Optimization of Metasurfaces for Computational Imaging

Dean Hazineh | PhD Candidate Applied Physics

Advisors: Todd Zickler, Federico Capasso

CSE Oral Presentation

April 26, 2024

Conventional Imaging Systems

1940 2010 2020

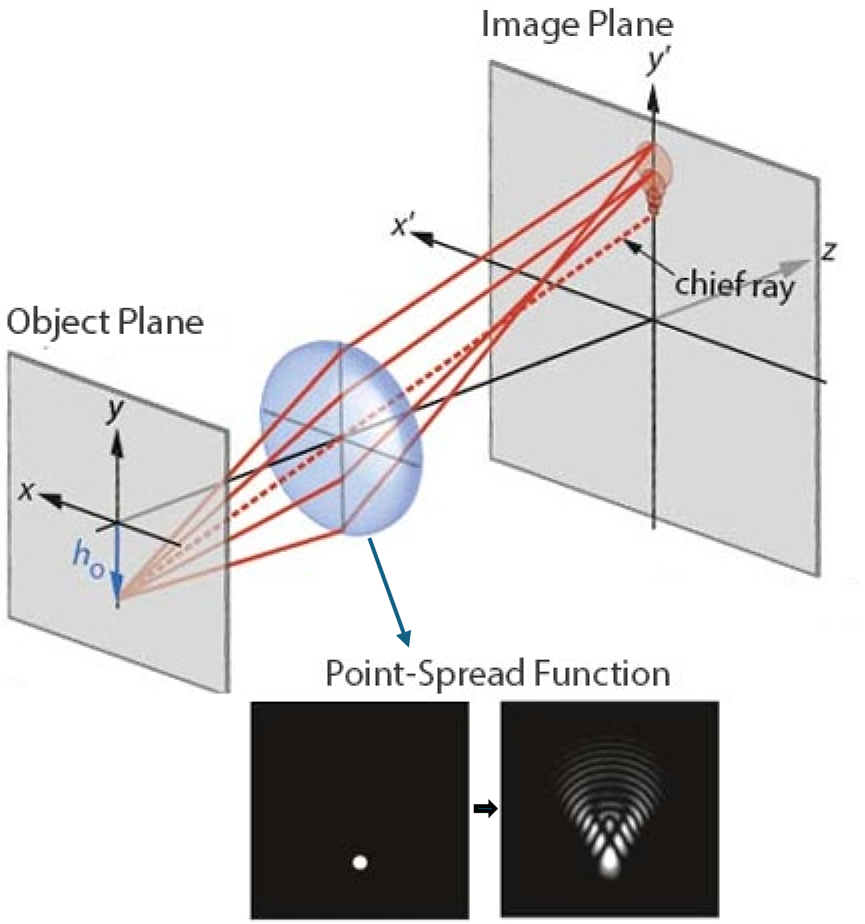
1839: First photographic lens – modern cameras
1550: Images made w/ concave mirror & lenses
-500 (BC): Camera Obscura (Pinhole Camera)

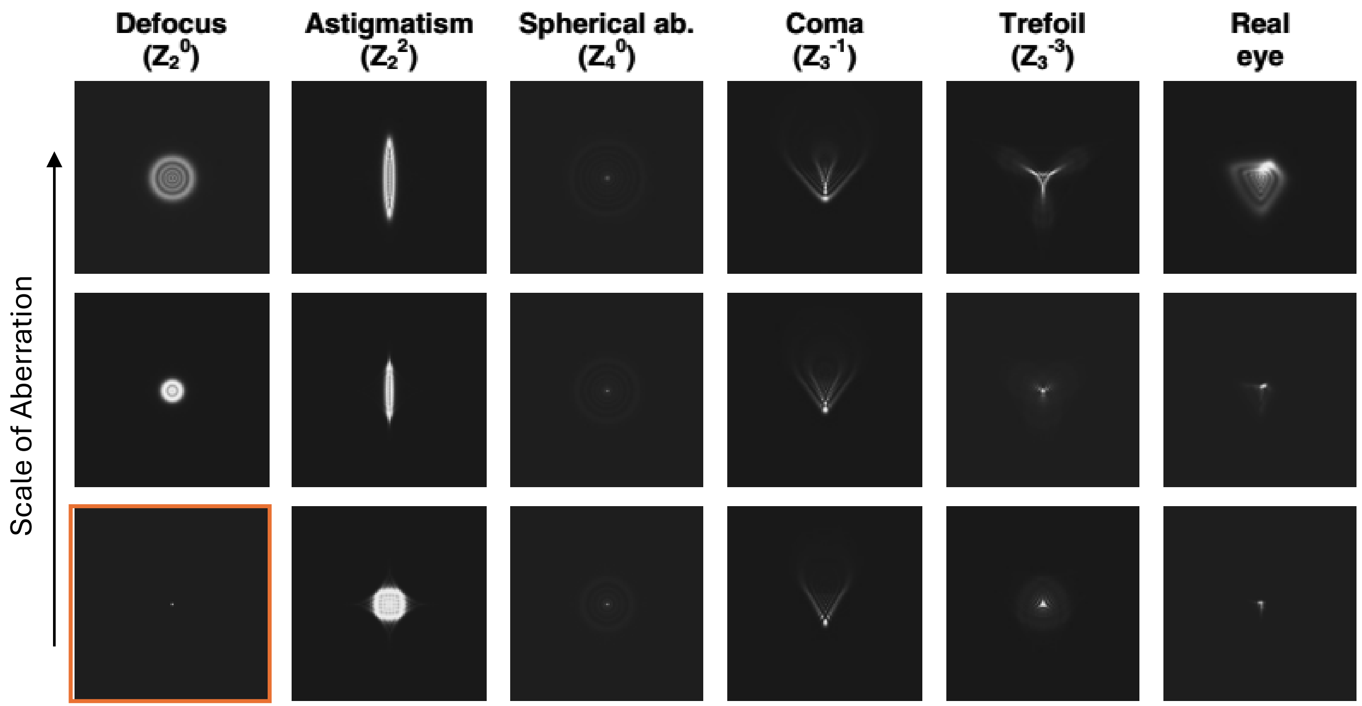
Design principle of all conventional cameras is largely the same:

- A single point in the scene should project to a single point on the photosensor
- Captured measurements are *undistorted projections* of scene, *close* the *final image*

→ The design of computational imaging systems deviates from this idea

Centuries pursuit of the ***Ideal* Lens**





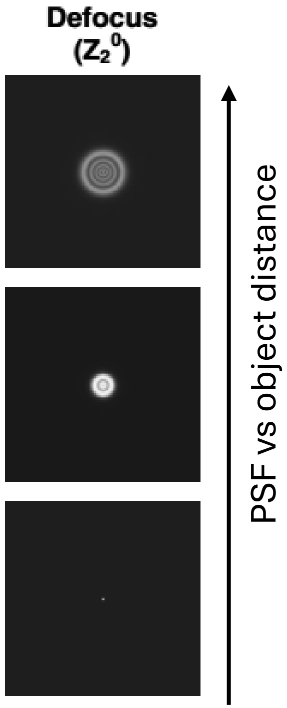
→ Moving away from *ideal* focusing to **structured point-spread functions** enable better vision systems and cameras

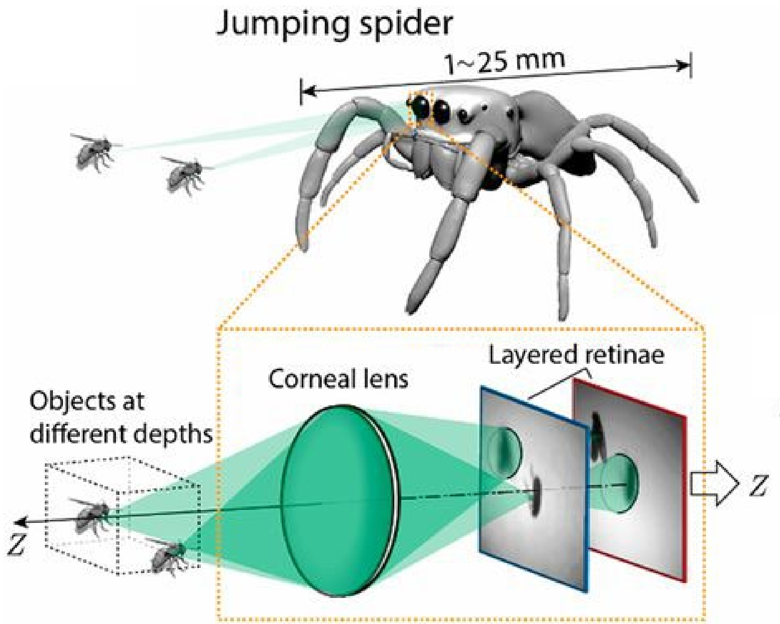
→ How we engineer the point-spread function for computational imaging with a new type of lens**, metasurfaces**

[1] Effect_of_third-order_aberrations_on_dynamic_accommodation

[2] https://wiley-vch.e-bookshelf.de/products/reading-epub/product-id/5030654/title/Optical

