Designing Cameras to **Detect the Invisible**: Imaging and Vision in Harsh Conditions

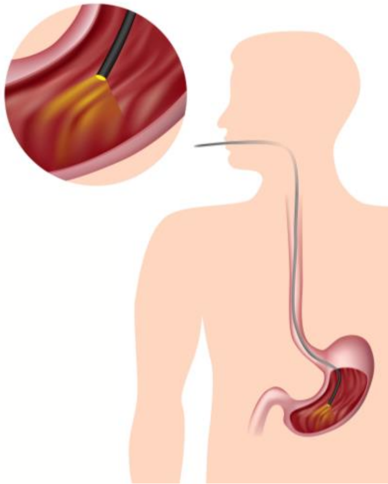
Felix Heide

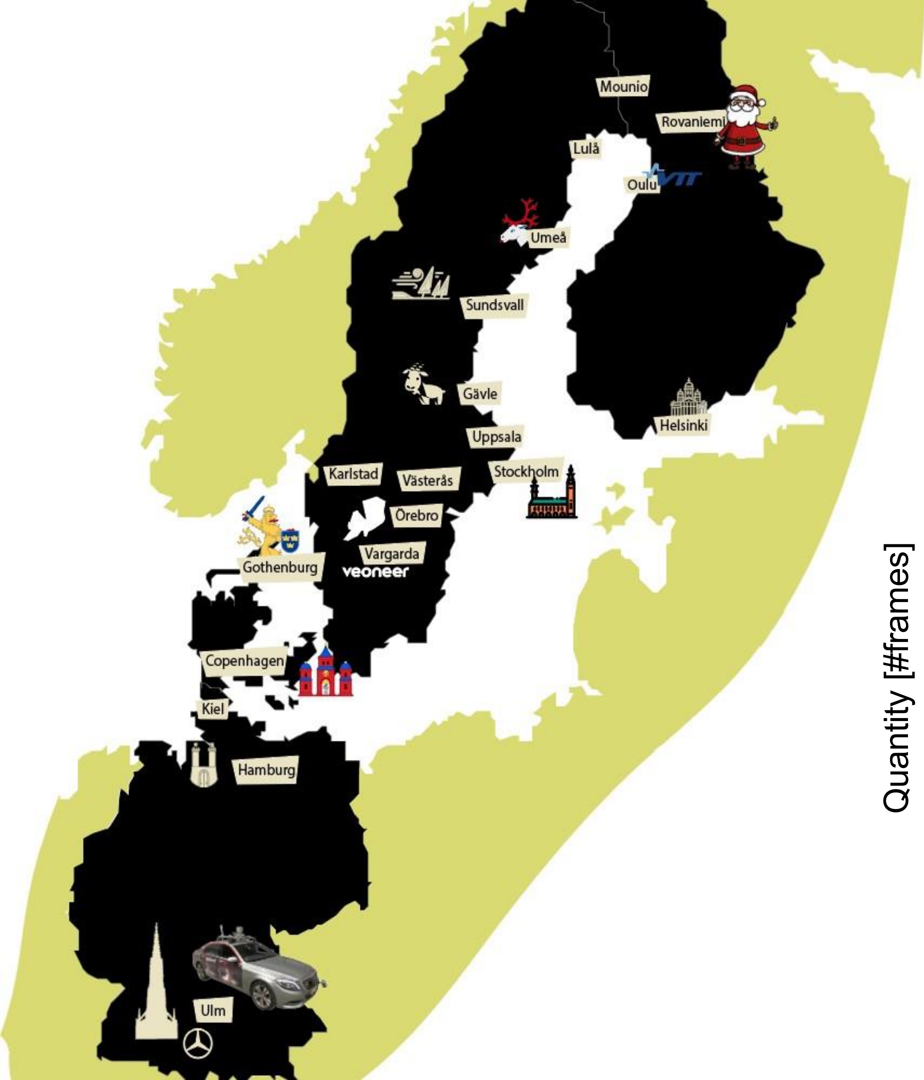
PRINCETON UNIVERSITY

algolux

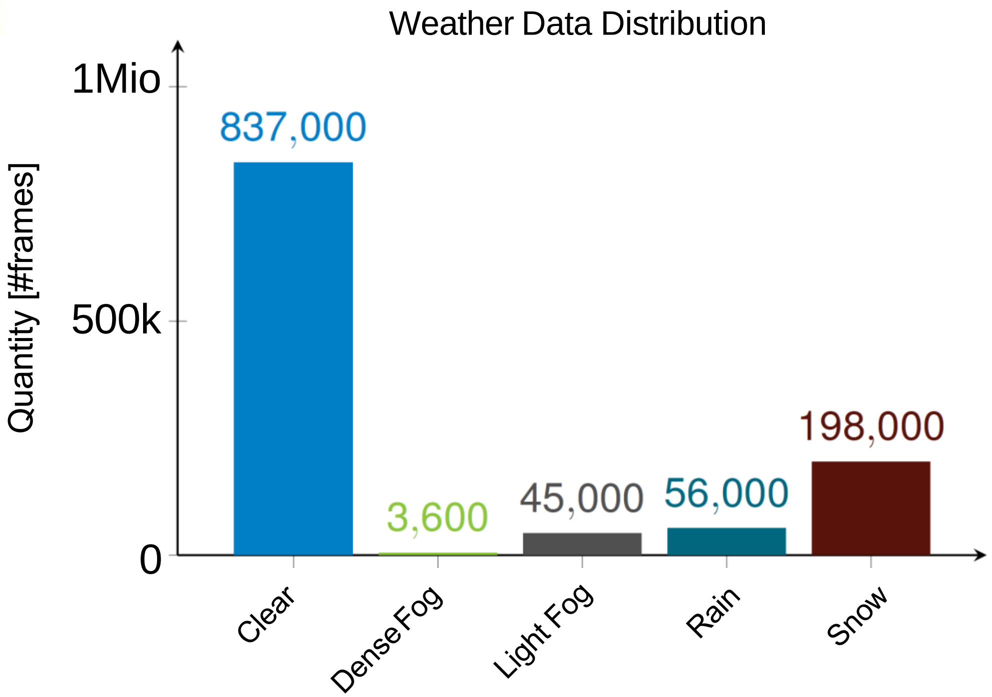
light.princeton.edu

Imaging and Vision are ubiquitous



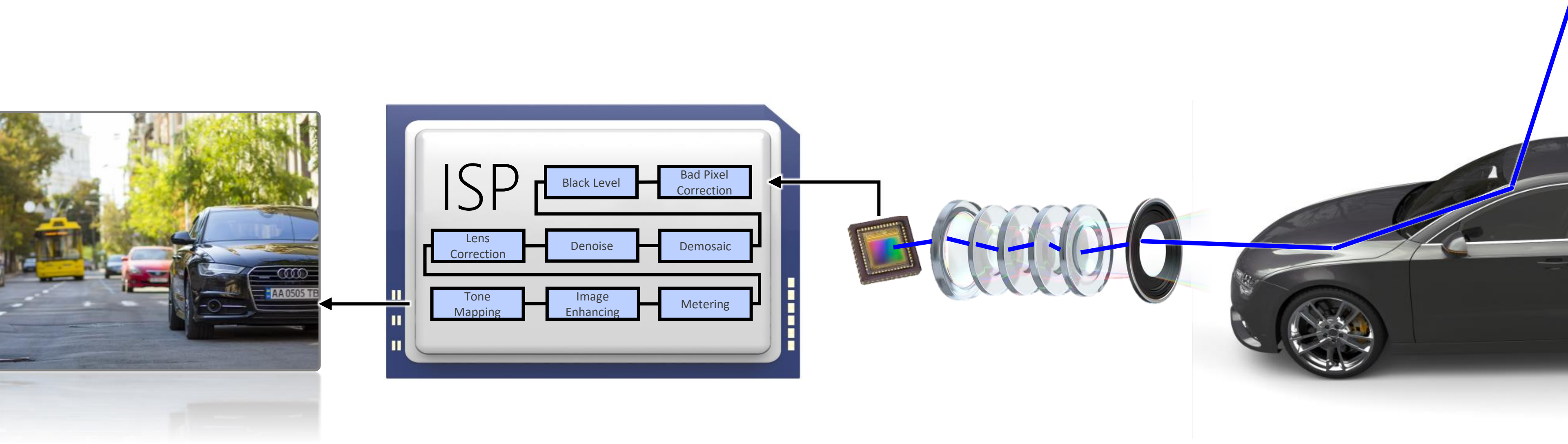


... we **can't stick just to supervision** to achieve robust vision.

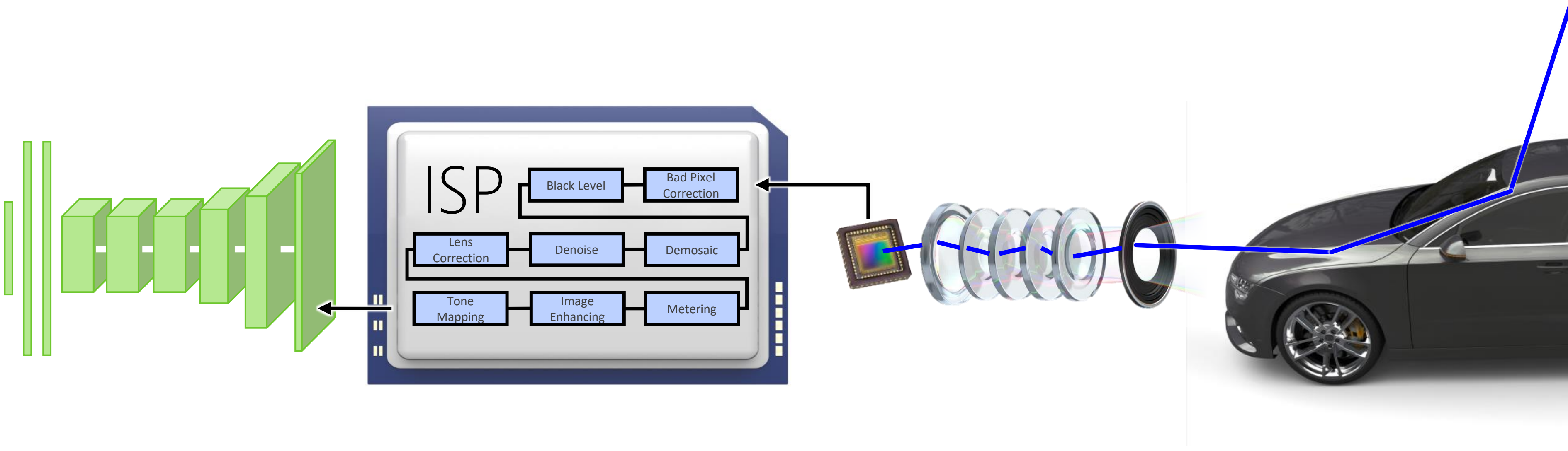


… we **can't stick just to supervision** to achieve robust vision.

