DiffOptics: Imaging Simulation with Differentiable Ray Tracing

Maxime Vandegar, Michael Kagan,
Murtaza Safdari, Sanha Cheong,
Sean Gasiorowski, Ariel Schwartzman

October 27, 2021

Introduction

Optics and imaging simulator development began within context of the Diagnostic Imaging System SLAC is developing

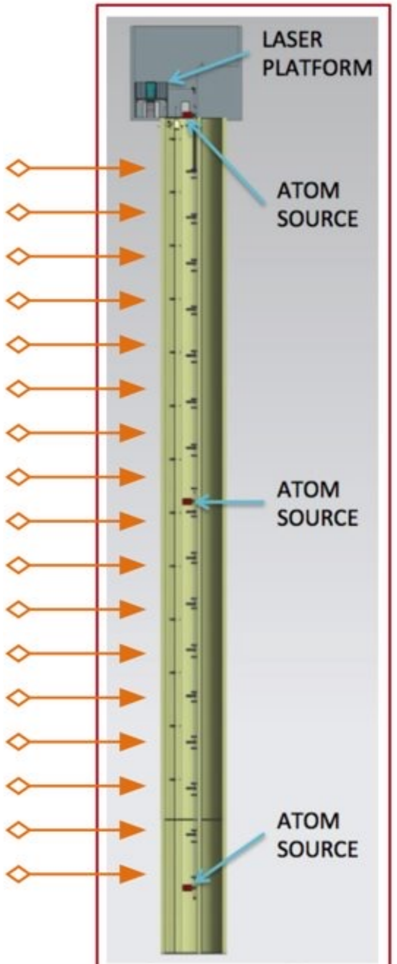
Goals:

- Test different setups, lenses, NA, …
- Understand impact of noise, QE, …
- Understand geometric aberration, Depth of Field, PSF, …
- Estimate wave function phase fits and system capabilities
- Develop new imaging system ideas

Beyond typical optics simulation, want to have simulator gradients:

- Want to use modern ML and rendering methods, and hardware (GPU)
- Want to be able to easily optimize and calibrate optics systems

- *Long term*: Differentiable simulation pipeline optimizable with gradients, and usable within analysis chain?

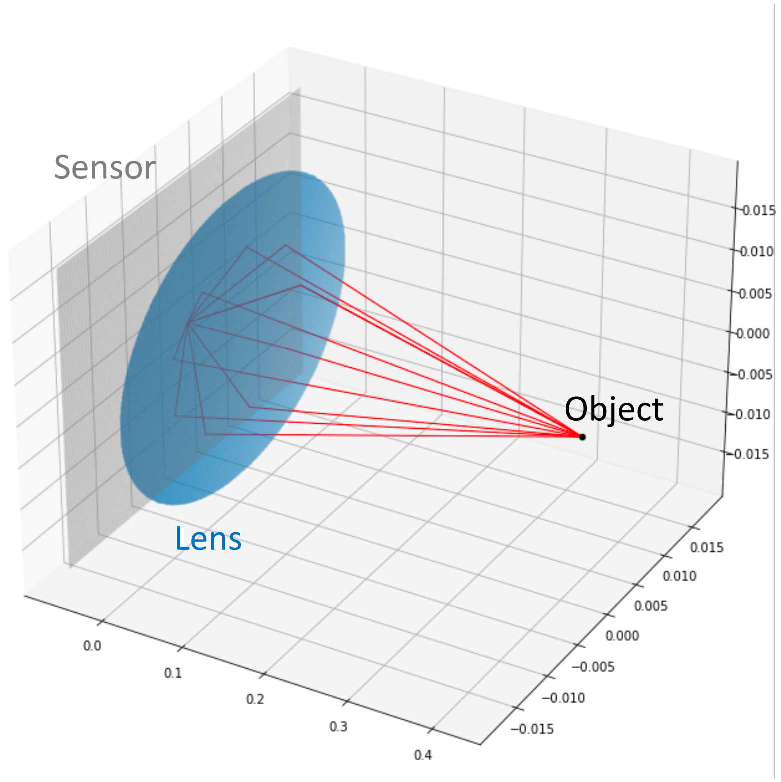


Cross section full detector

Overview

Differentiable-Optics (diffoptics): Simulator of optical system and image generation

- Geometric Optics Approach: Ray tracing
- Diffraction effects from Point Spread Function



Differentiable code enables new optimization and inference schemes

Inference

- Simulator can be used at inference time, i.e. solve A(x)=b
- Can be easily integrated into an ML pipeline

Software and differentiability → Differentiable Programming

Python with *PyTorch* (deep learning framework) backend

- Vectorized code, works on CPU and GPU
- Automatic differentiation framework

Simulations are fully differentiable

- When we write a function: `f(x) {…};`
  automatically get function to evaluate the derivative: `df(x) {…};`

- Can use gradients for e.g. calibration, reconstruction, optimization…
- Seamlessly used in Machine Learning pipelines