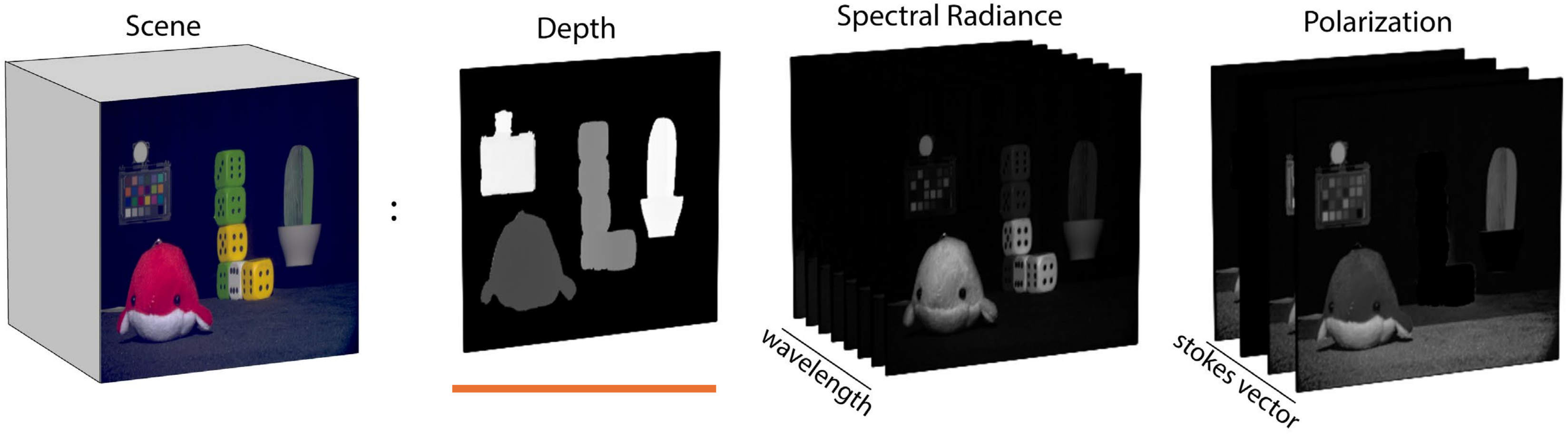
Biological vision: Depth from Defocus





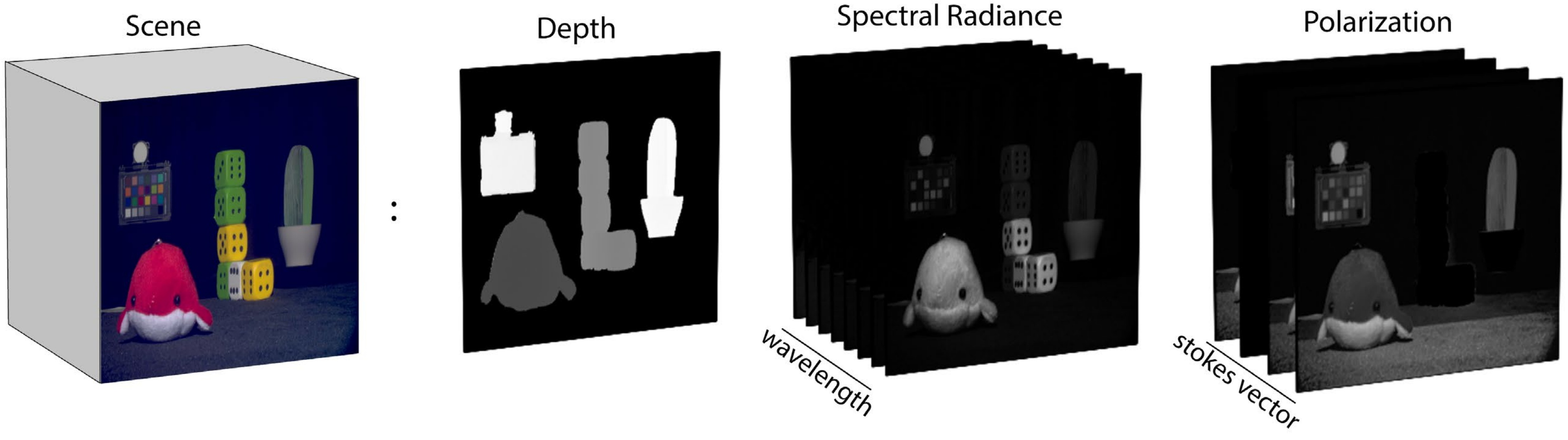
Qi Guo et al., Compact single-shot metalens depth sensors inspired by eyes of jumping spiders, 2019 PNAS 116 (46) 22959-22965

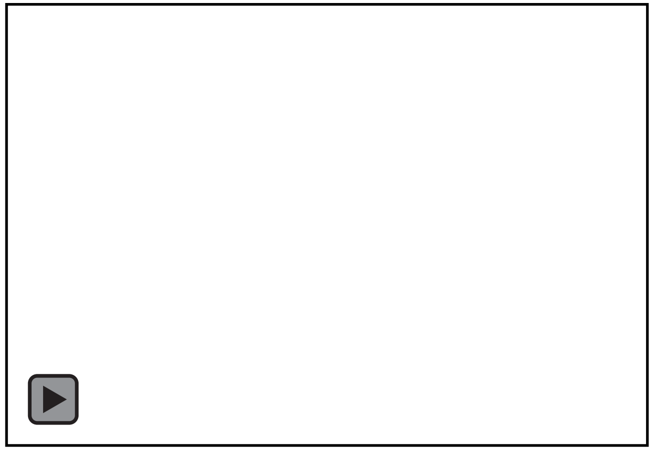
Features of a Scene: **Depth, Spectral Radiance, Polarization**

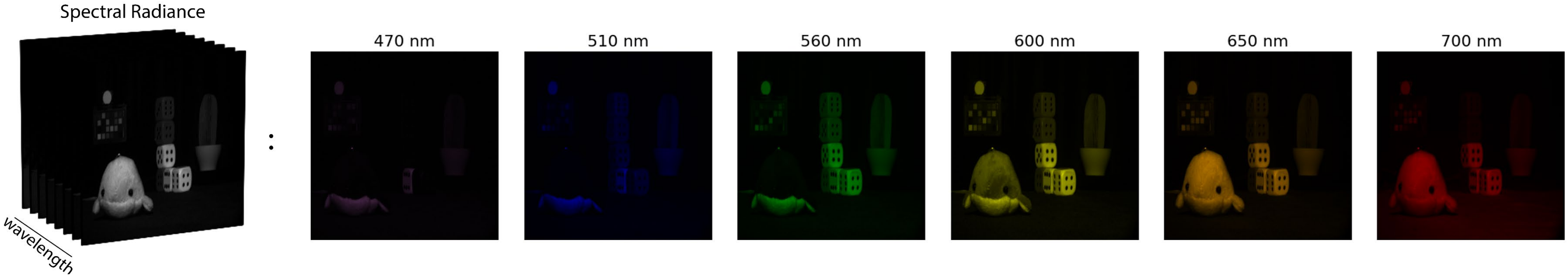


[3] Hyperspectral Depth Dataset From: https://vclab.kaist.ac.kr/iccv2021/dataset.html

Features of a Scene: **Depth, Spectral Radiance, Polarization**

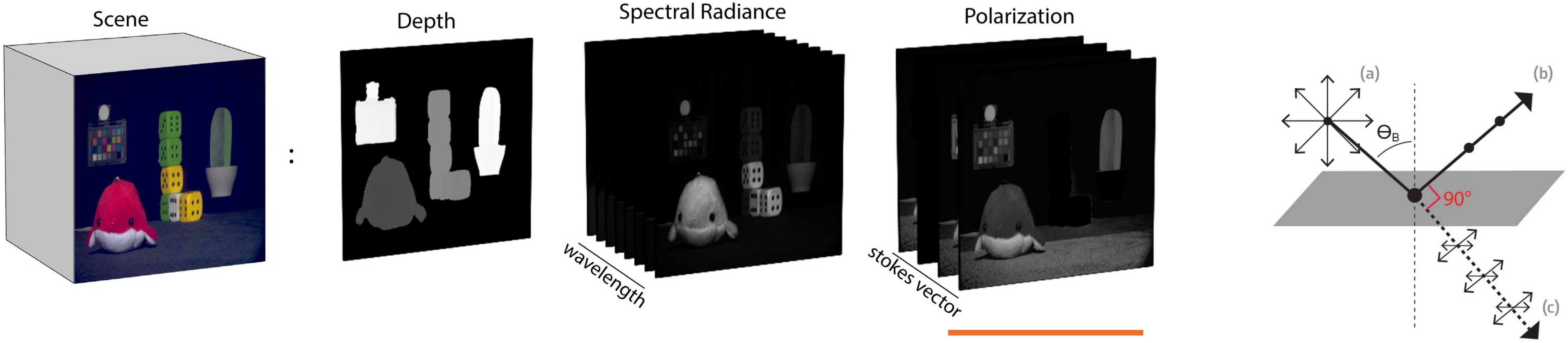


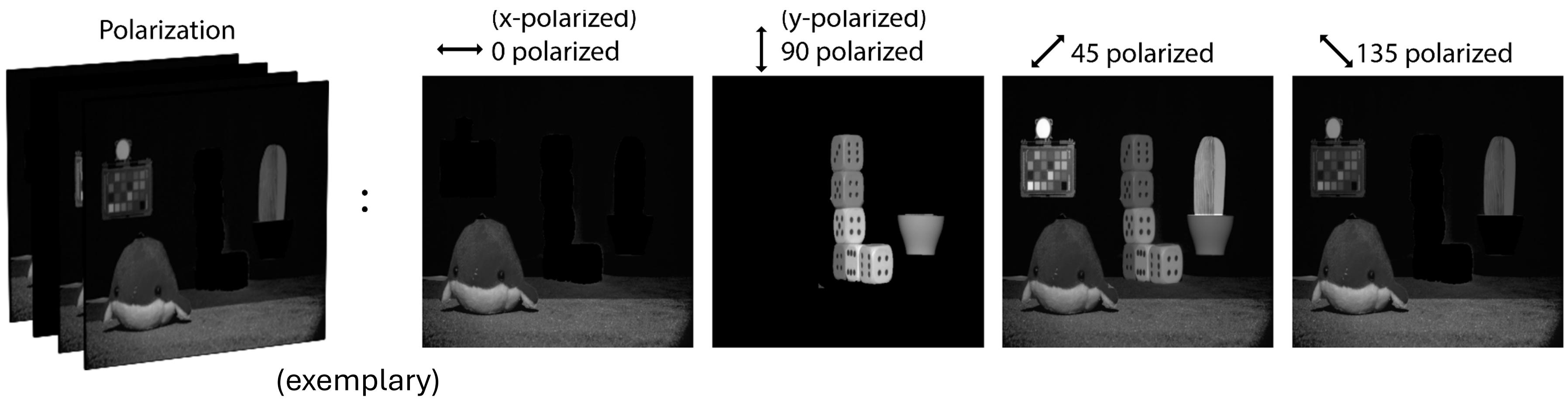




[3] Hyperspectral Depth Dataset From: https://vclab.kaist.ac.kr/iccv2021/dataset.html

Features of a Scene: **Depth, Spectral Radiance, Polarization**

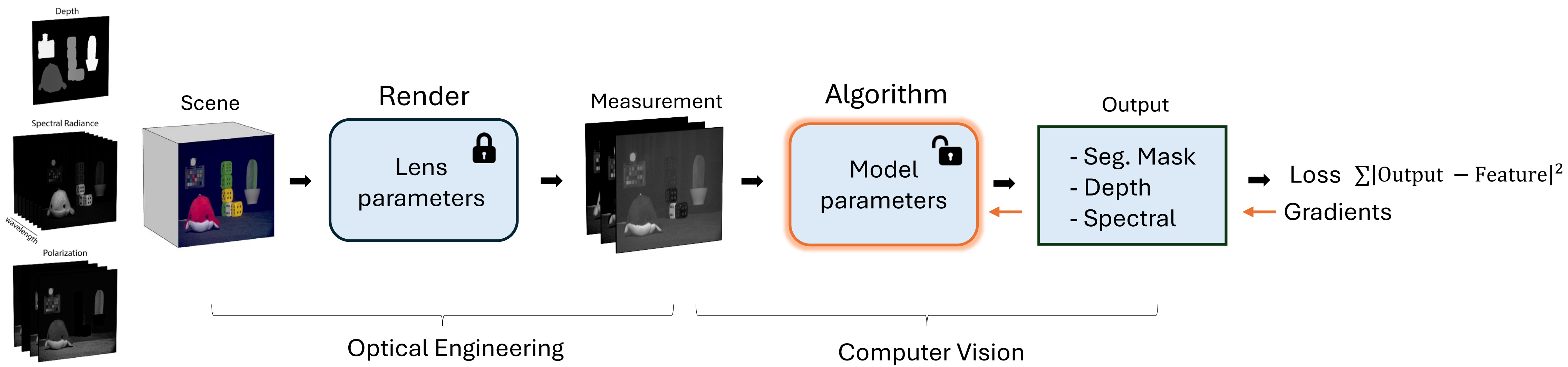




All three features of a scene can in principle be measured in a single snapshot then reconstructed with computational imaging, if they are optically encoded with a specialized lens (feature-dependent point-spread function)