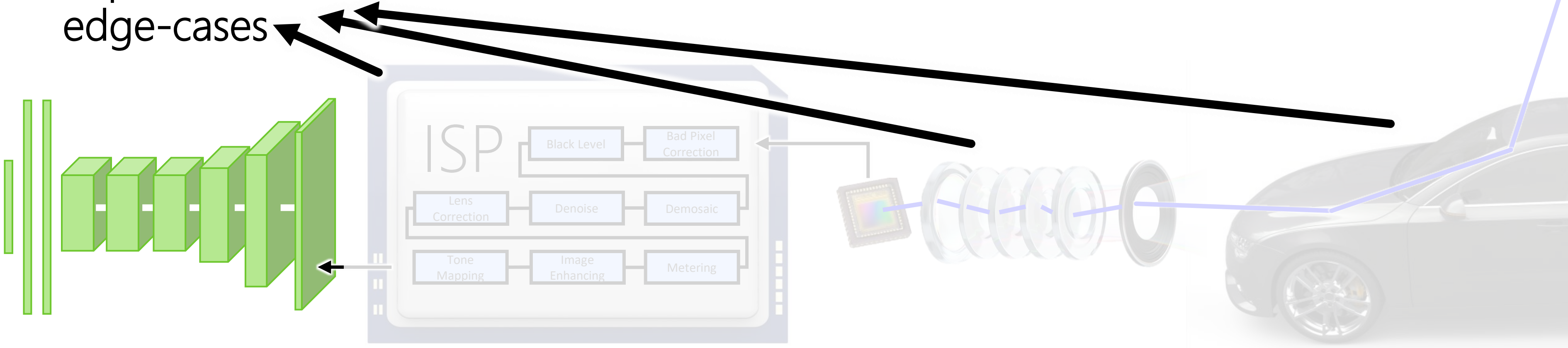
FRCNN Resnet 50



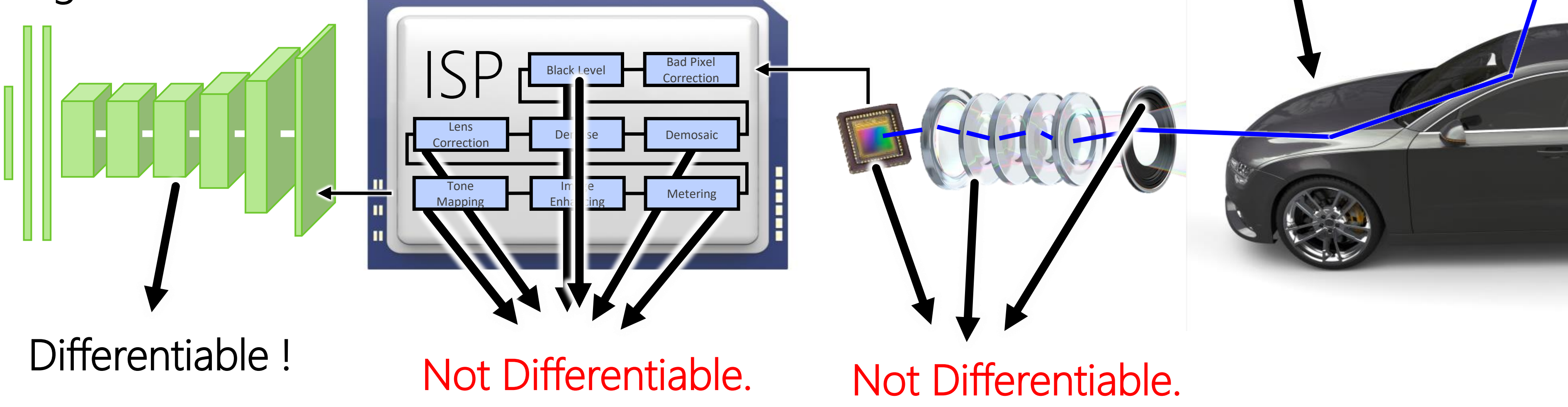
ISP
Black Level
Bad Pixel Correction
Lens Correction
Denoise
Demosaic
Tone Mapping
Image Enhancing
Metering
AA 0505 TB


ISP
Black Level
Bad Pixel Correction
Lens Correction
Denoise
Demosaic
Tone Mapping
Image Enhancing
Metering

Supervise to handle edge-cases

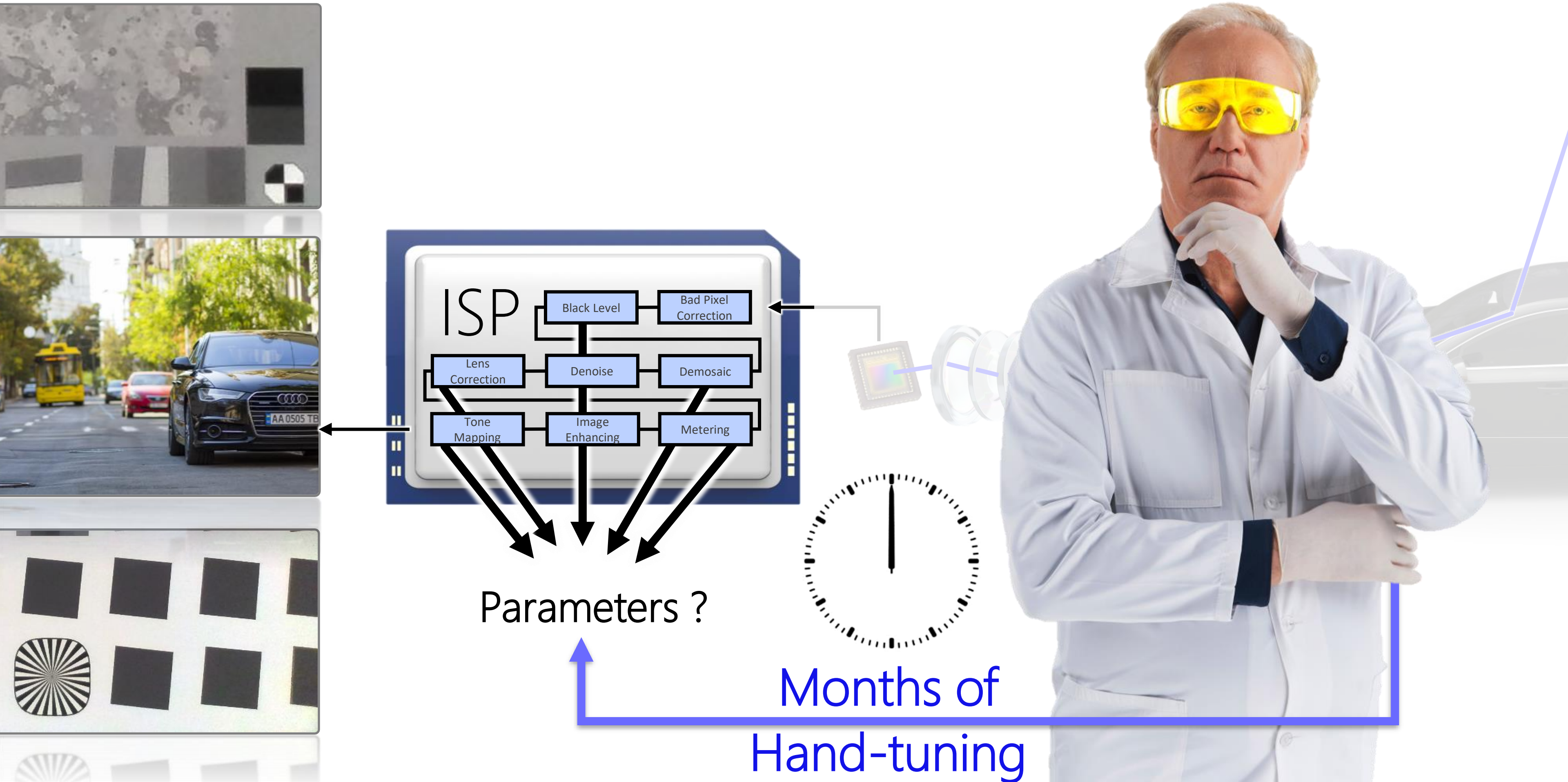


Best practice is to supervise to handle edge-cases



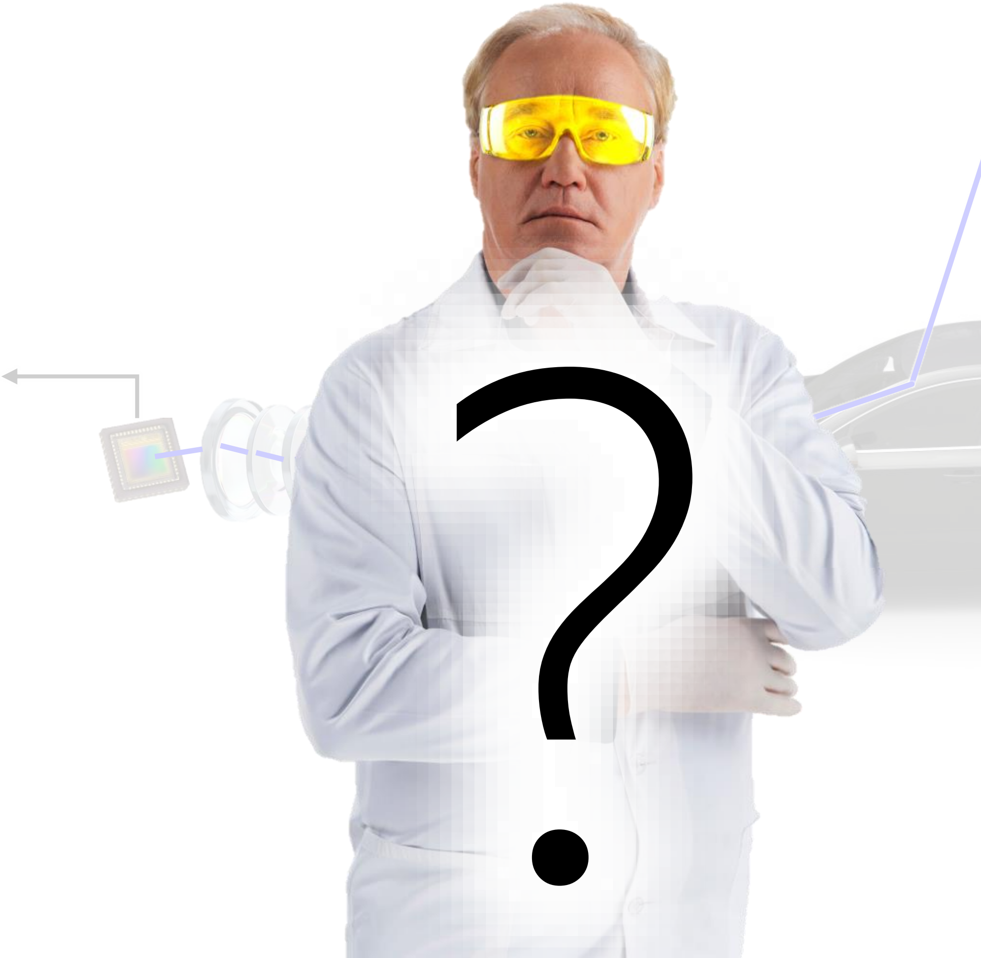
The "Golden Eye" Expert

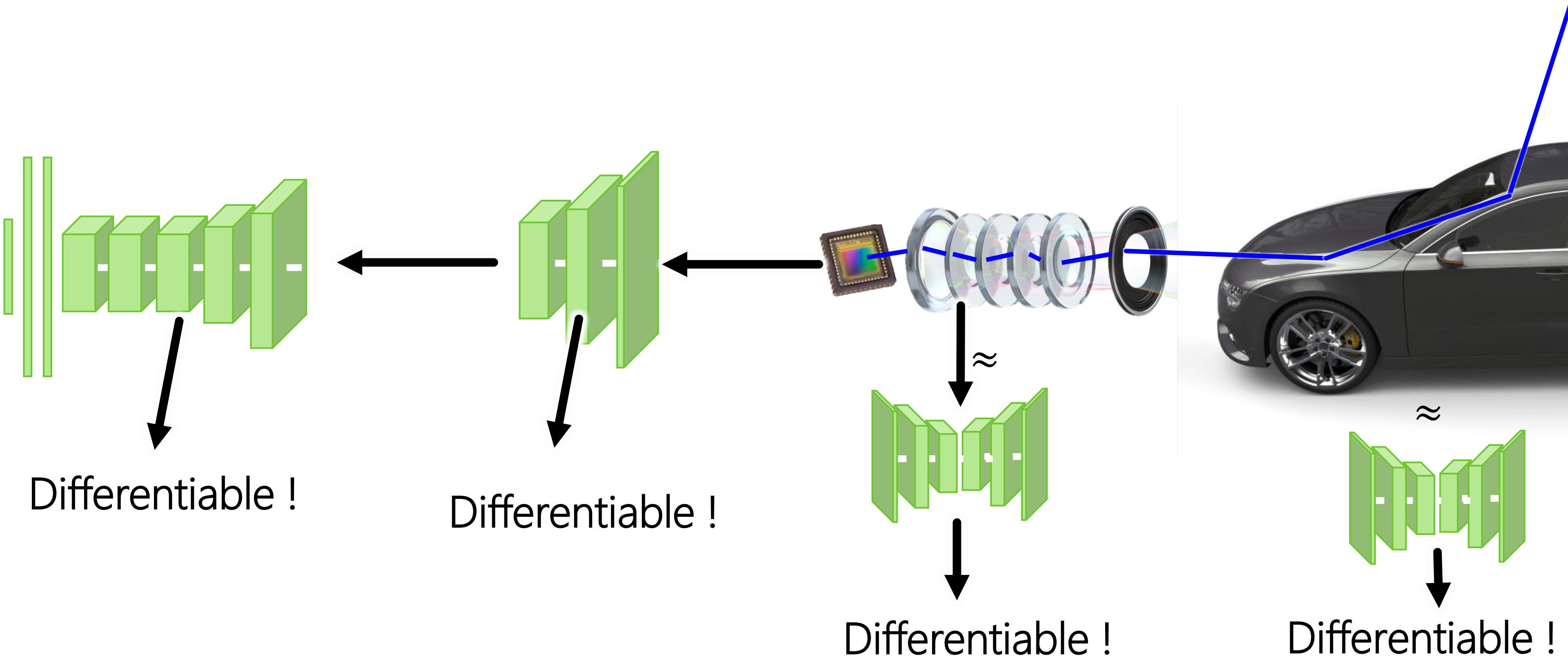
"Golden Eye"