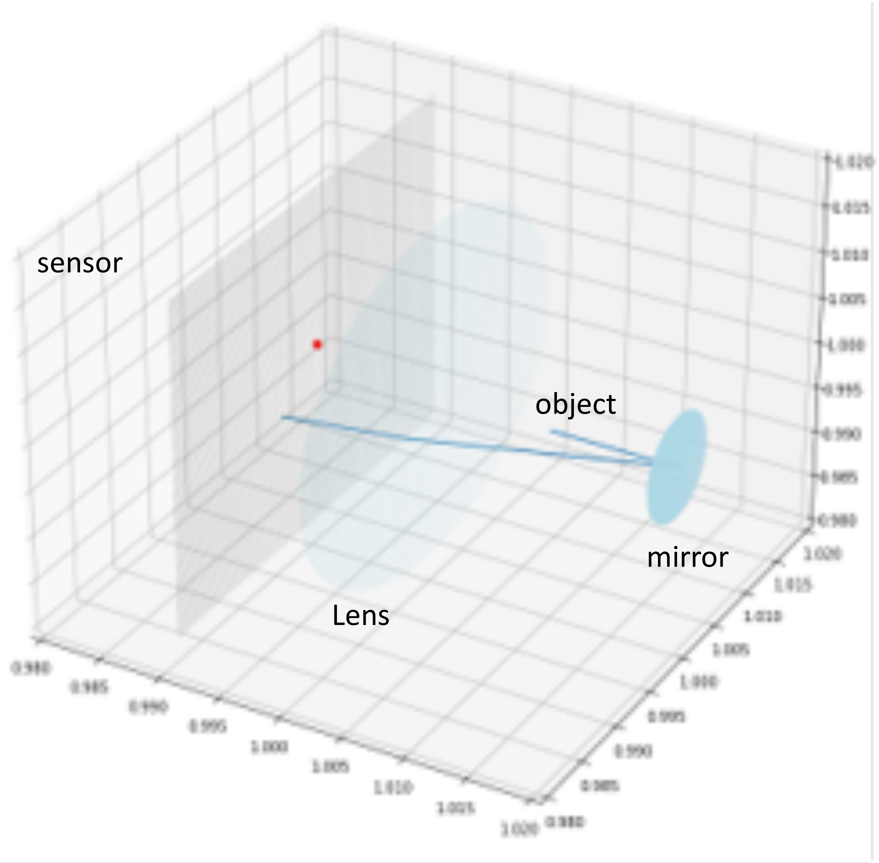
Software design based on graphics / rendering engines

Optics elements

Optical elements organized in a "scene":
Positions and orientations can be specified

Optical Elements implemented:

- Ideal (thin) lens and thick lens (in progress)
- Mirrors
- Windows
- Sensor
  - Can specify resolution, quantum efficiency, noise
  - Ongoing work to add electronics noise model
- Bounding box

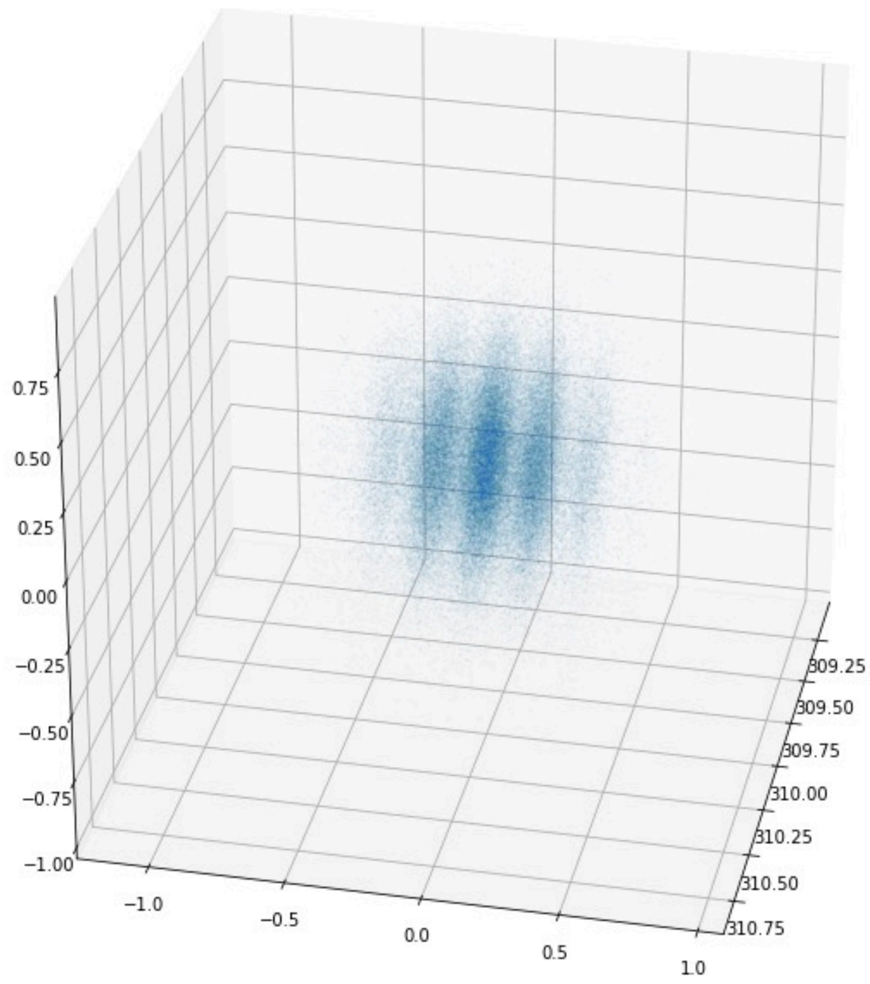


Every optical element has a `get_ray_intersection` method to determine where rays interact with elements