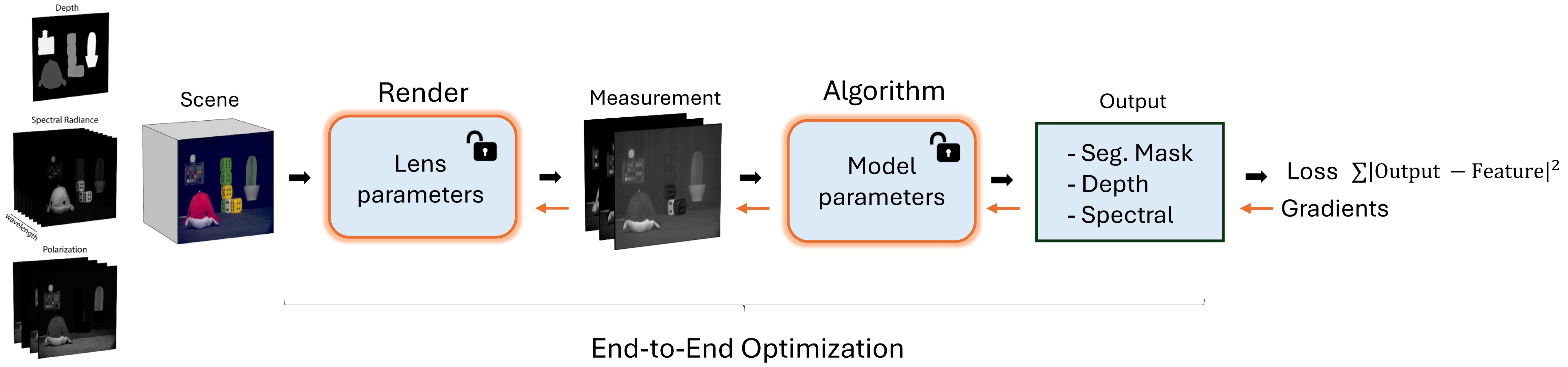
Computational Imaging and Sensing

- **How** can we recover the full information of a scene (or best encode the quantities in the measurement)
- **Why**: We require more than just a 2D spatial map of a scene (*ideal* image) to interact with the world (e.g. AR/VR)
  - Material ID/Classification - Segmentation
  - Depth (3D modeling) - Scientific sensing (wavefront phase, angle of incidence…)



Computational Imaging and Sensing

- **How** can we recover the full information of a scene (or best encode the quantities in the measurement)
- **Why**: We require more than just a 2D spatial map of a scene (*ideal* image) to interact with the world (e.g. AR/VR)
  - Material ID/Classification
  - Depth (3D modeling)
  - Segmentation
  - Scientific sensing (wavefront phase, angle of incidence...)



Render

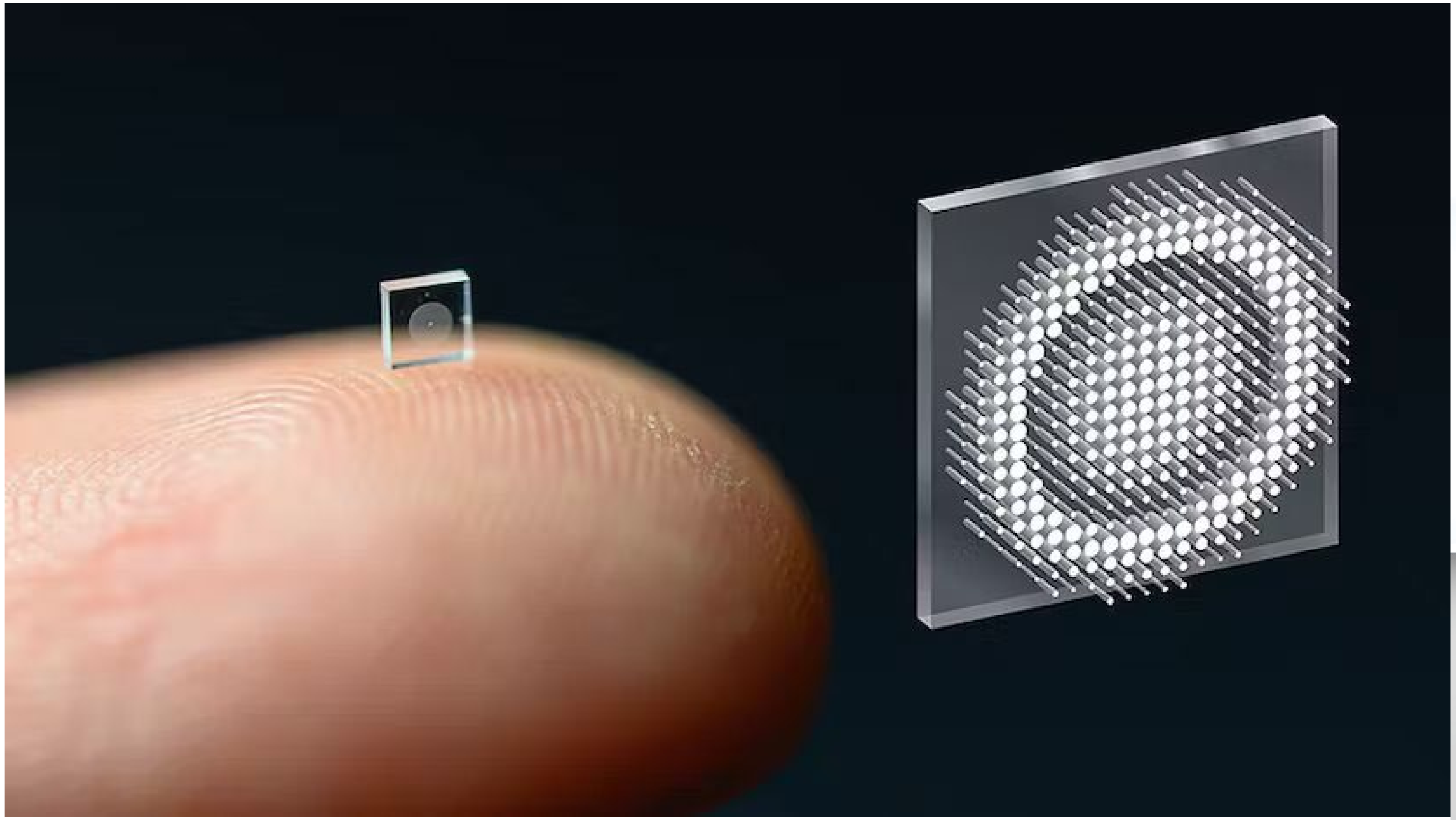
Lens parameters

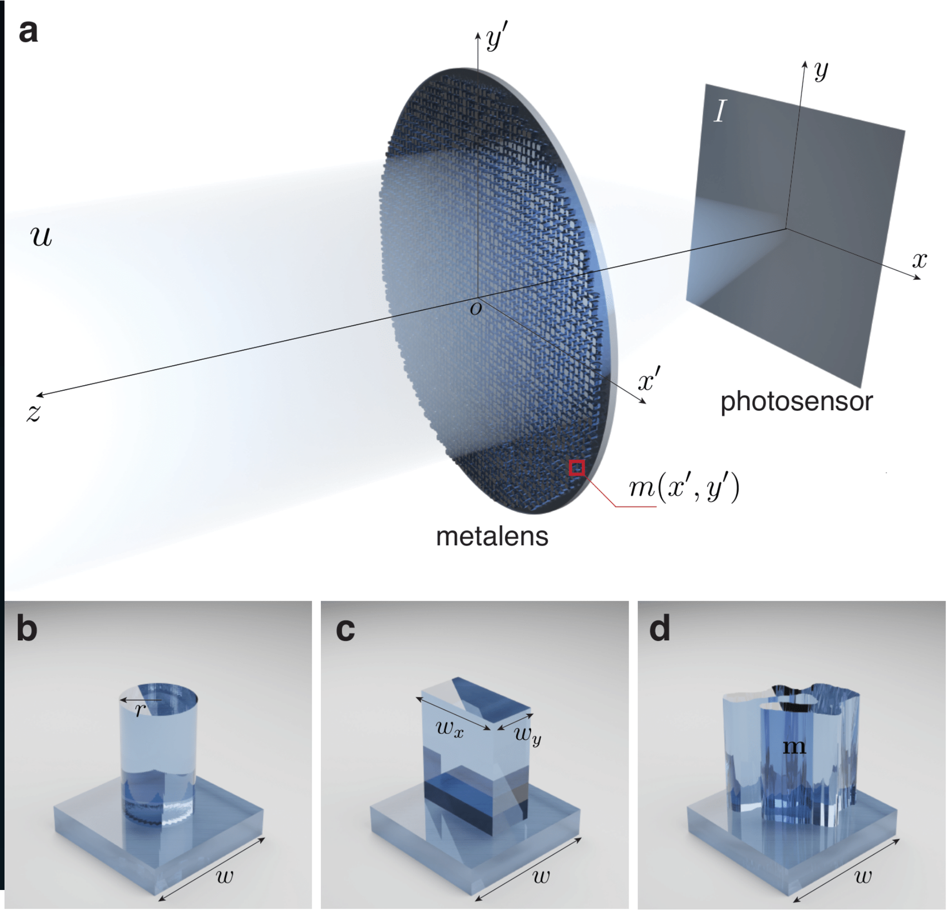
D-Flat

Open-source software (Tensorflow & Pytorch)
Auto-differentiable framework specially geared to co-optimization of flat optics with post-processing networks

D-Flat: A Differentiable Flat-Optics Framework for End-to-End Metasurface Visual Sensor Design, D. Hazineh et al, 2022

Metasurfaces (Metalens)





(Left) Photo E. Tseng et al., Neural Nano-Optics for High-quality Thin Lens Imaging

Metalenses can transform and structure incident light in ways that other devices cannot!

