdx : int , optional
index of which instrument to use. Defaults to 0.

mlaIdx : int , optional
index of which MLA array to use. Defaults to 0.

doInit : bool , optional
perform initialization of camera. Defaults to True.

ignoreErrors : bool , optional
disables printing from handleError method. Defaults to False.

Raises

RuntimeError