Atom Clouds and Sampling Tools

Atom cloud shape can be specified

- "Standard" wave function implemented
- Can specify density, position, imaging time, etc.
- Can specify a method to sample rays from cloud



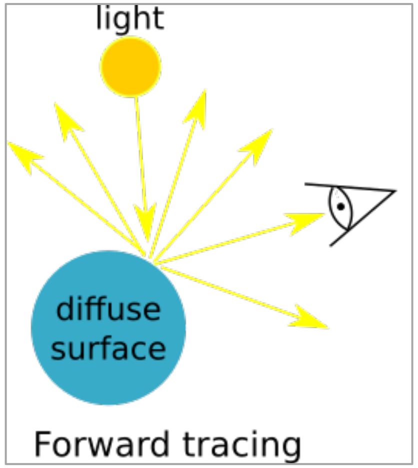
For arbitrary cloud densities, no direct method to sample rays

- Rejection sampling in (specifiable) bounded area
- Importance sampling in (specifiable) bounded area

Ray Tracing

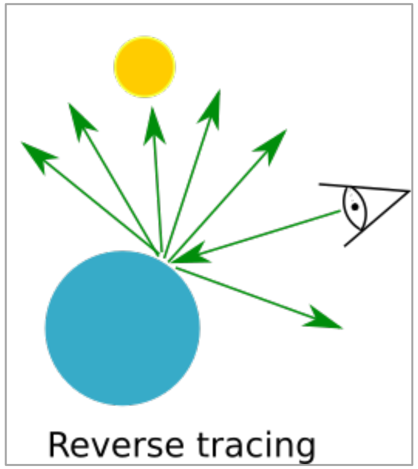
Forward ray tracing (Object to Sensor)

- Sample ray position and direction, trace forward through system



Reverse Ray tracing (Sensor to Object)

- Position and angle sampling from each pixel
- Integration of cloud density along rays in (specifiable) bounded area



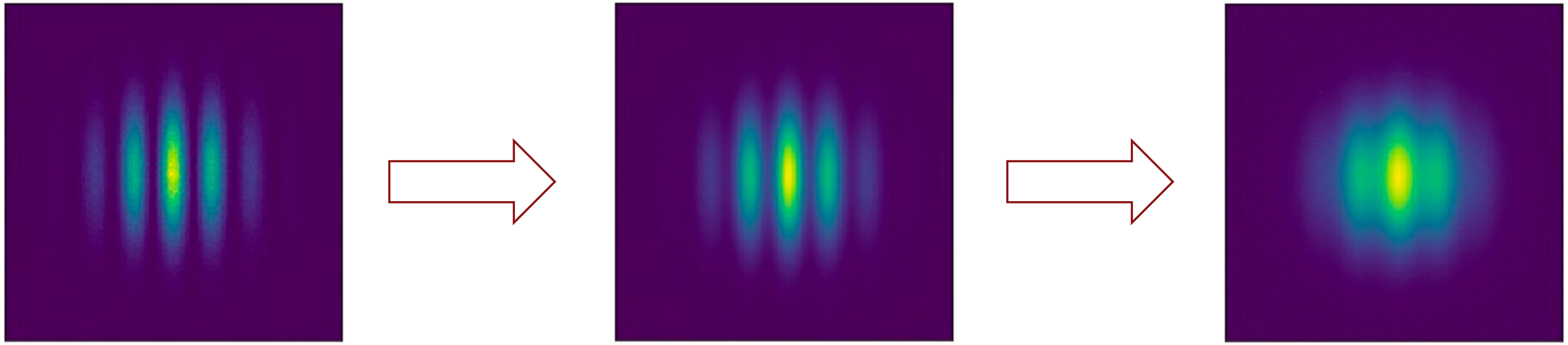
Vectorization and GPUs allow us to greatly speed up this process

SLAC

Point Spread Functions

PSF implemented as image post-processing step

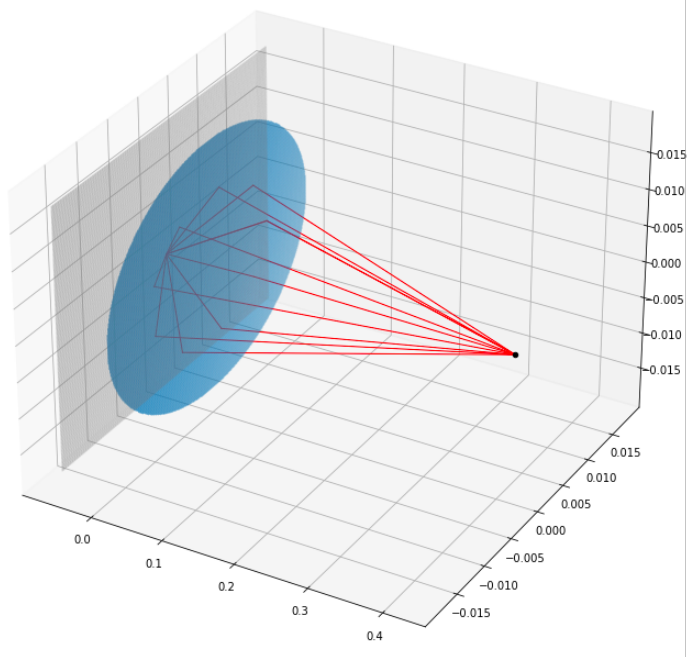
- No diffraction / interference in ray tracing
- Impact implemented through convolution
- Position and depth dependent PSF can be specified