- **Polarization, depth, and wavelength dependent point-spread functions**

Metasurfaces (Metalens)

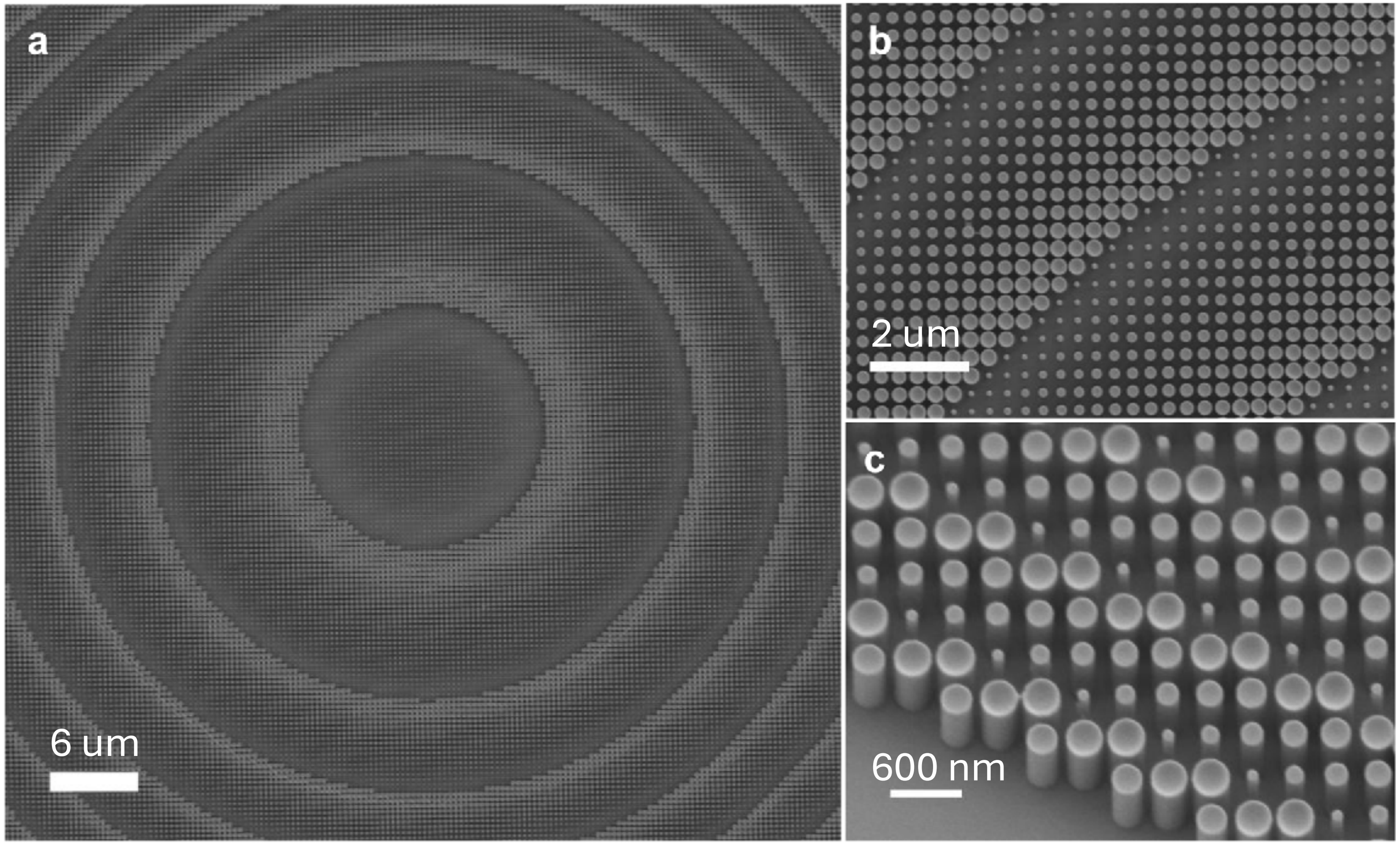
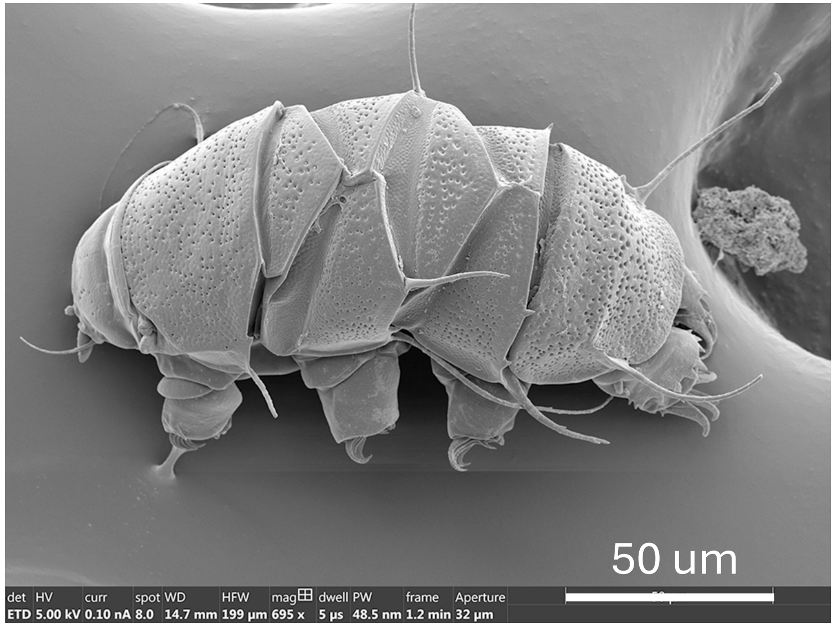
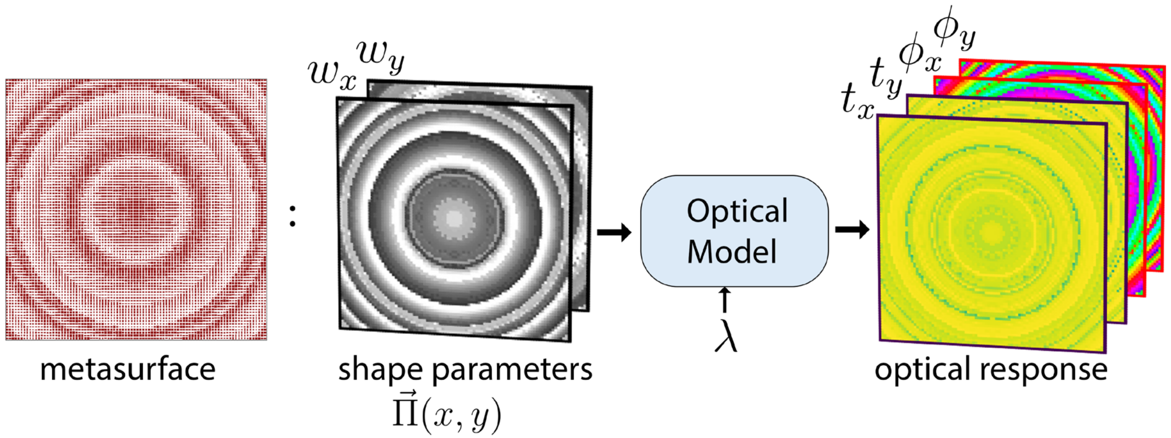


Photo: M. Khorasaninejad et al. (Capasso Group)