Tune ISP for Object Detection

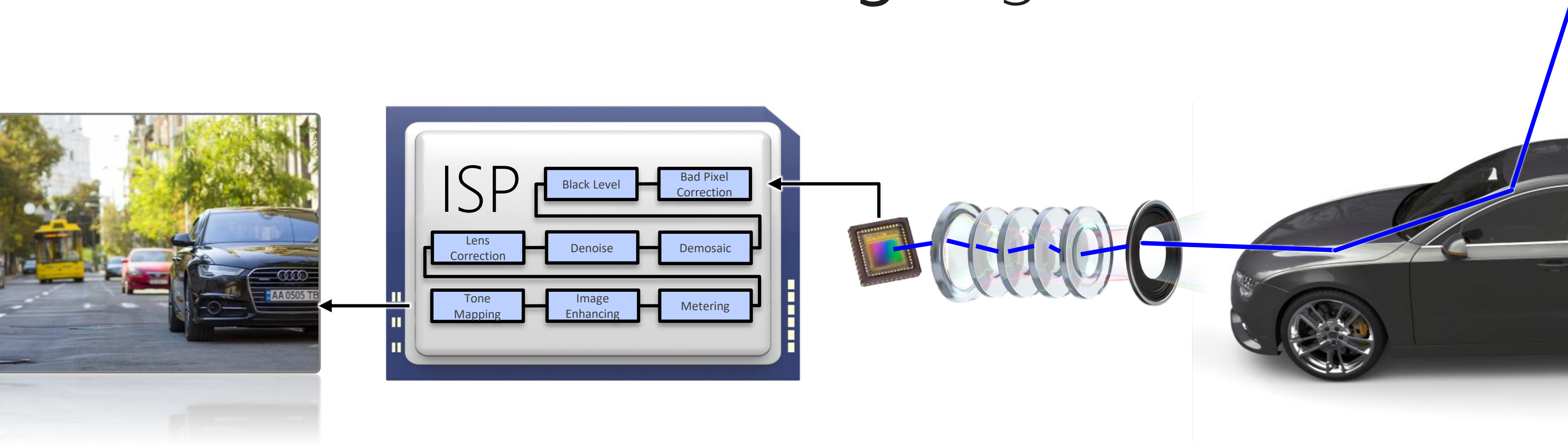
“Golden Eye”



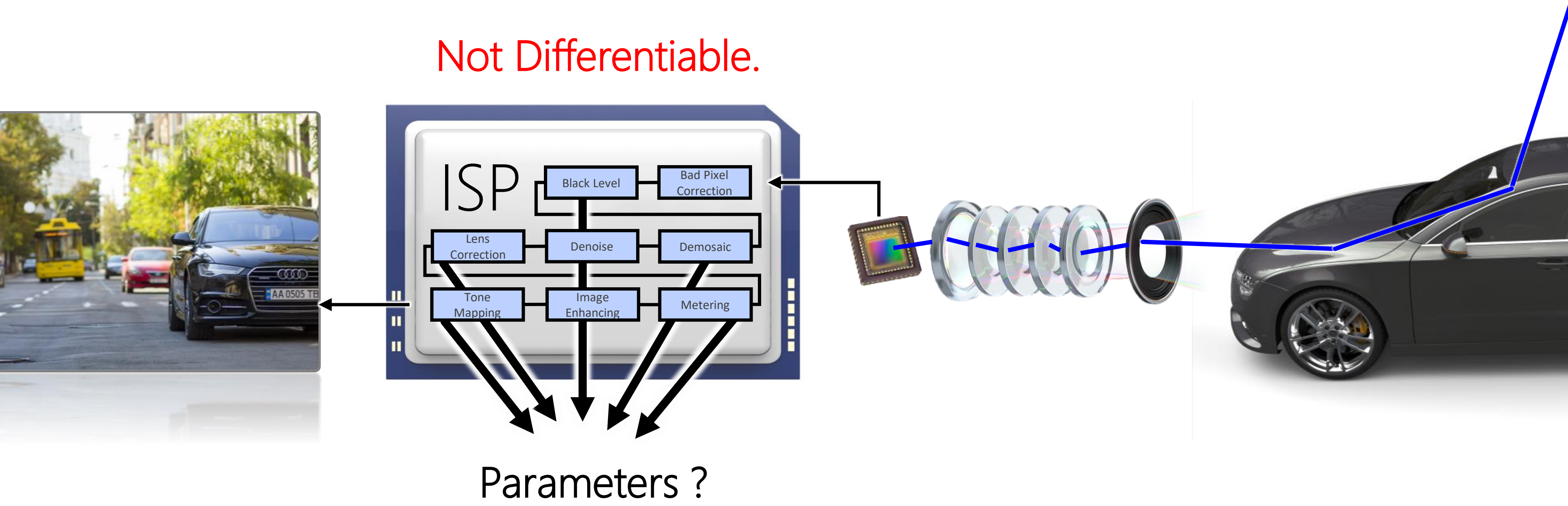

≈
≈
Differentiable !
Differentiable !
Differentiable !
Differentiable !