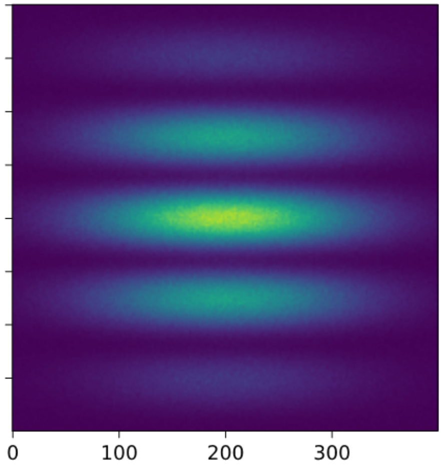


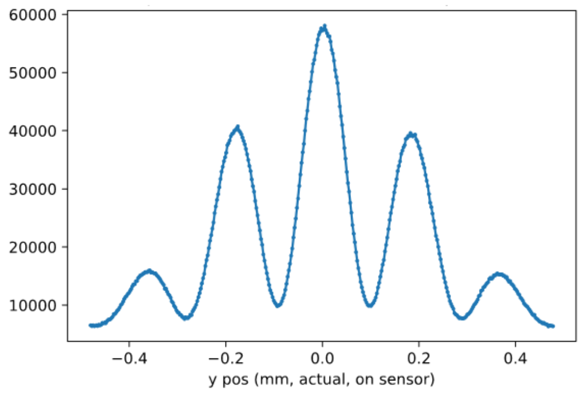
PSF must be measured / simulated elsewhere

- No first principles calculation (could be included if desired)
- Many lenses we want to buy do not have open source models in any case