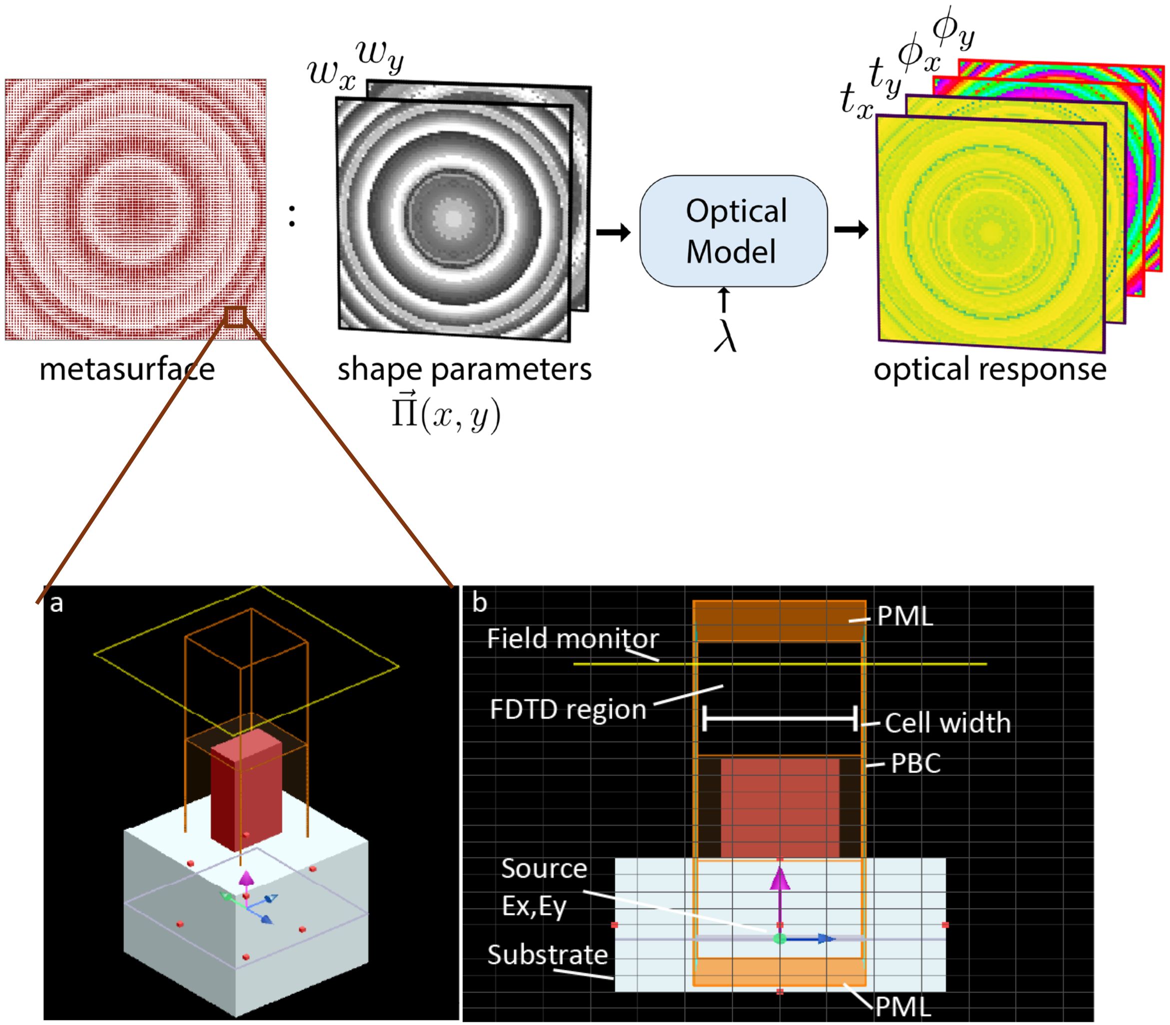


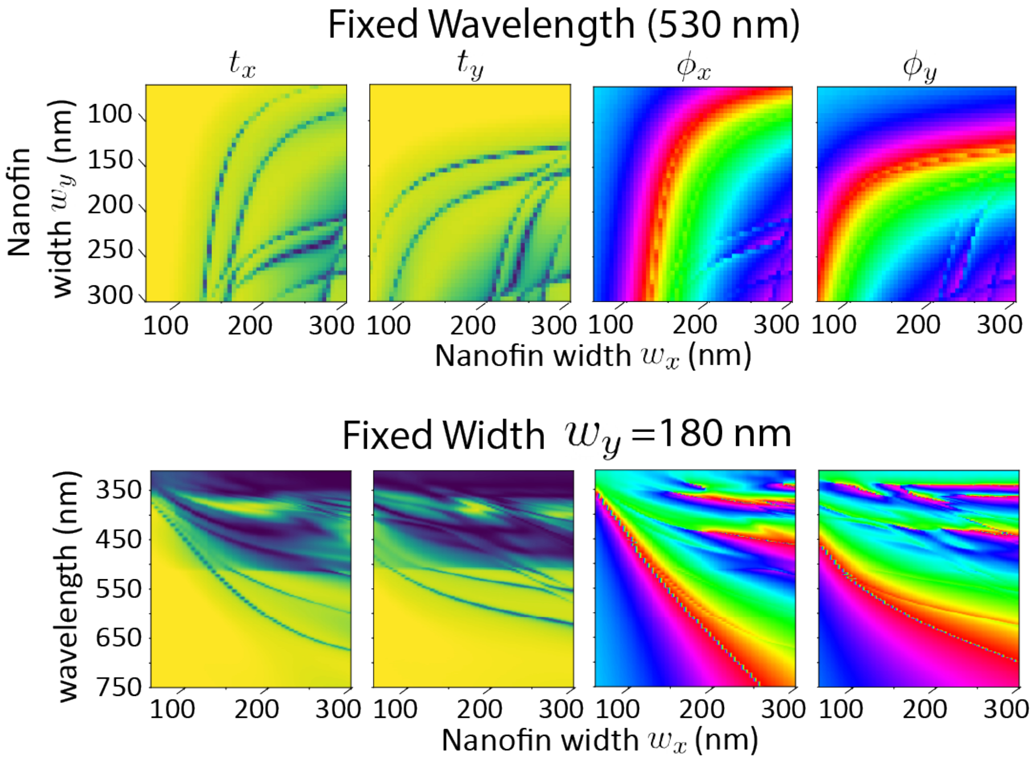
Metalenses can be millimeters to centimeter scale in diameter but must be modeled at a subwavelength scale (~300 nanometers)

Metasurface **Design Theory**



Metasurface **Design Theory**

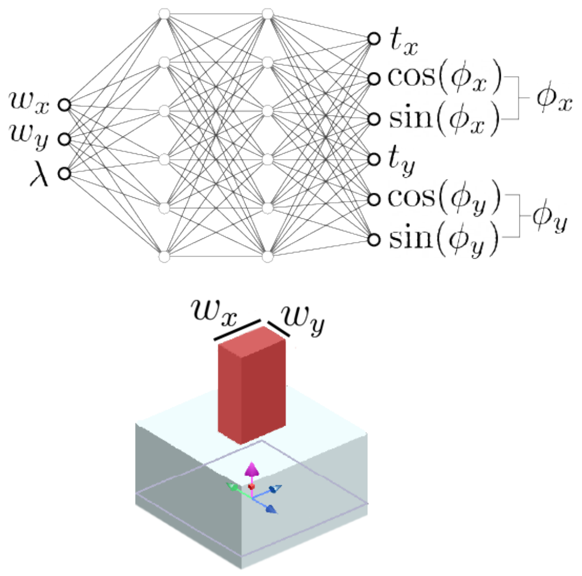


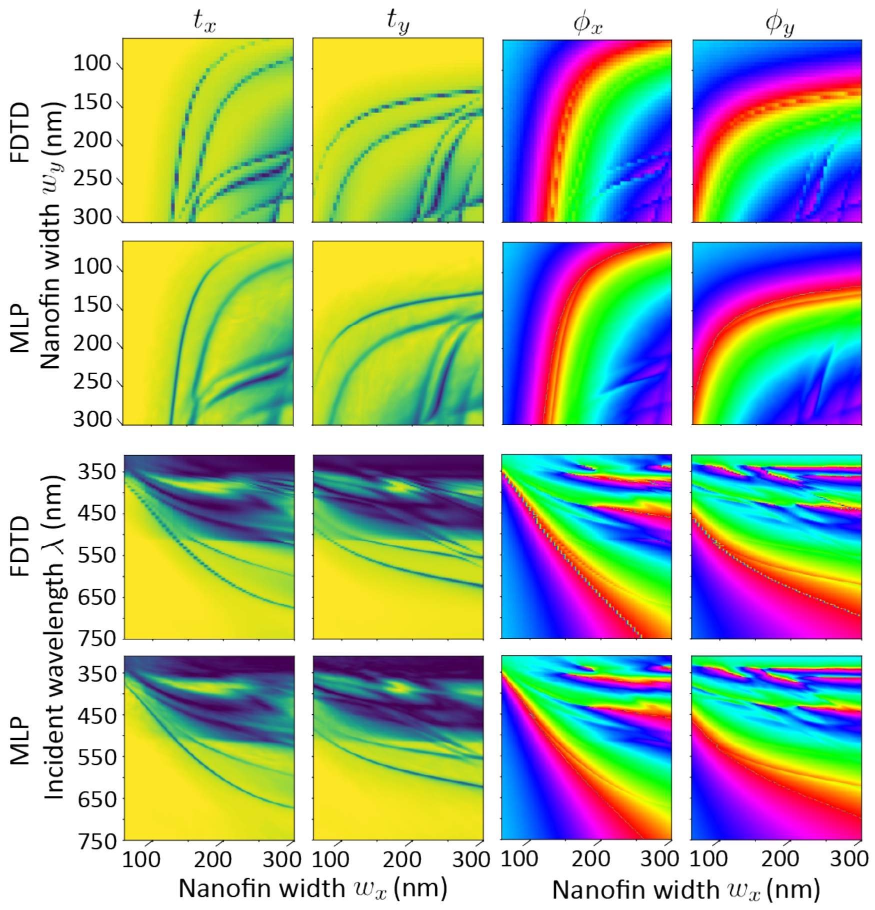


→ (approximation) Meta-atom as building blocks
→ Build dataset by sweeping & solving Maxwell's Equations

Auto-Differentiable **Optical Model**

1. Auto-differentiable field solver (RCWA, FDFD)
2. Neural Representation

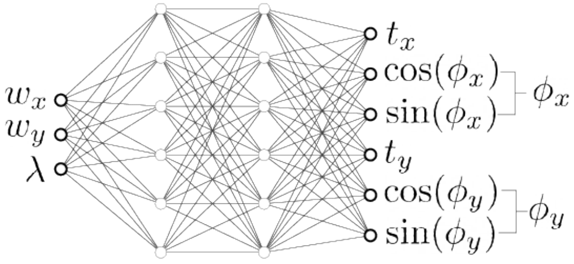


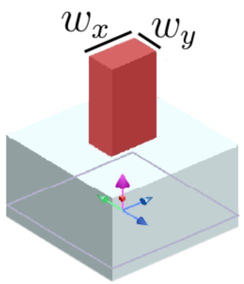


MLP requires a factor of at least $\times 10^5$ fewer FLOPS per cell evaluation vs our auto-differentiable field solver

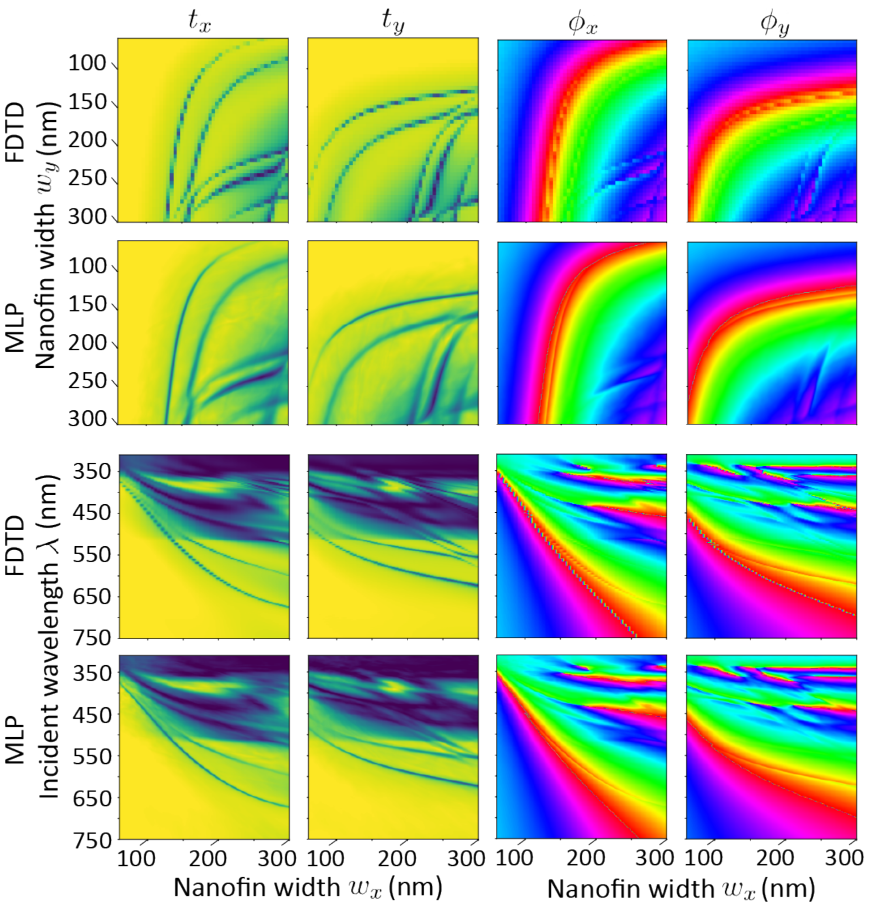
Auto-Differentiable **Optical Model**

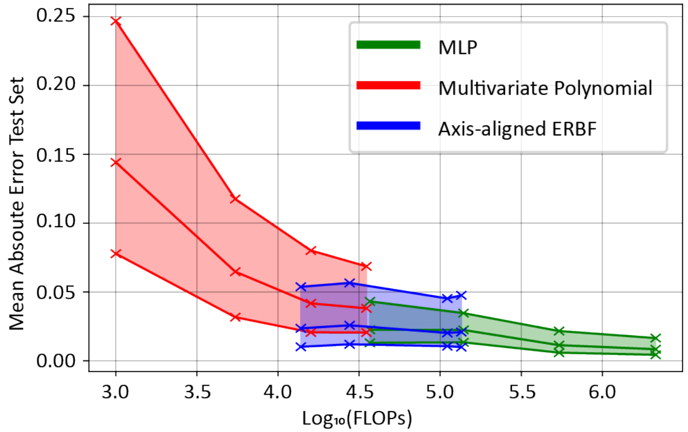
1. Auto-differentiable field solver (RCWA, FDFD)
2. Neural Representation