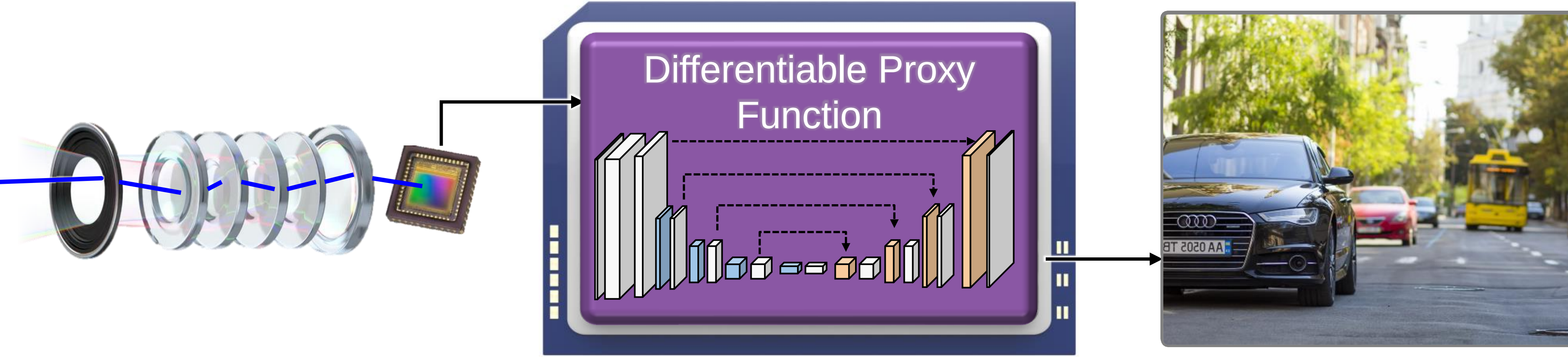
End-to-End Models for Edge-Cases

Instead Of Labeling Edge-Cases



Typical Imaging Stack

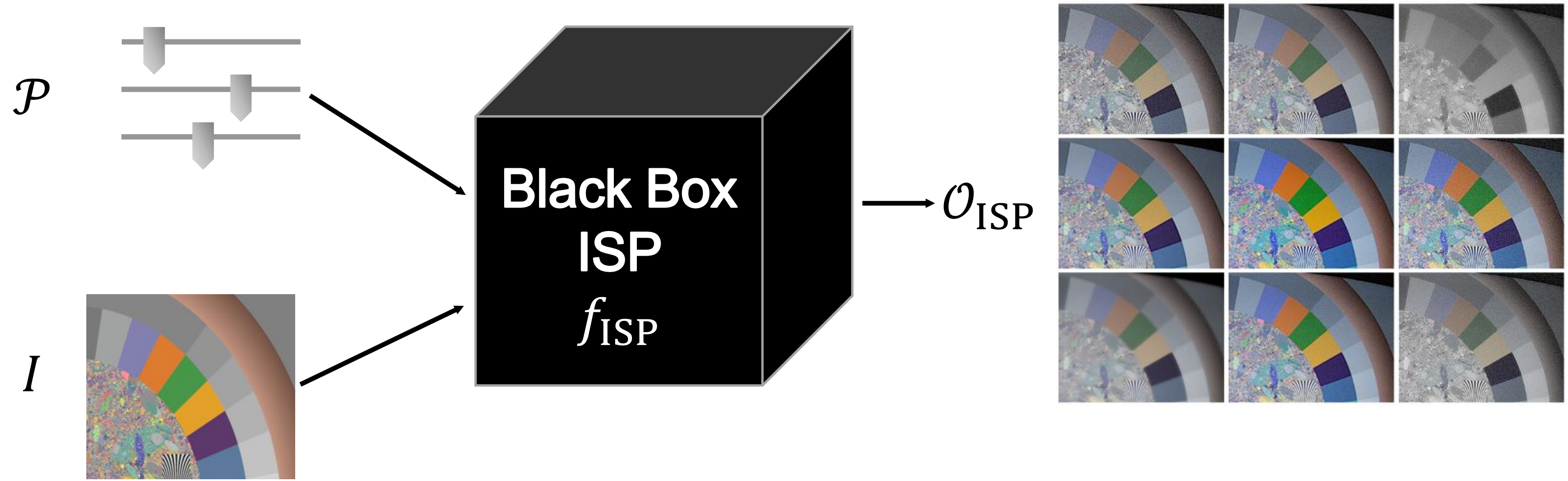




Stage 1: Learning the Differentiable Proxy Function

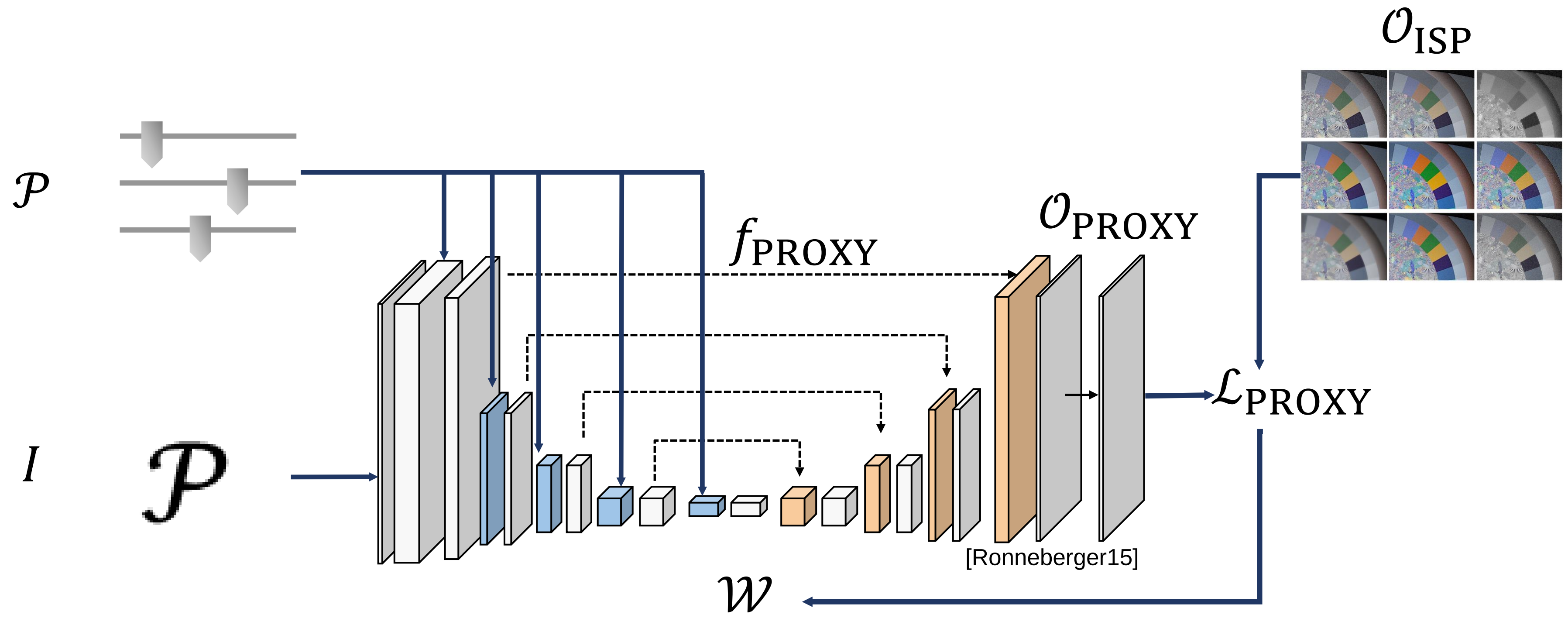
Stage 2: Optimizing Hyperparameters for Task-Specific Outputs

Stage 1: Learning the Differentiable Proxy Function



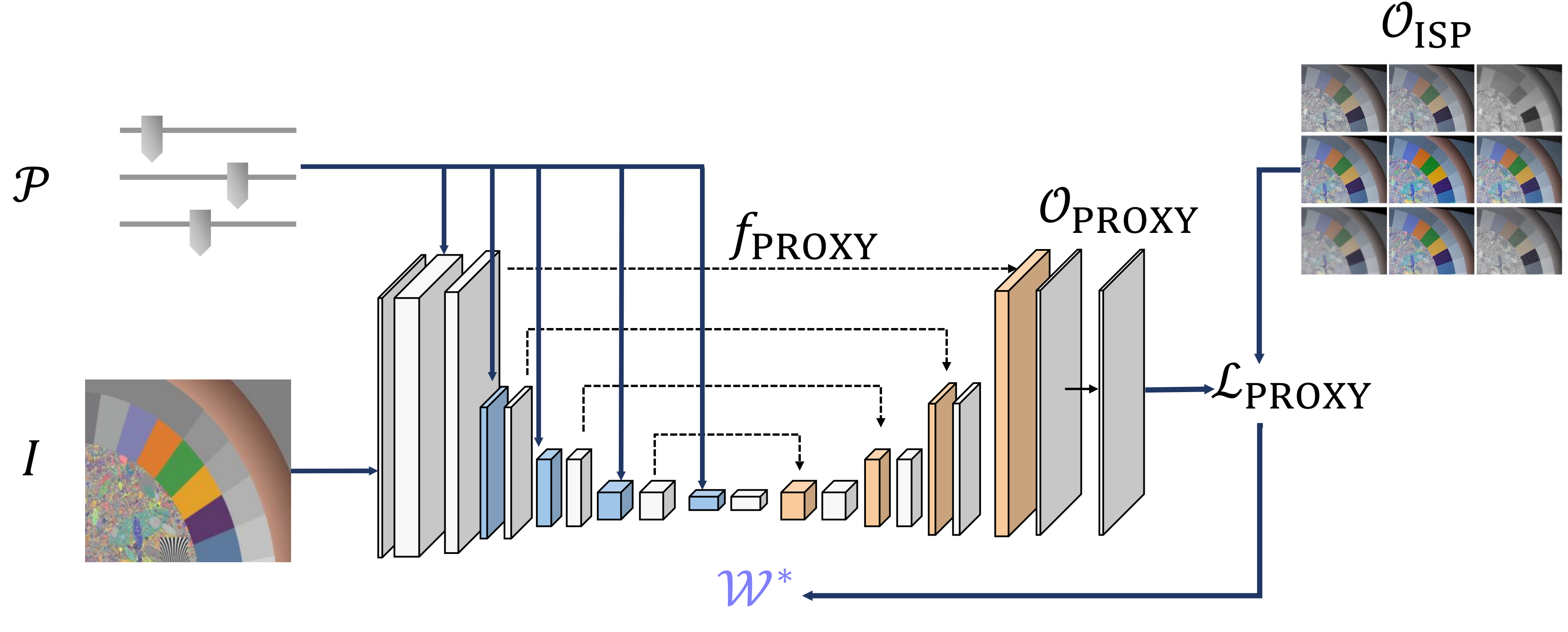
$$\mathcal{O}_{\mathrm{ISP}} = f_{\mathrm{ISP}}(\, I \,, \mathcal{P}\,)$$

Stage 1: Learning the Differentiable Proxy Function

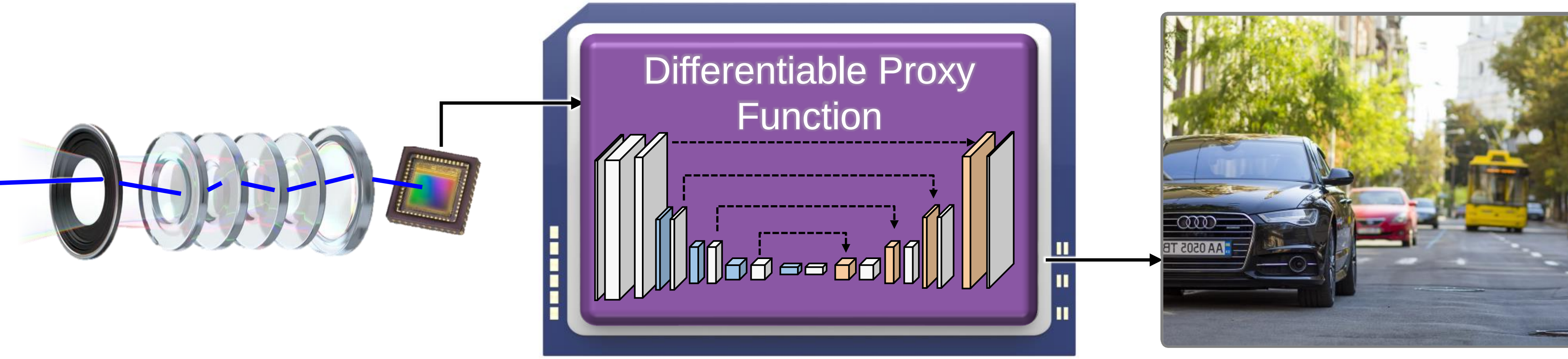


$$\mathcal{O}_{\text{PROXY}} = f_{\text{PROXY}}(I, \mathcal{P}, \mathcal{W})$$

Stage 1: Learning the Differentiable Proxy Function



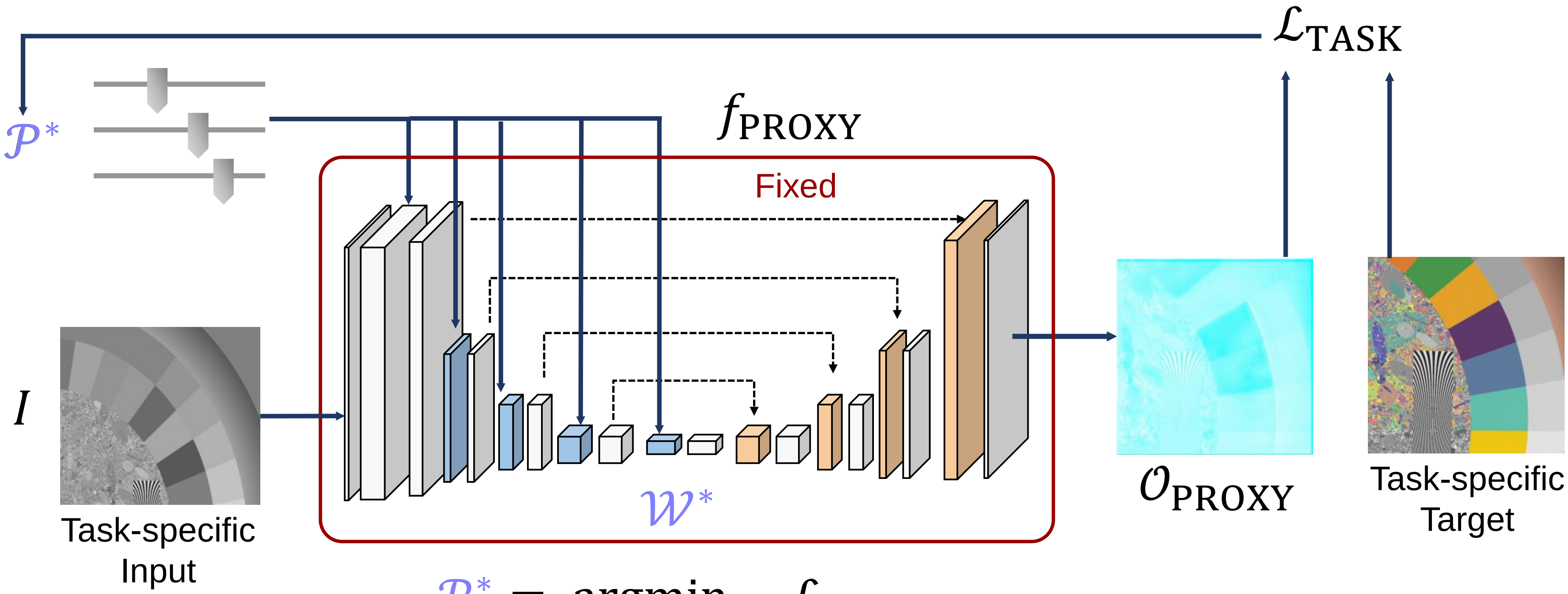
$$f_{\mathrm{PROXY}}(I, \mathcal{P}, \mathcal{W}^*) \approx f_{\mathrm{ISP}}(I, \mathcal{P})$$



Stage 1: Learning the Differentiable Proxy Function

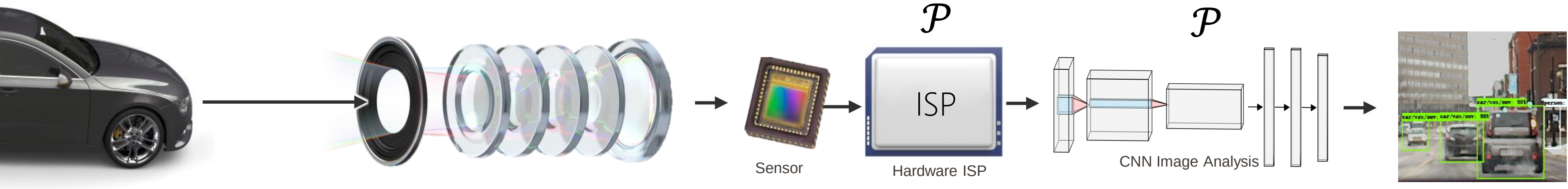
Stage 2: Optimizing Hyperparameters for Task-Specific Outputs