Image generation + Fit



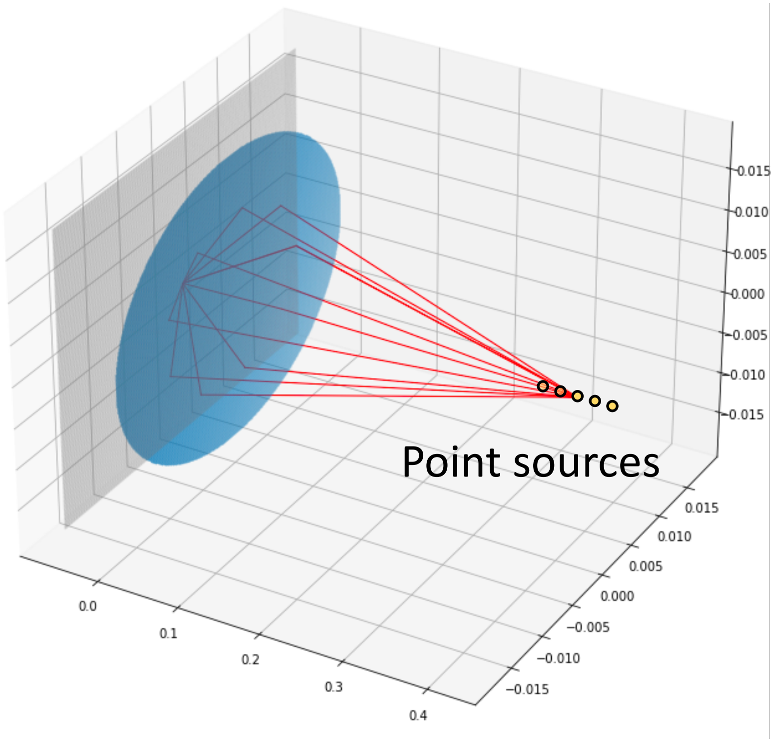




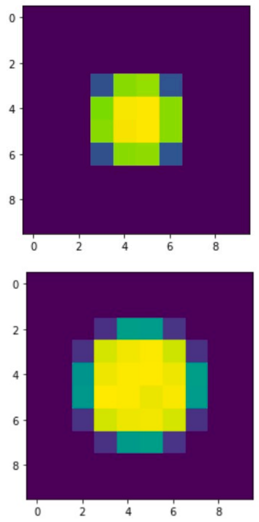
ϕ

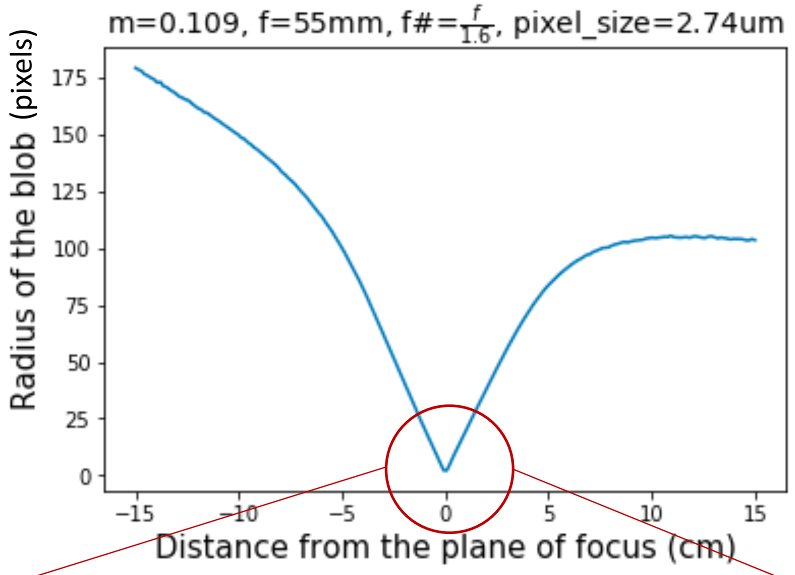
SLAC

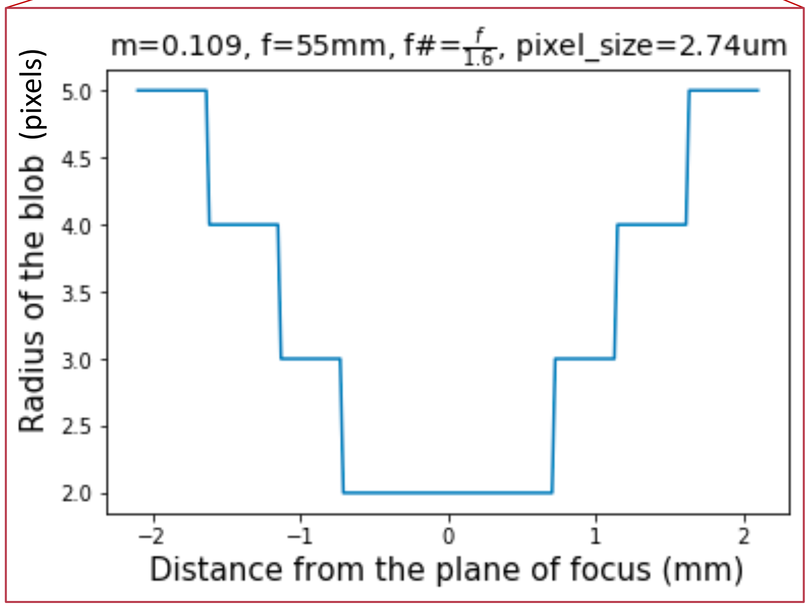
Probing Depth of Field



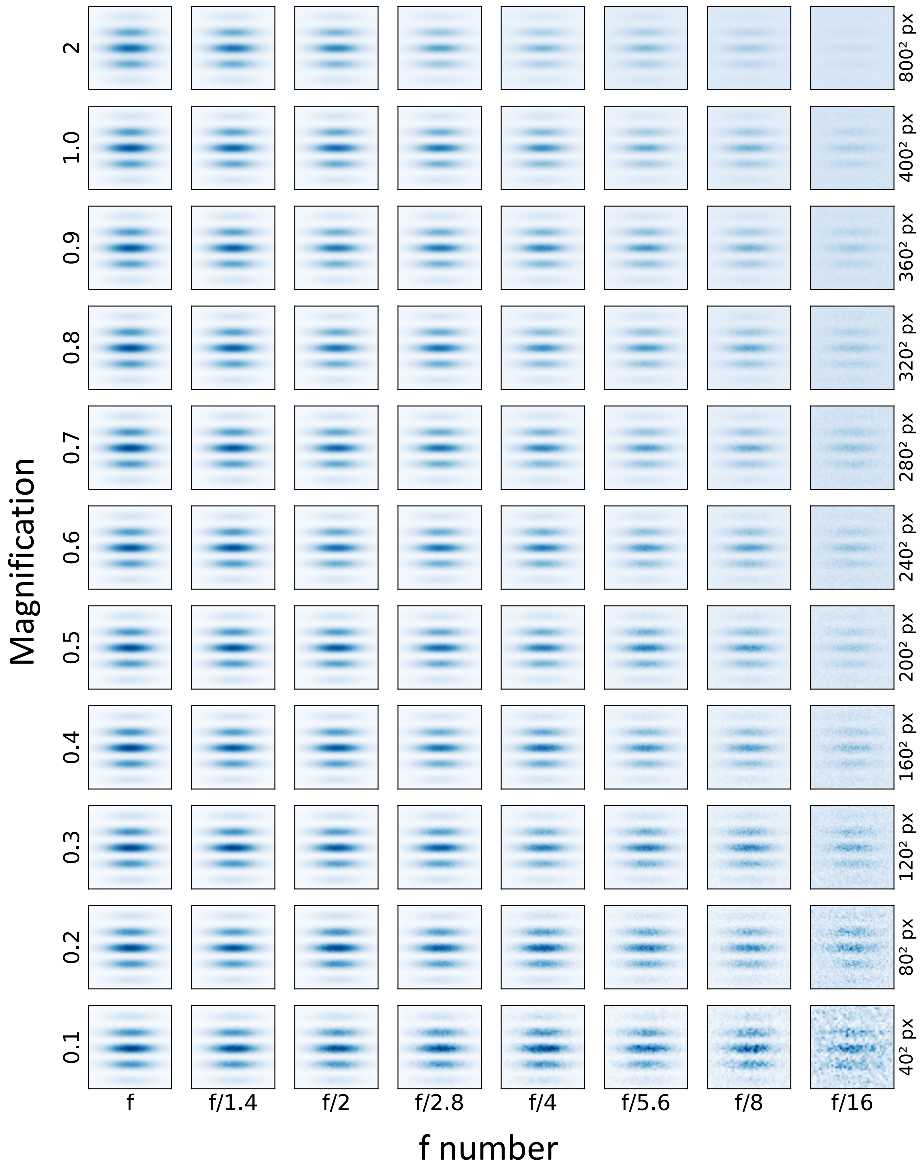