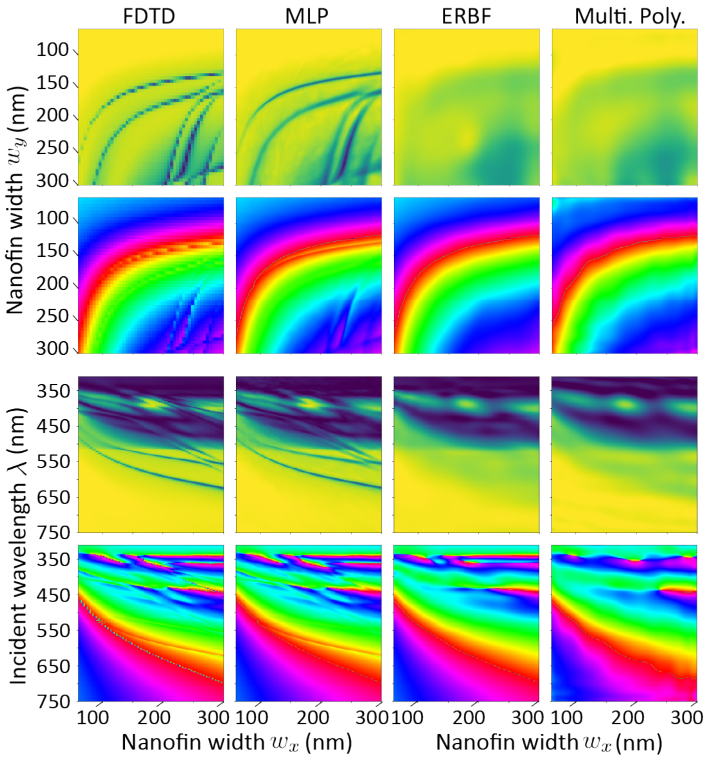




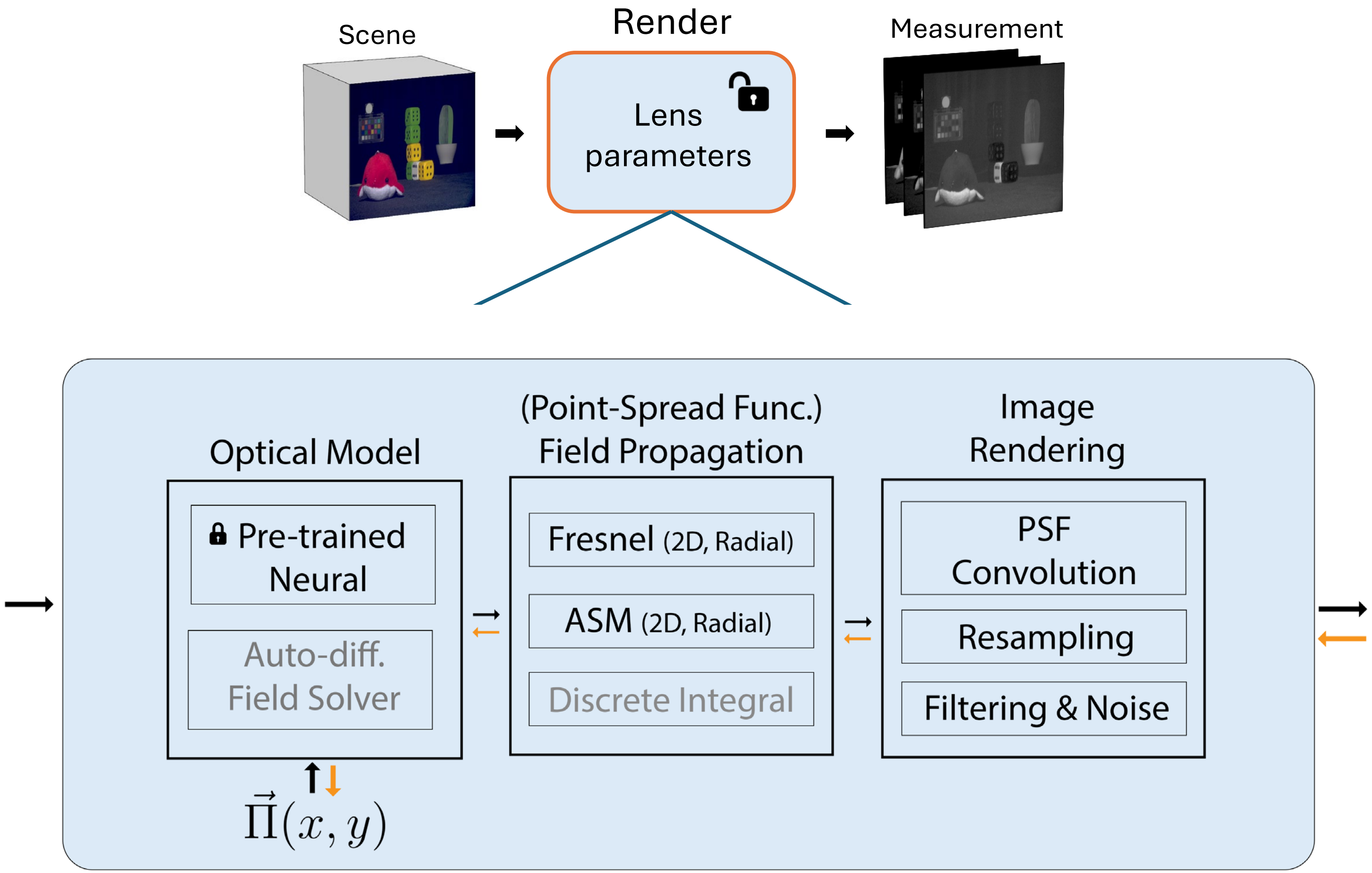
MLP requires a factor of at least x10^5 fewer FLOPS per cell evaluation vs our auto-differentiable field solver



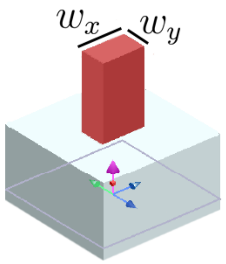


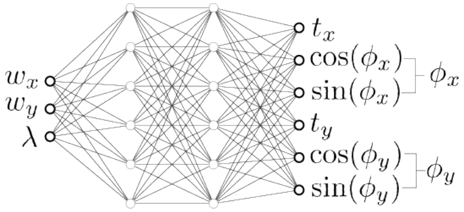


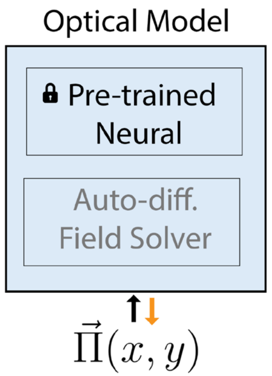
Auto-Differentiable **Optical Model**



Auto-Differentiable **Optical Model**





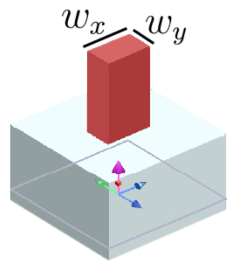


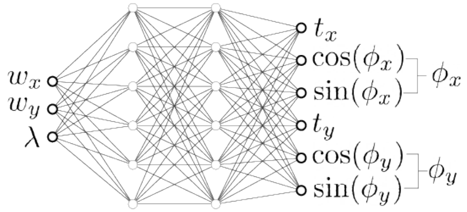
Code designed for versatility and scale
(One of the key value propositions of DFlat):

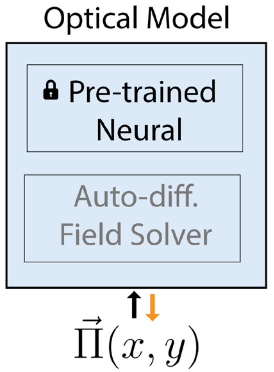
- **<u>Manage large collection of meta-atom datasets</u>**
  - Every shape type, material, block size, requires a new pre-computed dataset

  → Query a dataset by name and automatically download it from server
  → Datasets have a standard form (class) speeding up integration to ML pipeline
  → Integrated field solver to generate new datasets

Auto-Differentiable **Optical Model**







Code designed for versatility and scale
(One of the key value propositions of DFlat):

- **<u>Manage large collection of meta-atom datasets</u>**
  - Every shape type, material, block size, requires a new pre-computed dataset

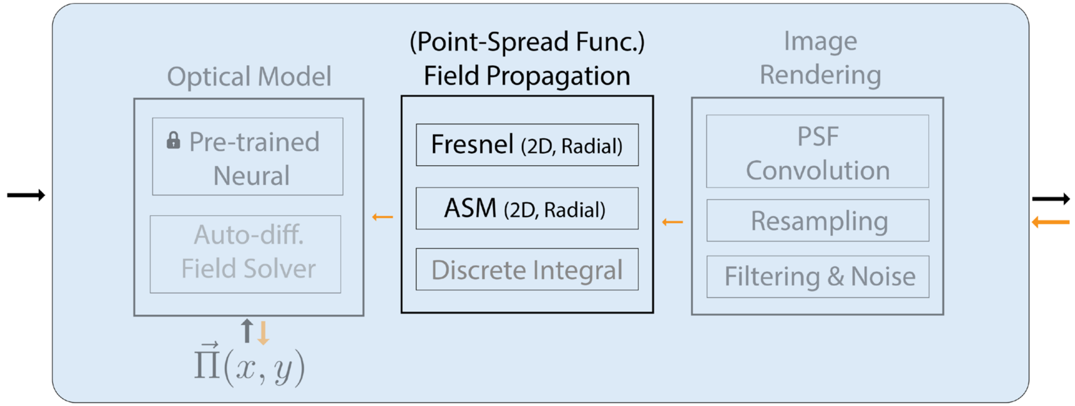
  → Query a dataset by name and automatically download it from server
  → Datasets have a standard form (class) speeding up integration to ML pipeline
  → Integrated field solver to generate new datasets