Stage 2: Optimizing Hyperparameters

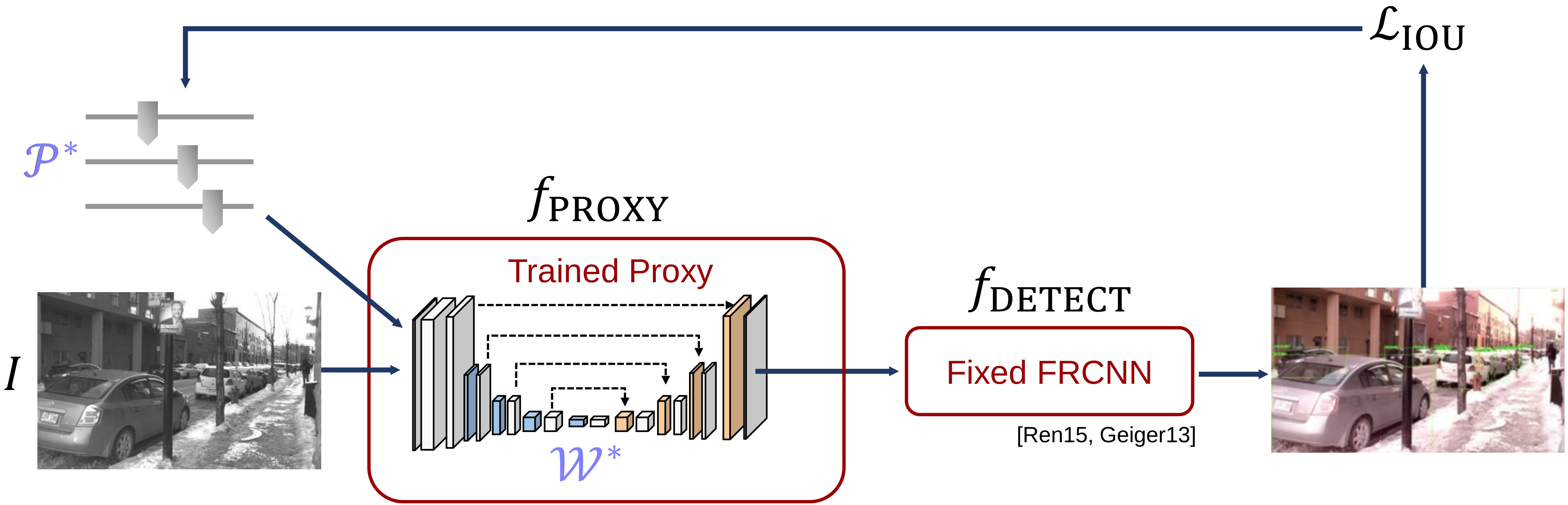


$$\mathcal{P}^* = \operatorname{argmin}_{\mathcal{P}} \mathcal{L}_{\text{TASK}}$$

Joint Optimization of
Hardware Image Processing & Detection

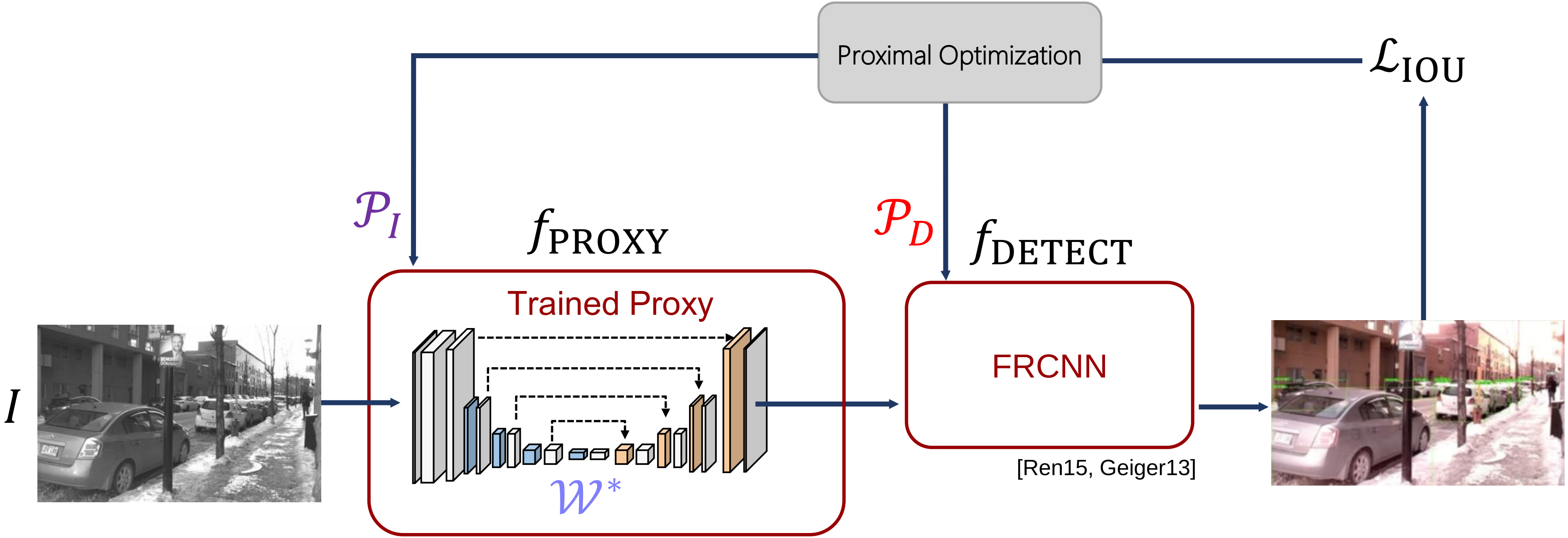


Domain-specific ISP Fine-Tuning



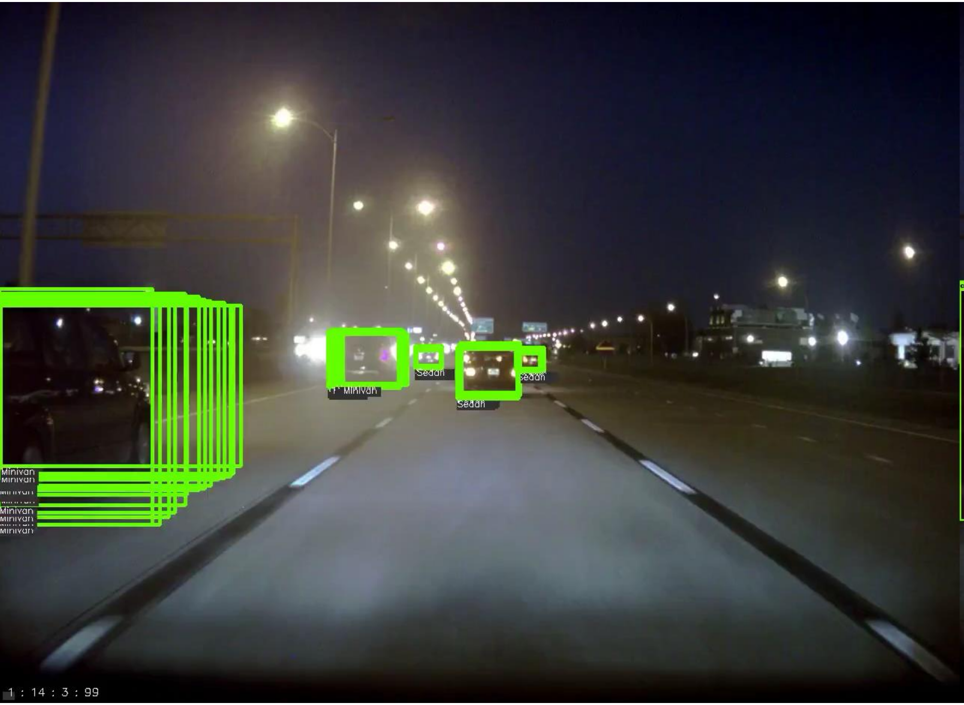
$$\mathcal{P}^* = \operatorname{argmin}_{\mathcal{P}} \mathcal{L}_{\mathrm{IOU}}(f_{\mathrm{DETECT}}(f_{\mathrm{PROXY}}(I, \mathcal{P}, \mathcal{W}^*)))$$

End-to-End Composite Proximal Optimization

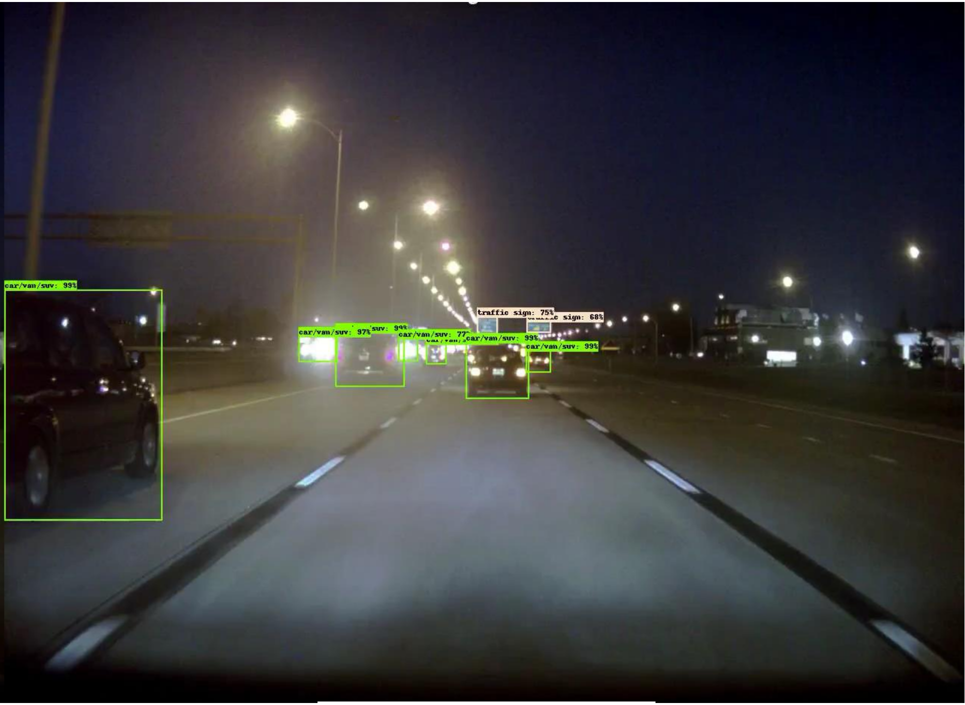


$$\mathcal{P}^* = \operatorname{argmin}_{\mathcal{P}} \; \mathcal{L}_{\text{IOU}}\big(f_{\text{DETECT}}\,(f_{\text{PROXY}}(I, \mathcal{P}_I, \mathcal{W}^*), \mathcal{P}_D)$$