Circle of Confusion

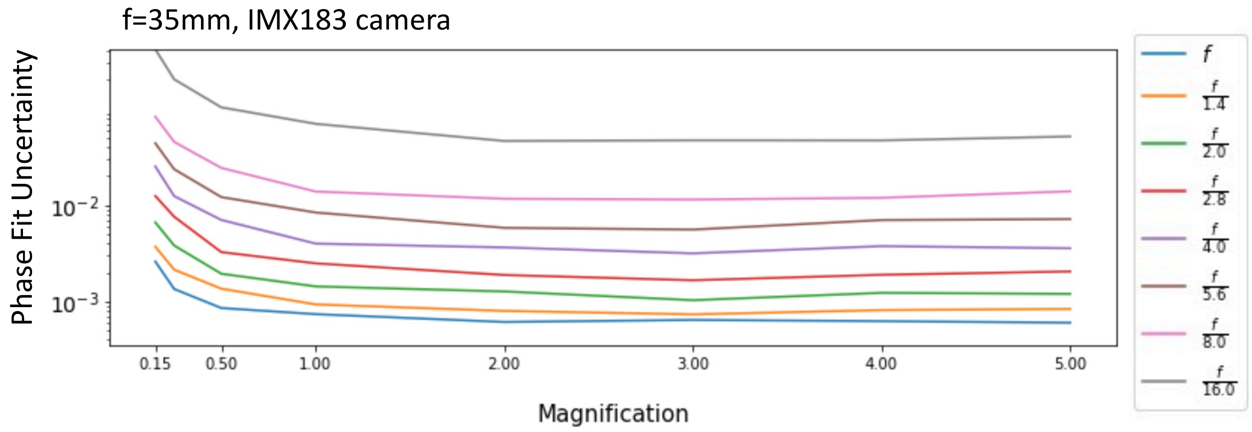




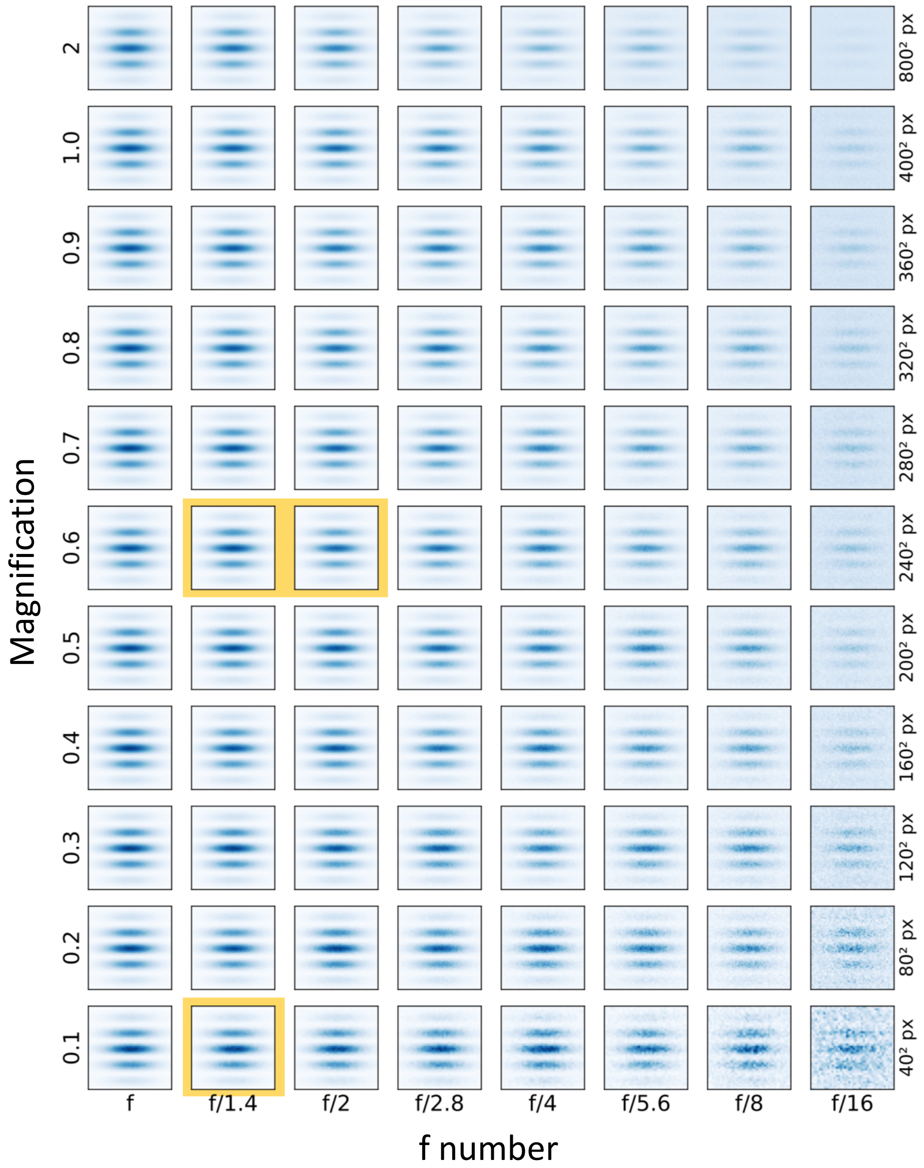


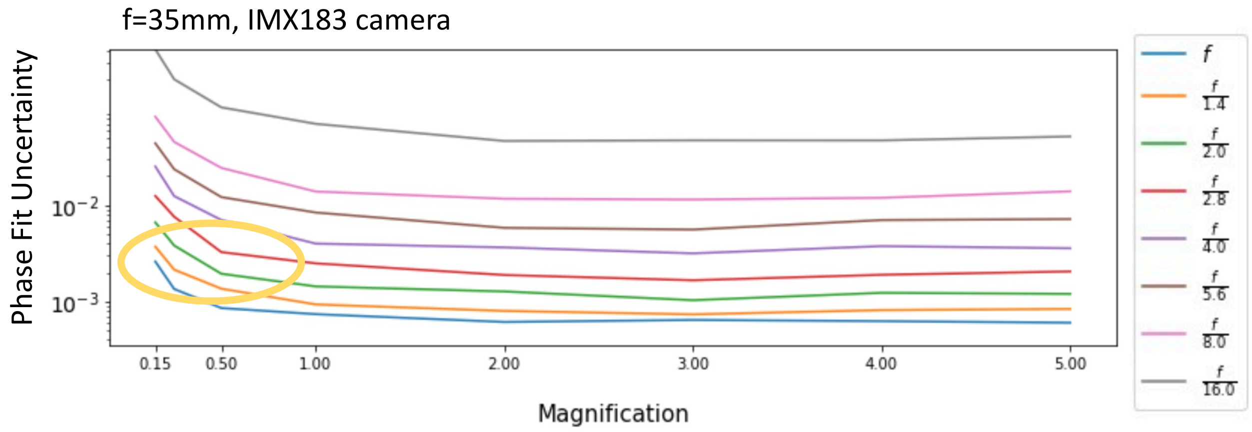
Application to MAGIS Diagnostic Imaging System Design: Testing different setups



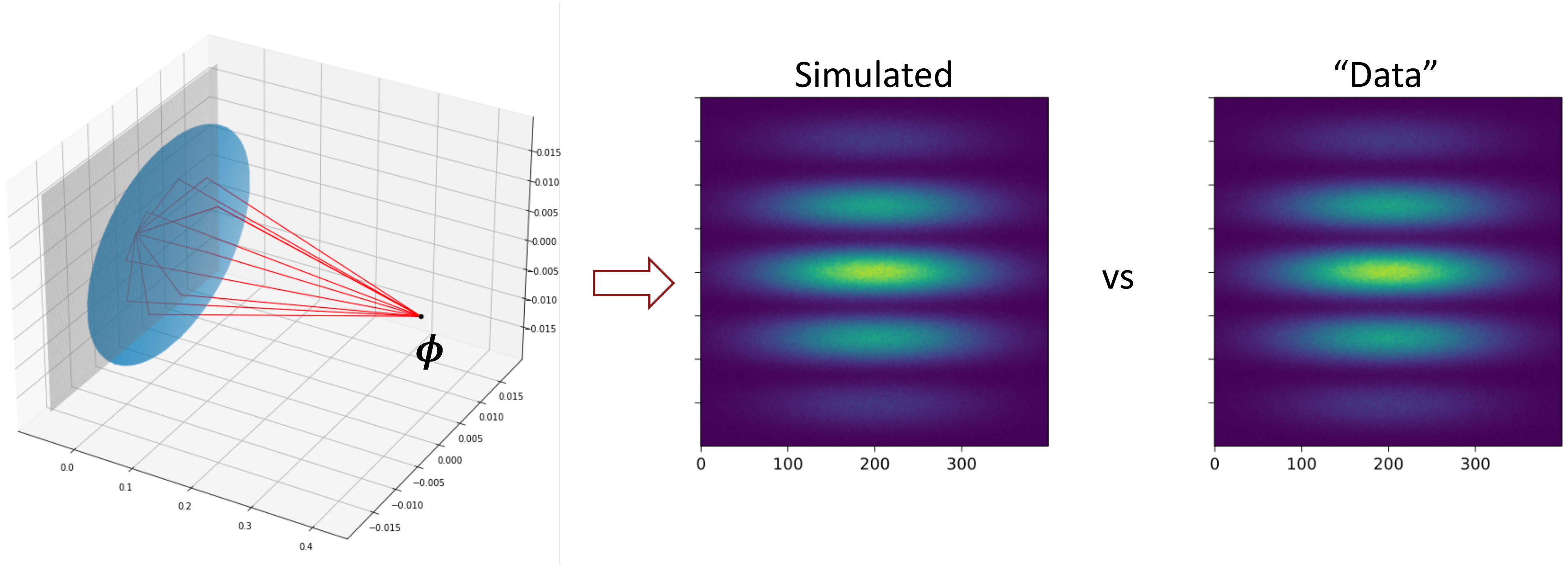


Application to MAGIS Diagnostic Imaging System Design: Testing different setups





Fit directly with Differentiable Simulator



Can also fit phase directly through simulator

Cloud(ϕ) → Simulated image X_ϕ → Compare to data Y (e.g. with Likelihood)