- **<u>Manage large collection of pretrained neural networks</u>**
  - MLPs will have different input-output dimensions (for different meta-atoms)
  - Different architectures (ex. number layers, attention, etc.)
  - Different pre-processing steps and normalization terms (min/max shapes)

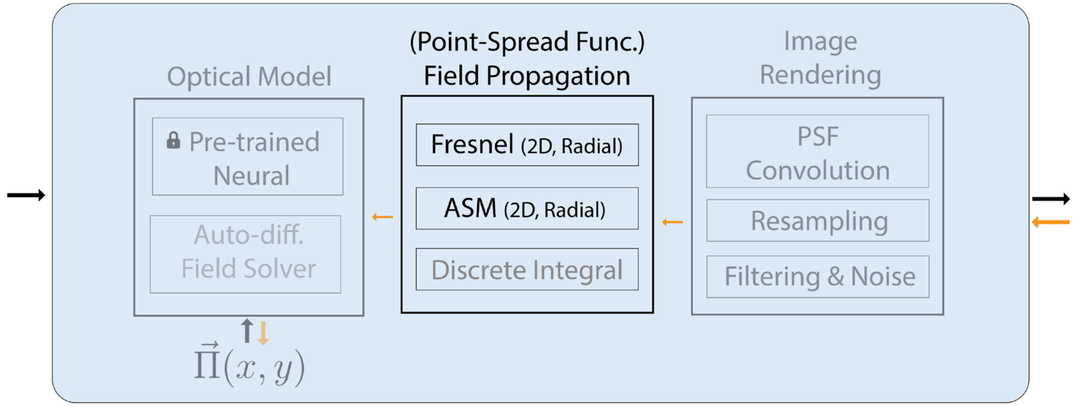
  → Pre-trained models called by name and model weights/configuration files are downloaded from server
  → Models are assembled on the fly according to the configuration file
  → Inherit standardized class/parent so user functionality is the same regardless of the meta-atom/dataset choice

Auto-Differentiable **Optical Model**

Auto-Differentiable **Field Propagation**



Auto-Differentiable **Field Propagation**



Different numerical implementations of field propagation:

- **Fresnel Method:** x1 Fourier/Hankel transform (approximate)
- **Angular Spectral Method:** x2 Fourier/Hankel transforms (exact)
- **Discrete Integration:** Pixel space transformation (exp., exact)

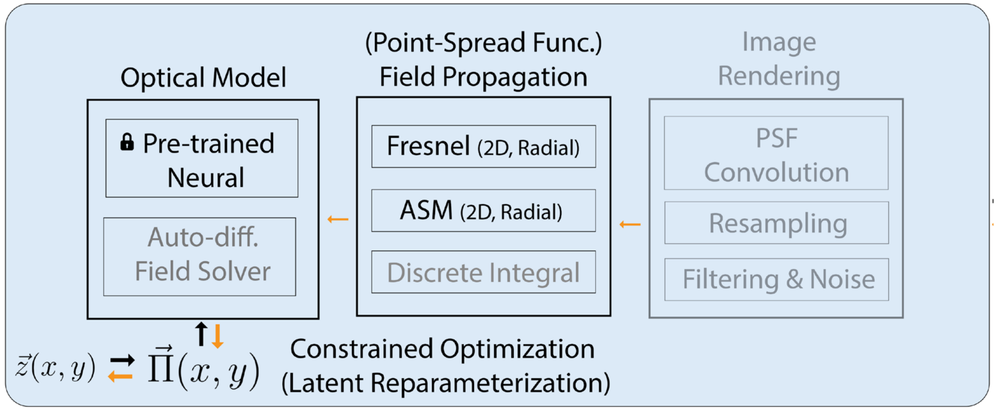
Desirable implementation involves many steps (method dependent)

- Up-sample user provided profile according to few conditions
- Zero-padding determines output field discretization (λ, z)-dependent
- Resample output field to a pixel size grid if sub-pixel sampling

→ There are many scenarios where one method is more efficient than the other

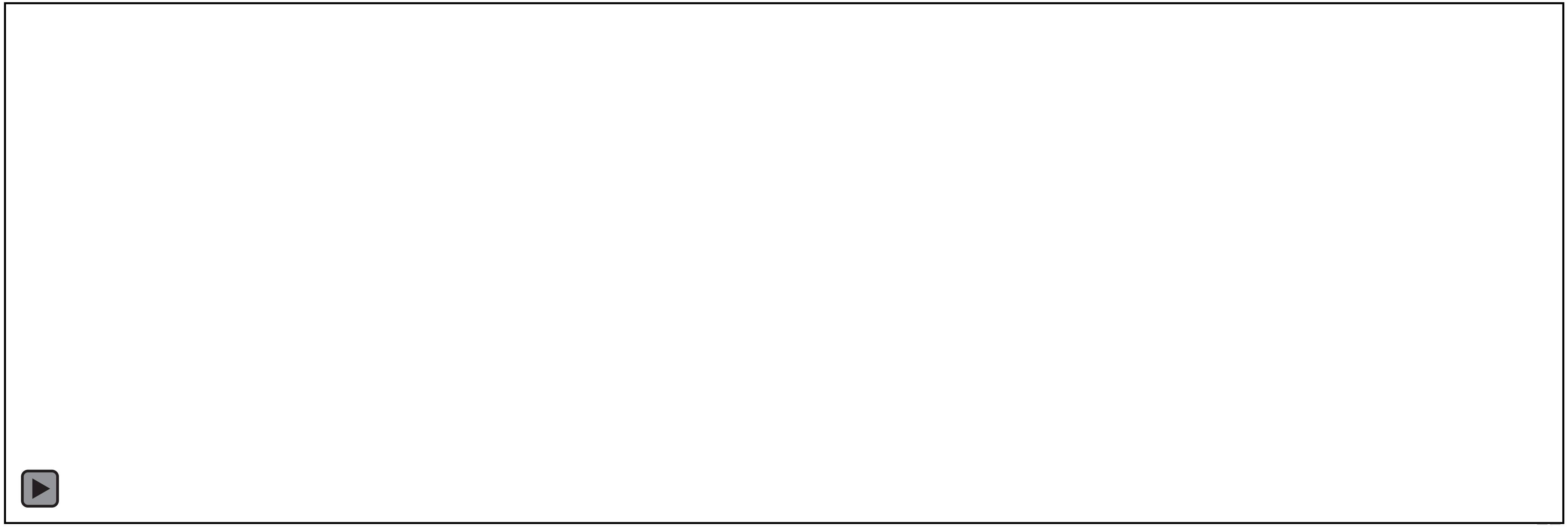
Operation in computational imaging research that we wanted to standardized and provide for others

Point-Spread Function Optimization



Optimize the shapes on a metasurface to produces a simple multi-focci point-spread function at the sensor plane

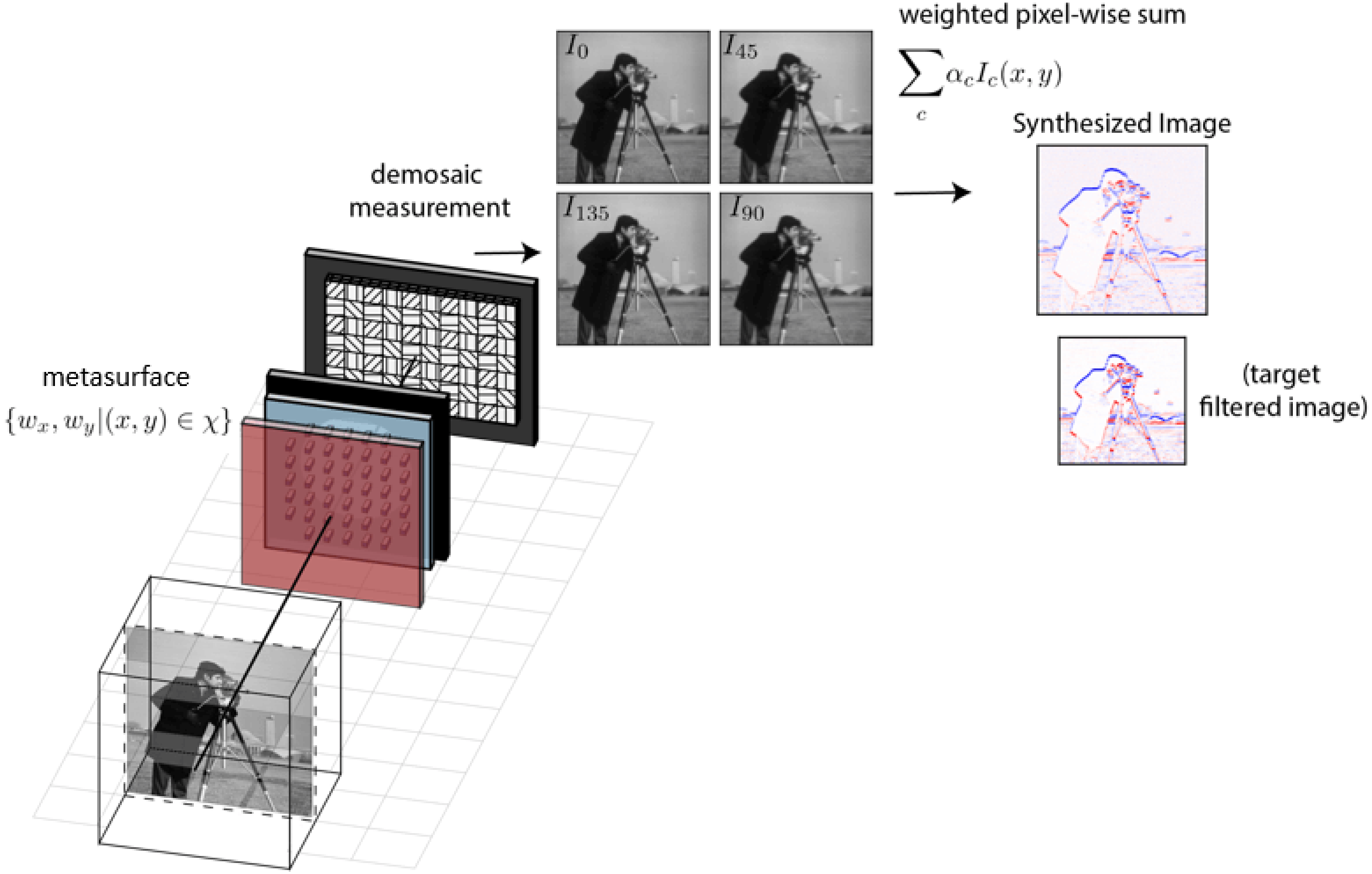
→ Exists analytic solution for the ideal phase and transmission profile to produce each focal lobe alone but not obvious how to optimally merge the many behaviors into one lens.