Object Detection Result vs. Tesla Autopilot

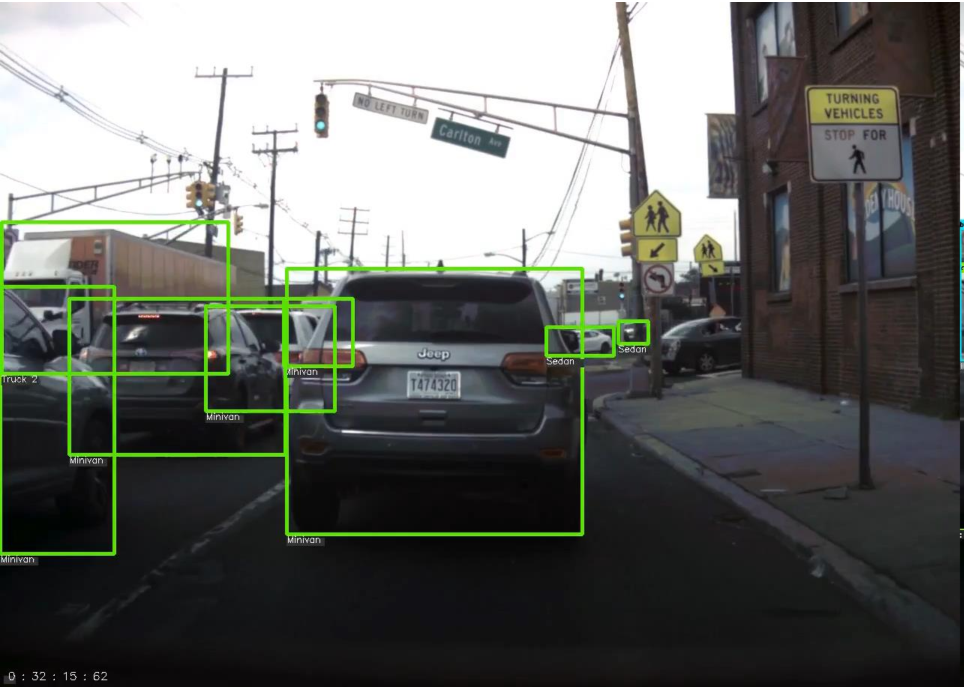


Tesla Autopilot
(Camera + Radar)

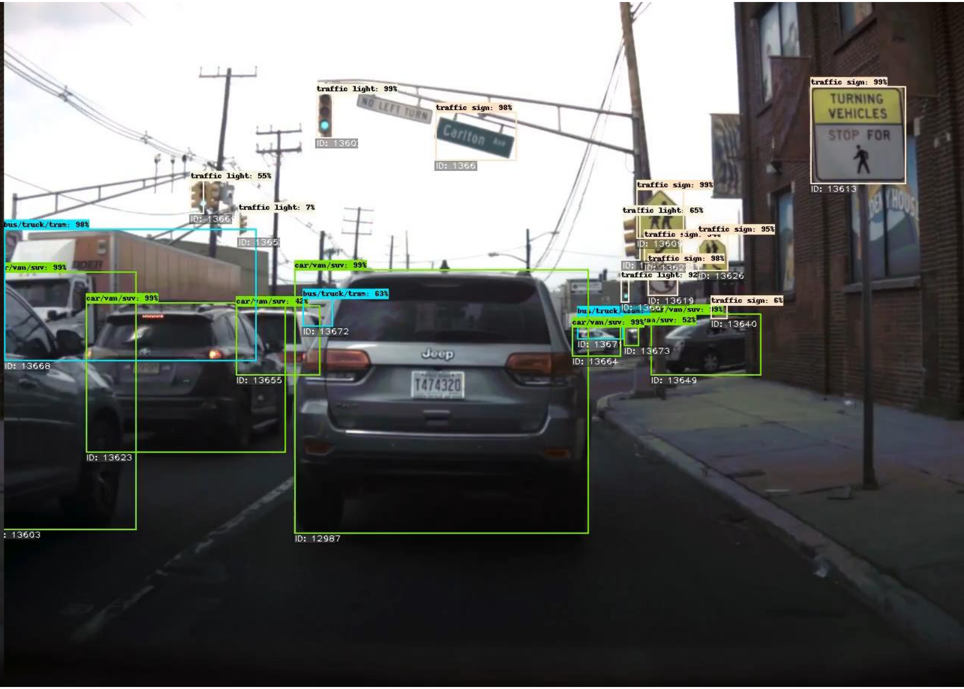


Proposed
(Camera-only)

Object Detection Result vs. Tesla Autopilot

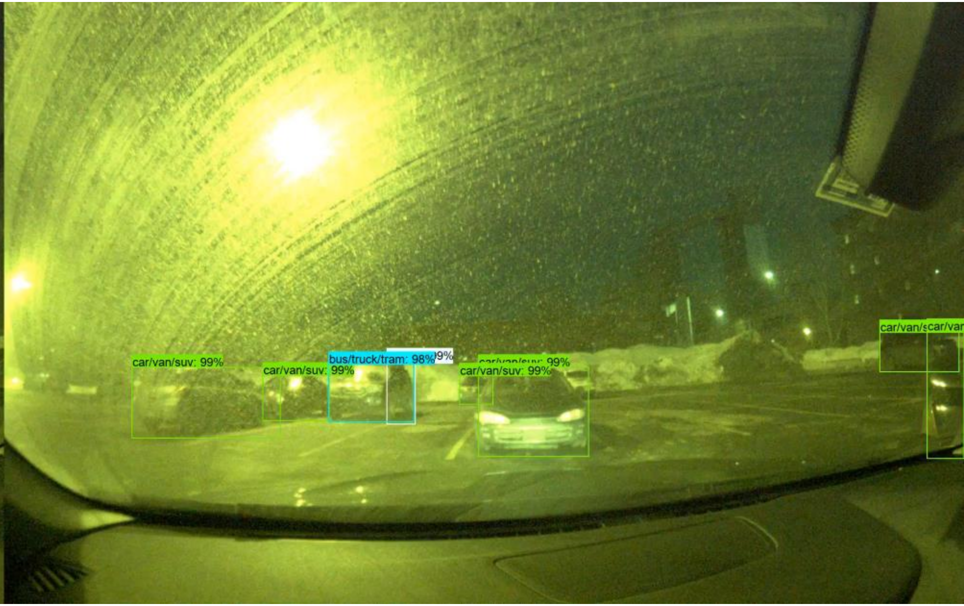


Tesla Autopilot
(Camera + Radar)



Proposed
(Camera-only)

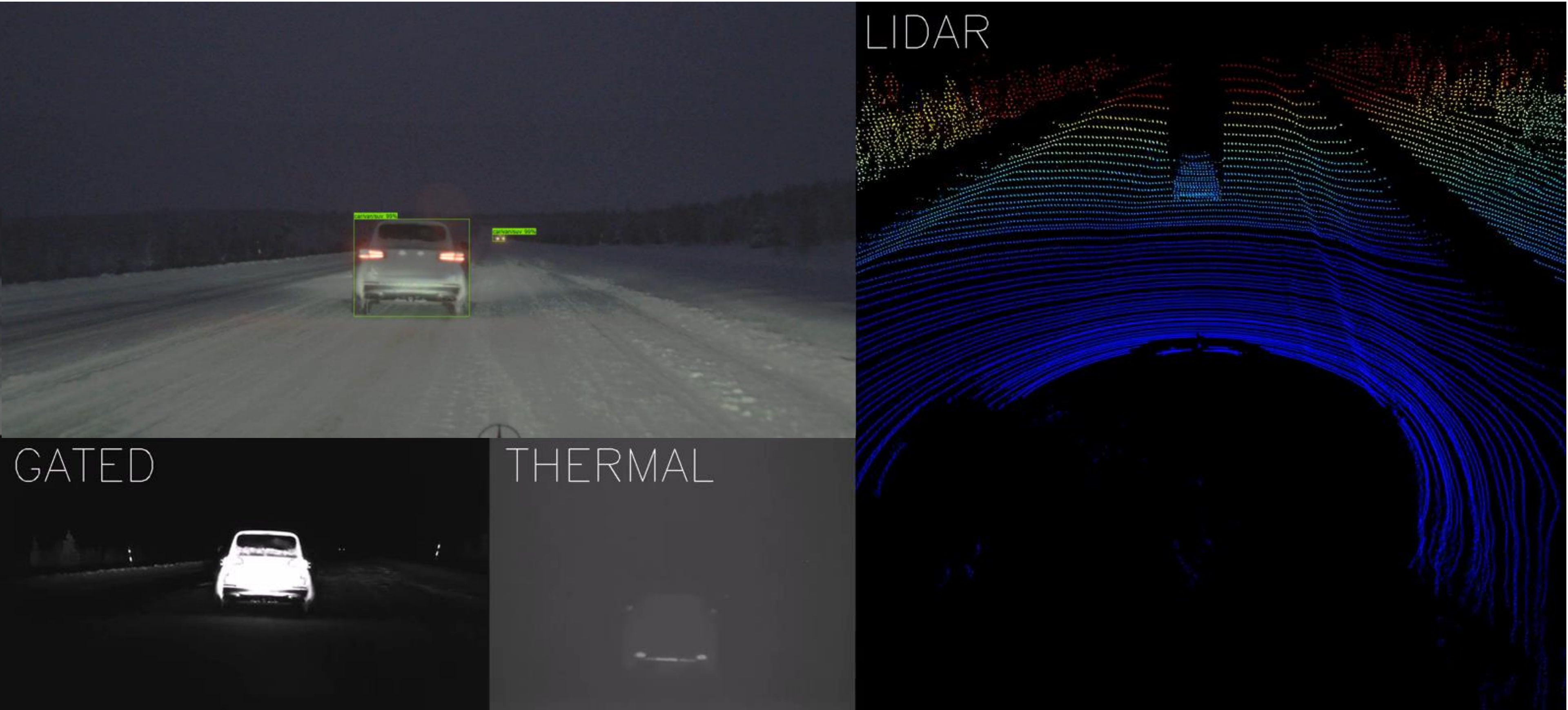
Object Detection Result vs. Nvidia DriveWorks



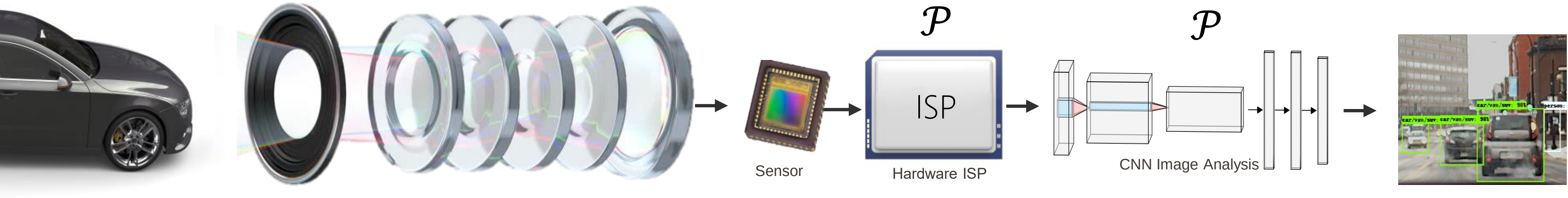
Nvidia Drive
Finetuned for this sensor (AR0231)

Proposed

Low-contrast Measurements in Bad Weather



Optimizing Entire Cameras
Differentiable Compound Optics



Today’s Compound Optics Design in a Box!

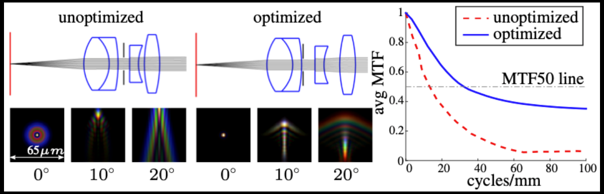
Today's Compound Optics Design in a Box!