$$\phi^* = \arg\min_{\phi} L(Y, X_\phi)$$

Design / Calibration with gradient descent

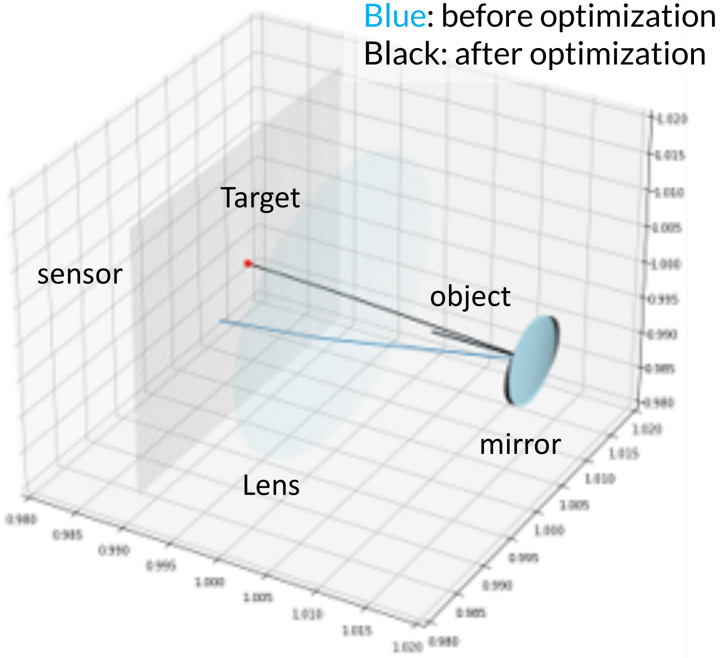
Simulated images are differentiable w.r.t. optical element parameters (size, shape, position, tilt…)

Can define optimization objective for specific goals

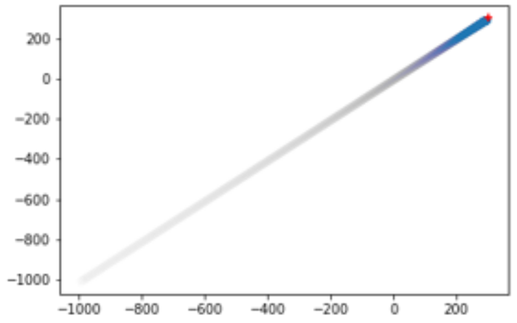
- Design Optimization
- Simulator Calibration
- Optimal Experiment Design

Can do optimization with gradient descent!

Blue: before optimization
Black: after optimization



Optimize mirror angle



Position of simulated object on sensor during optimization

Getting Started

"Getting Started" tutorial provided in Jupyter Notebook

Singularity and Docker containers with all needed dependencies are available

- Can run on CPU and GPU

Code currently in private GitHub repo

- Happy to give access
- Still plenty to do → come contribute
- Aim to make it public soon!

```python
import diffoptics as optics
sensor = optics.Sensor(position=(-f * (1 + m), 0, 0), poisson_noise_mean=2)
scene = optics.Scene(sensor)
lens = optics.PerfectLens(f=f, m=m)
scene.add_object(lens, False)
atom_cloud = optics.AtomCloud(position=(cloud_x_pos, 0., 0.))
atom_cloud = optics.LightSourceFromDistribution(atom_cloud)
for batch in range(20): # Trace 1B rays with minibatches of 50M rays
  rays = atom_cloud.sample_rays(50e6, device='cuda)
  photon_mapping(rays, scene)
img = sensor.readout()
```

```python
# Study the impact of DoF
import numpy as np
import diffoptics as optics
sensor = optics.Sensor(position=(-f * (1 + m), 0, 0), poisson_noise_mean=2)
scene = optics.Scene(sensor)
lens = optics.PerfectLens(f=f, m=m)
scene.add_object(lens, False)
for delta_x in np.linspace(-1, 1, num=100):
  atom_cloud = optics.AtomCloud(position=(cloud_x_pos + delta_x, 0., 0.))
  atom_cloud = optics.LightSourceFromDistribution(atom_cloud)
  for batch in range(20): # Trace 1B rays with minibatches of 50M rays
    rays = atom_cloud.sample_rays(50e6, device='cuda)
    photon_mapping(rays, scene)
  img = sensor.readout()
```

Conclusion

Active development of fully differentiable ray tracing simulator

- Novelty: merging differentiable ray tracing and optics
- Not MAGIS specific, useful to other experiments and applications

Broad applicability from physics imaging to experiment design optimization

Near-term plans to expand

- PSF recently merged into main code
- Implementation of optical elements to model compound systems with thick lenses
- Transmission / reflectance
- Could include optical path length for estimating diffraction effects