Task-Specialized Vision: Optical Computing

Replace digital image processing (e.g. edge-detection) with cheaper opto-electronic operations

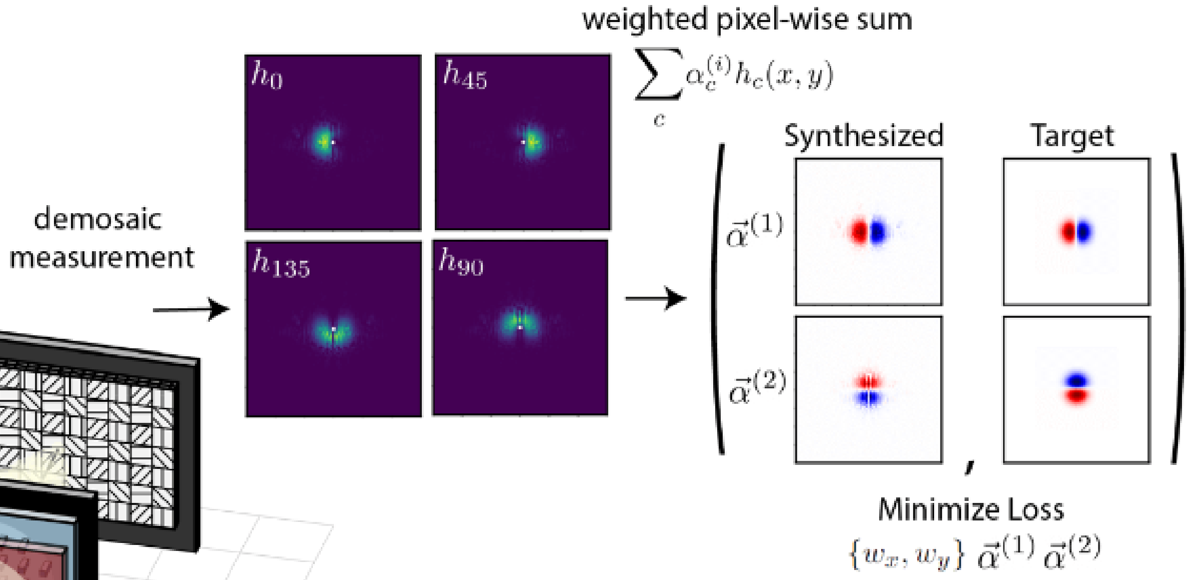
(More info)

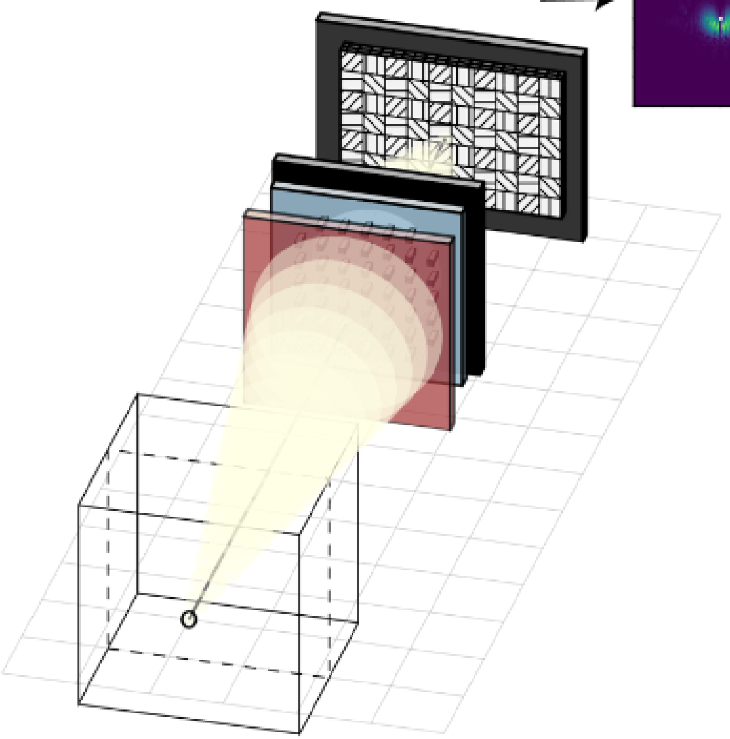


Task-Specialized Vision: Optical Computing

Replace digital image processing (e.g. edge-detection) with cheaper opto-electronic operations

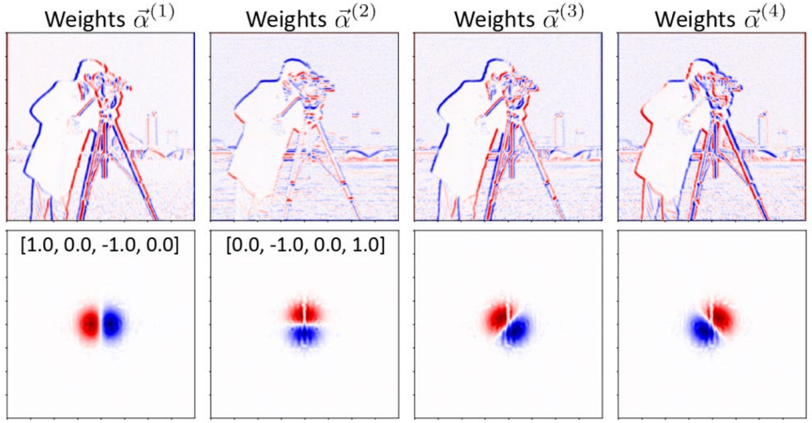
(More info)





Steerable gaussian derivative by changing the digital summation weights

→ First time that opto-electronic processing this was done in a single snapshot with a single sensor



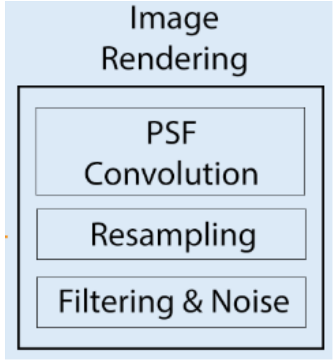
Point-Spread Function Optimization

Dflat Goal: Make the optimization of flat optics as easy as optimizing a deep neural network (code perspective)

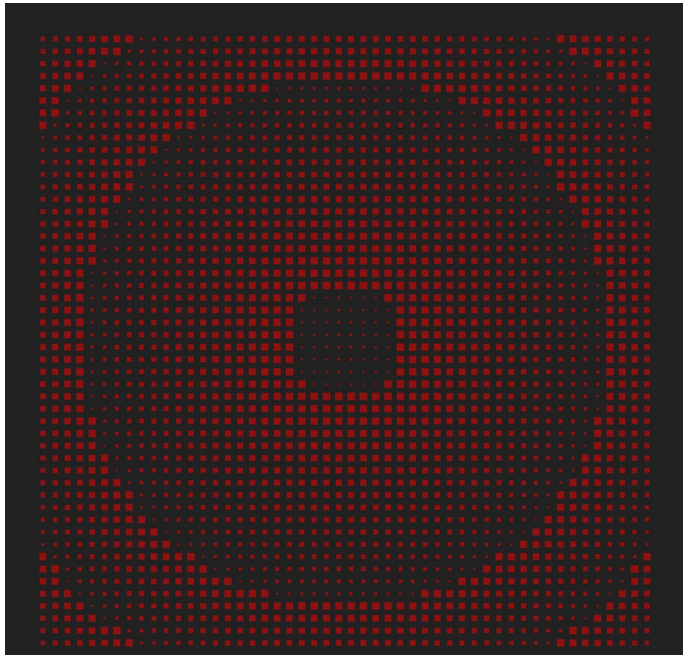
→ Using pre-trained **metasurface models should be as easy as using a pre-trained CLIP or Perceptual Loss**
Challenge: Each metasurface model is like a CLIP trained in a different language (different tokenizers, datasets…)

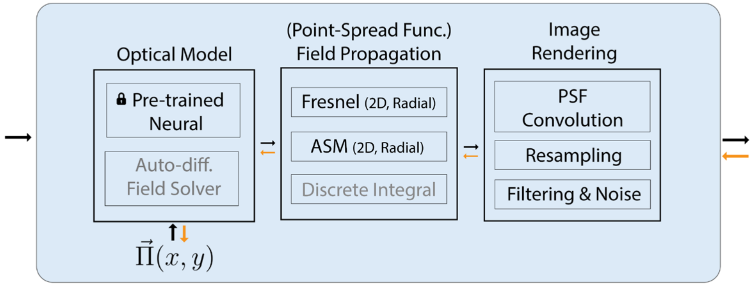
→ Modules for **point-spread functions/propagations should be as easy as using Conv2D layer**
Challenge: Managing large number of branches in code for different approximations and configurations

Additional piece not discussed as much in this talk:



D-Flat





Free and Open Source:

- Continuous Integration / Pytest Workflow
- Complete docstrings (missing new readthedocs)
- Available on Python Package Index (PyPi) as “dflat_opt”
- *Nightly* versions on Github

(Github)

A lot of open computational projects for all skill levels:
(email @ dhazineh@g.harvard.edu or give it a try)
(Contact Todd Zickler for CS / Federico Capasso on Fabrication)

- Differentiable field solvers
- Large area topology optimization for high-dimensional shapes
- *Memory efficient propagation (bottleneck)
- Co-Optimization with CNNs
- Rendering without shift-invariant PSF assumption
- Gradient checkpointing and other code improvements

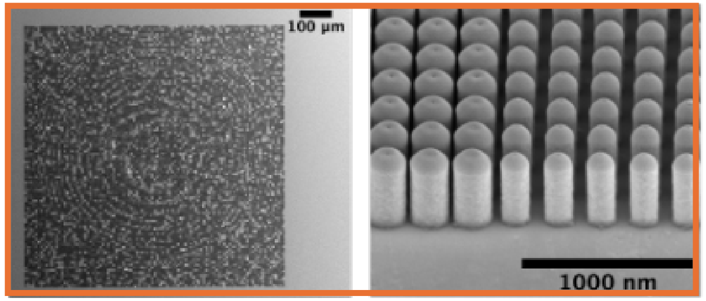
Extra

Auto-Differentiable **Field Propagation**

Validation of implementations against experiment

Different numerical implementations of field propagation:

- **Fresnel Method:** x1 Fourier/Hankel transform (approximate)
- **Angular Spectral Method:** x2 Fourier/Hankel transforms (exact)



Experimental data from Dr. Daniel Lim, Capasso Group