Optics Design Software

- Isolated design
- Employ heuristic merit functions
- Black box

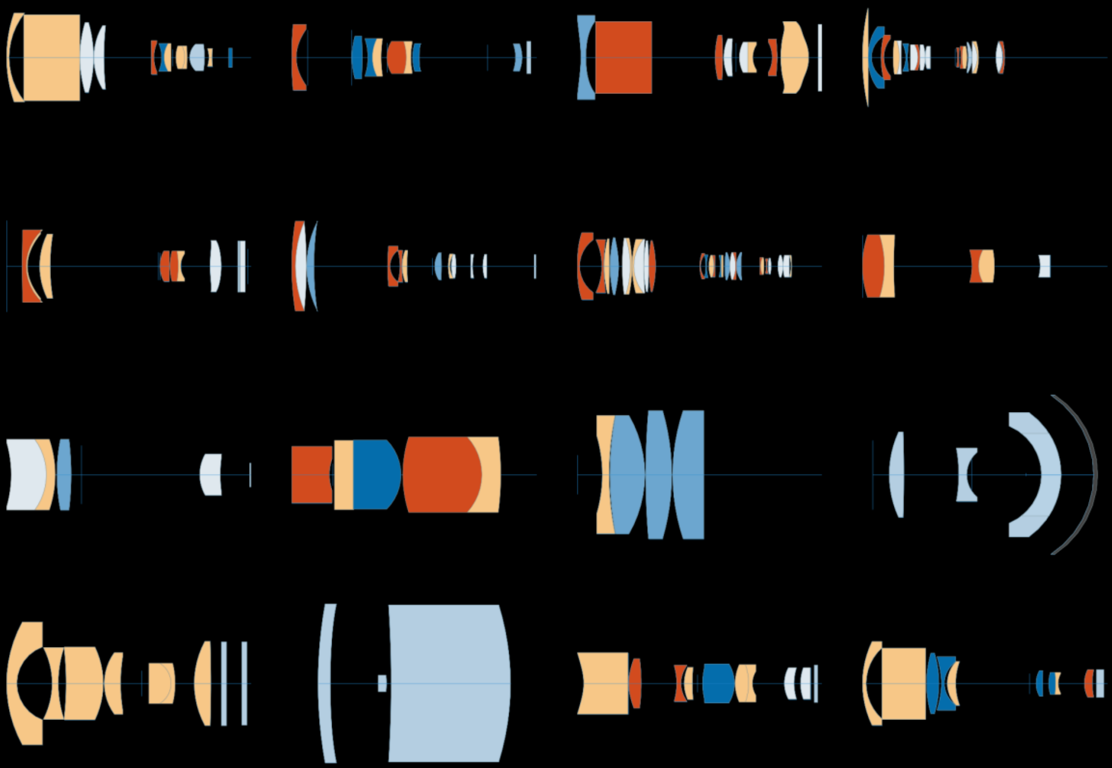
Zemax | OpticStudio™ 15

CODE V Optical Design Software

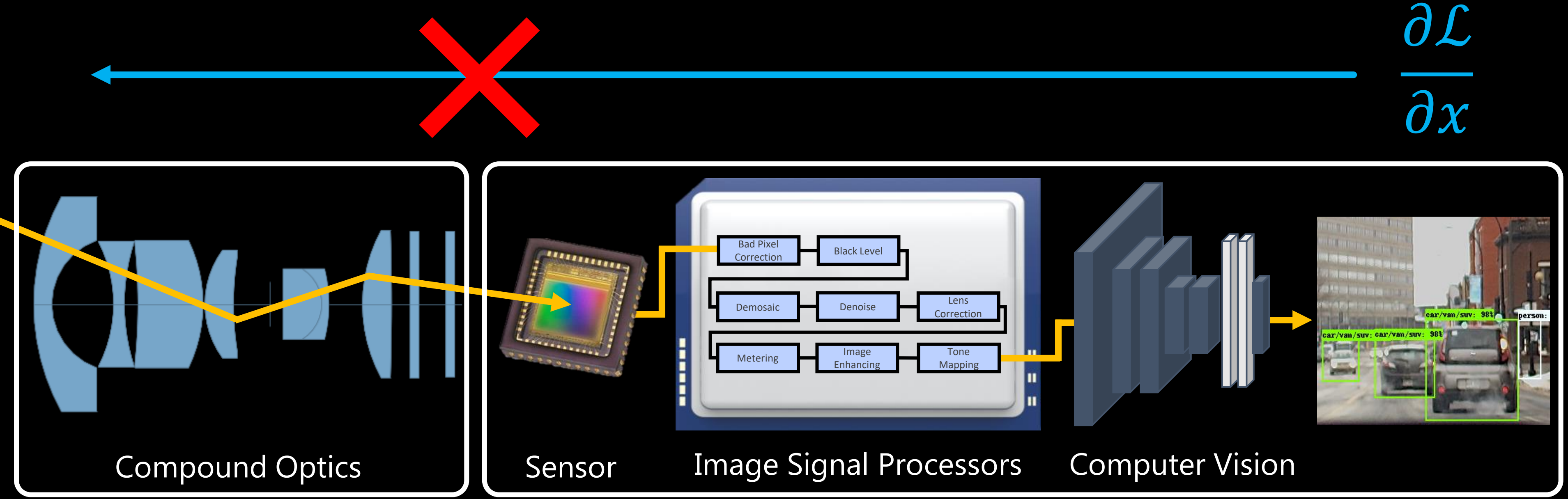


[Geary2002,Garrard2005, Walker2008,Sun2015]

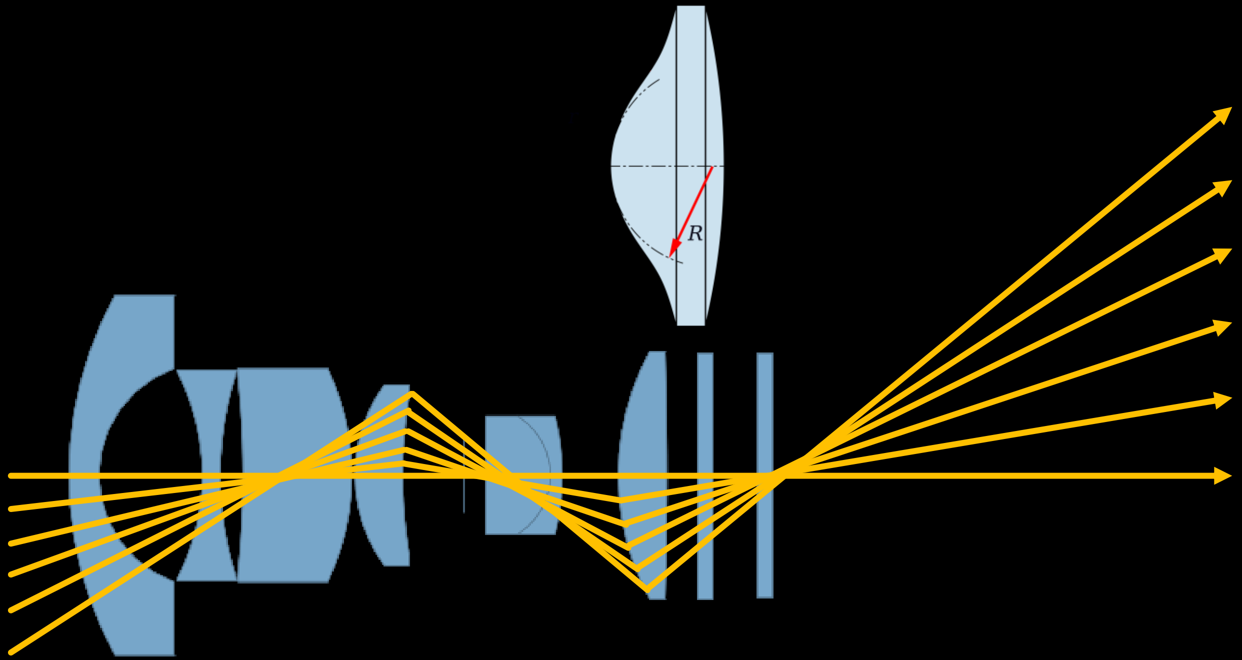
Thousands available online!



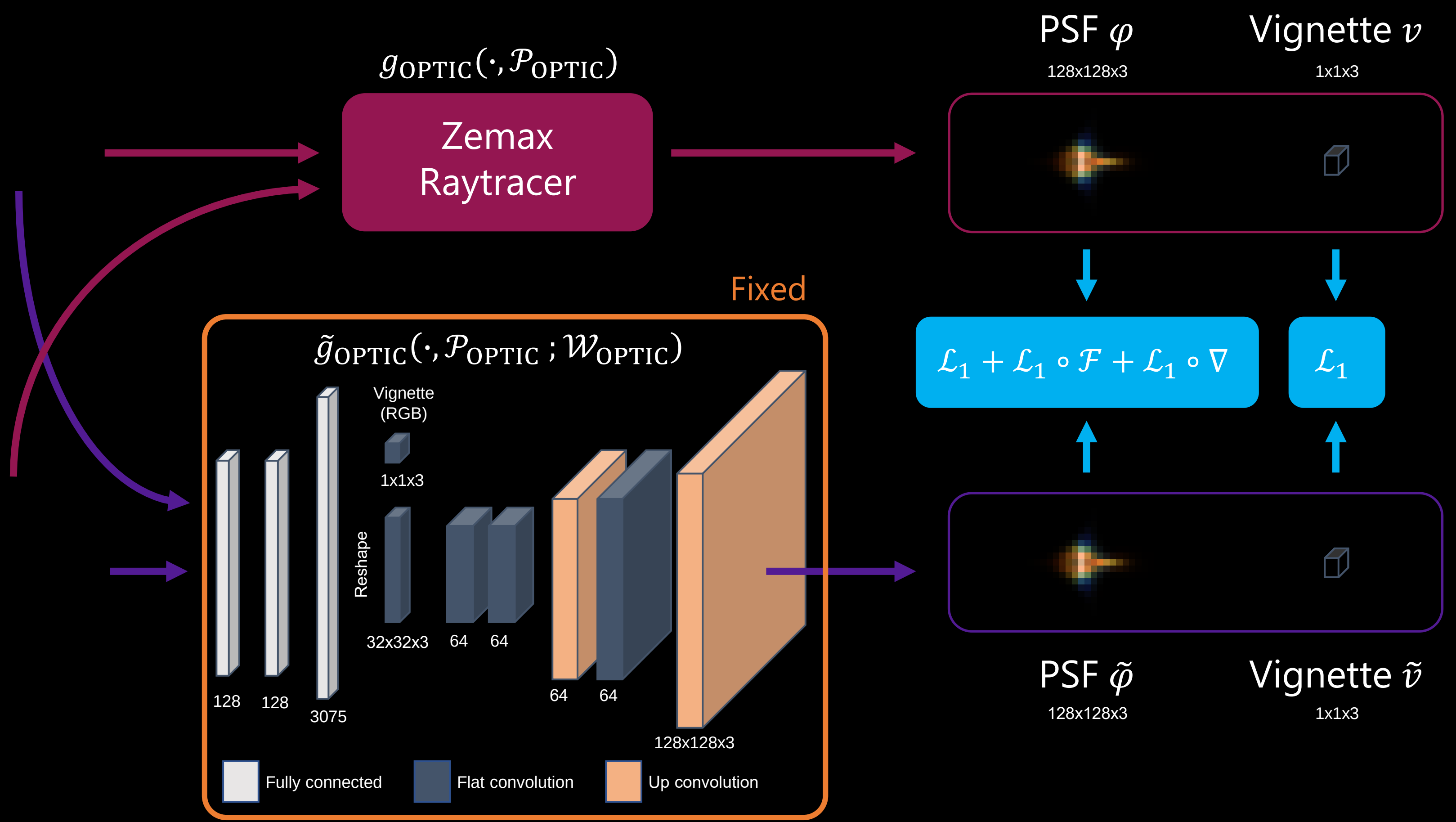
Today's Compound Optics Design in a Box!



This Work – Differentiable Compound Optics

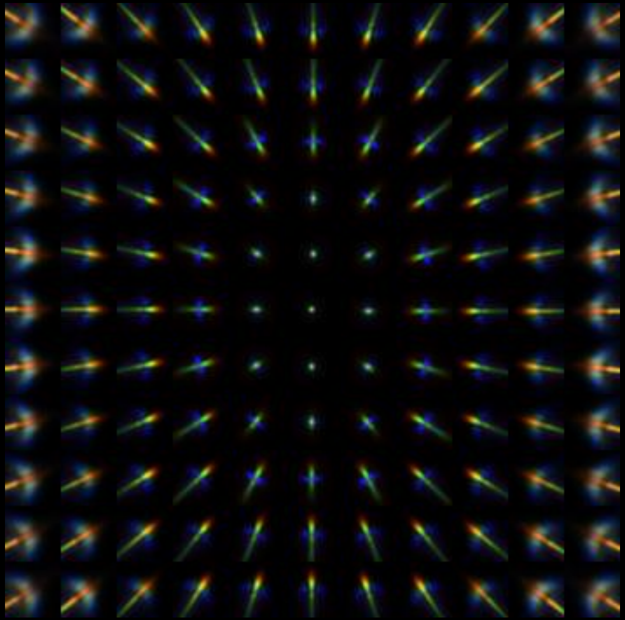


End-to-end Camera Design – Optics Modeling

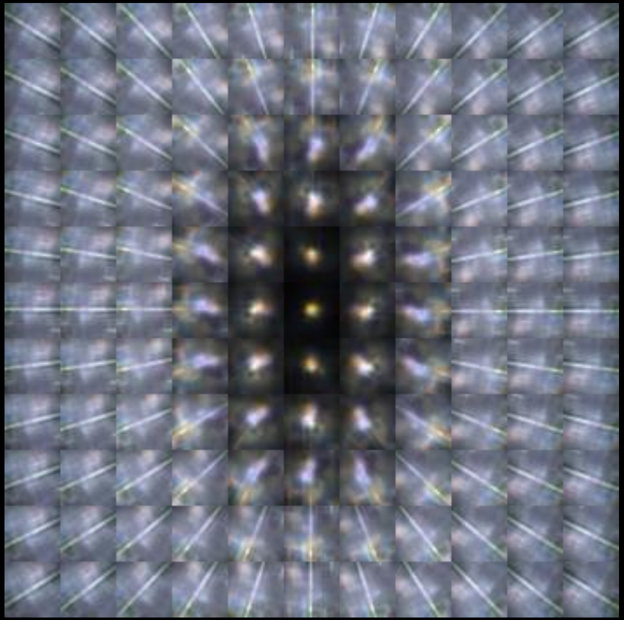


End-to-end Camera Design – Proximal Optimization

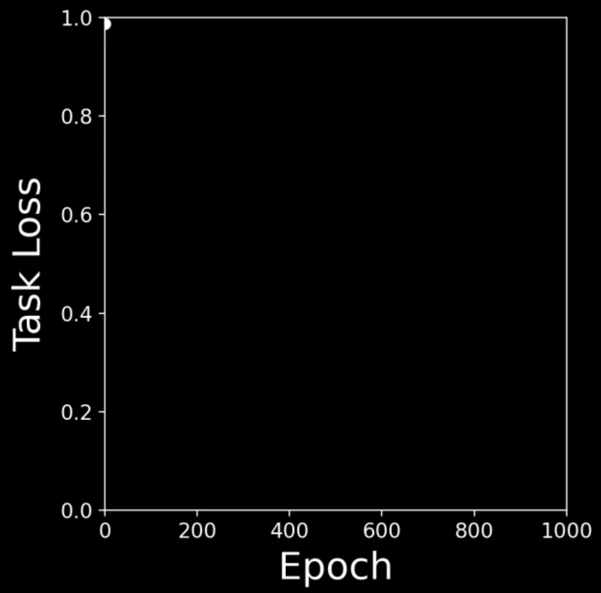
Nominal Optics Design



End-to-end Optimization

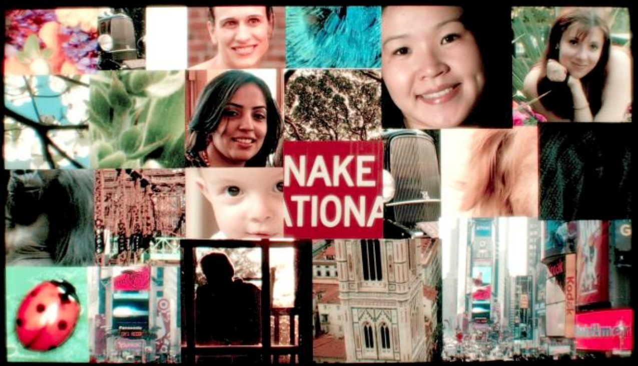


Training Curve

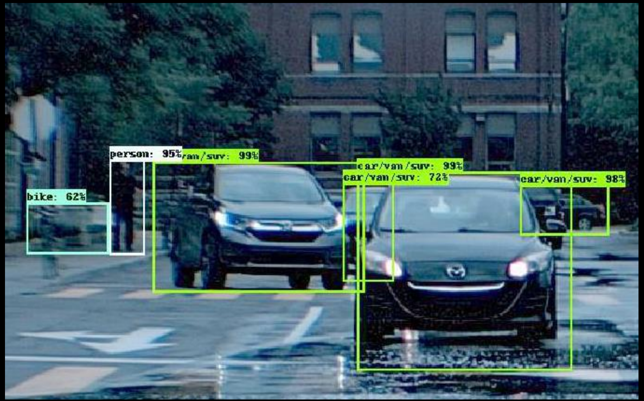


Experimental Results – Task Specific Compound Optics

Natural Image Capture



Object Detection



Traffic Light Detection

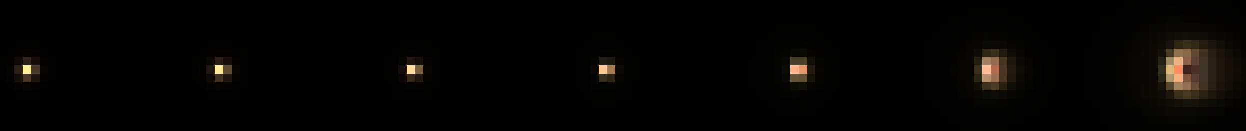
Experimental Results – Natural Image Capture

Optimize $\mathcal{P}_{\mathrm{OPTIC}}$ and $\mathcal{P}_{\mathrm{ISP}}$ to minimize $\mathcal{L}_{\mathrm{TASK}} = \mathcal{L}_1 + \mathcal{L}_{\mathrm{PERCEPTUAL}}$ [Zhang18]

End-to-end Optimized

f/# = 5.8, Focal Length = 33.1mm

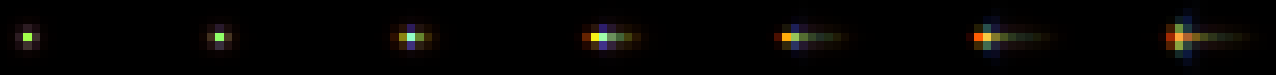
0° 2° 4° 6° 8° 10° 12°



Nominal

f/# = 4.4, Focal Length = 25.0mm

0° 2° 4° 6° 8° 10° 12°



Experimental Results – Natural Image Capture

Optimize $\mathcal{P}_{\text{OPTIC}}$ and $\mathcal{P}_{\text{ISP}}$ to minimize $\mathcal{L}_{\text{TASK}} = \mathcal{L}_1 + \mathcal{L}_{\text{PERCEPTUAL}}$ [Zhang18]

Nominal

f/# = 4.4, Focal Length = 25.0mm

End-to-end Optimized

f/# = 5.8, Focal Length = 33.1mm

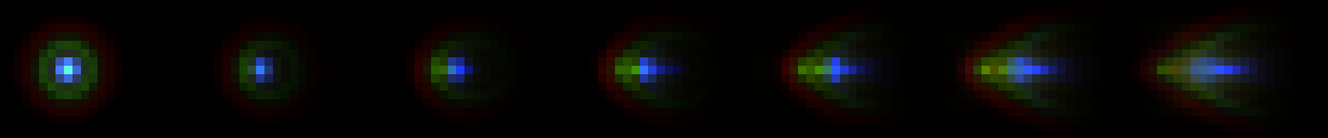
Experimental Results – Automotive Object Detection

Optimize $\mathcal{P}_{\mathrm{OPTIC}}$, $\mathcal{P}_{\mathrm{ISP}}$, and $\mathcal{P}_{\mathrm{NN}}$ to minimize $\mathcal{L}_{\mathrm{TASK}}$ = Intersection over Union loss

End-to-end Optimized

f/# = 3.2, Focal Length = 18.5mm

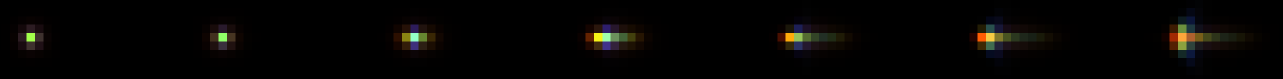
0° 2° 4° 6° 8° 10° 12°



Nominal

f/# = 4.4, Focal Length = 25.0mm

0° 2° 4° 6° 8° 10° 12°



Experimental Results – Automotive Object Detection

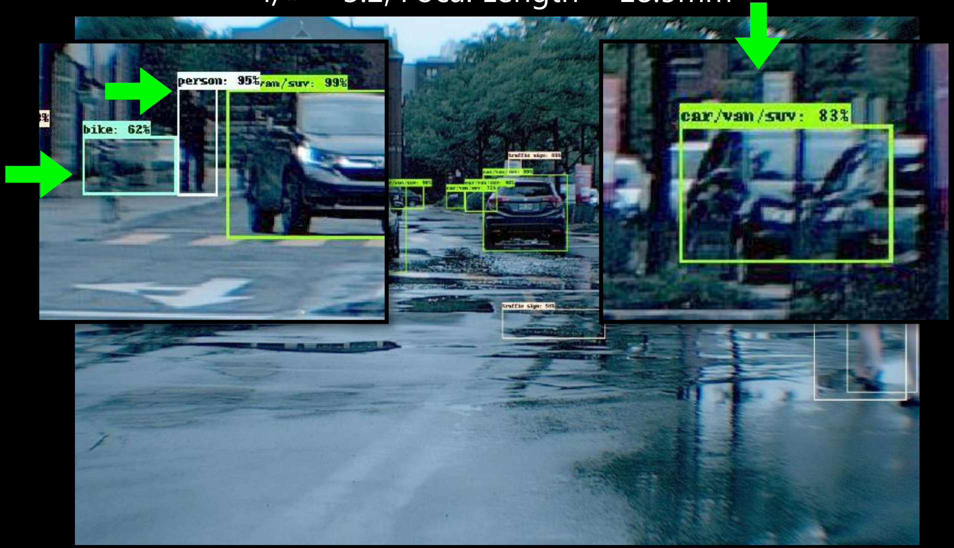
Optimize $\mathcal{P}_{\mathrm{OPTIC}}$, $\mathcal{P}_{\mathrm{ISP}}$, and $\mathcal{P}_{\mathrm{NN}}$ to minimize $\mathcal{L}_{\mathrm{TASK}}$ = Intersection over Union loss

Nominal

f/# = 4.4, Focal Length = 25.0mm

End-to-end Optimized

f/# = 3.2, Focal Length = 18.5mm



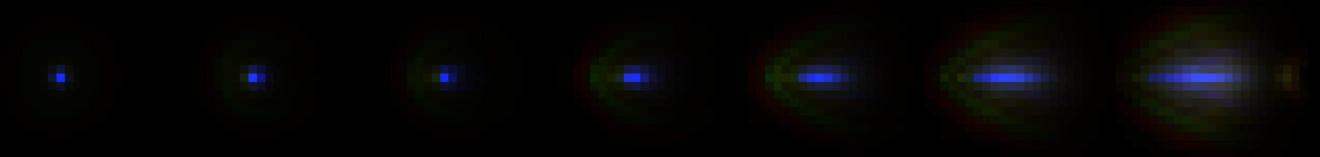
Experimental Results – Traffic Light Detection

Optimize $\mathcal{P}_{\mathrm{OPTIC}}$, $\mathcal{P}_{\mathrm{ISP}}$, and $\mathcal{P}_{\mathrm{NN}}$ to minimize $\mathcal{L}_{\mathrm{TASK}}$ = Intersection over Union loss

End-to-end Optimized

f/# = 3.3, Focal Length = 18.8mm

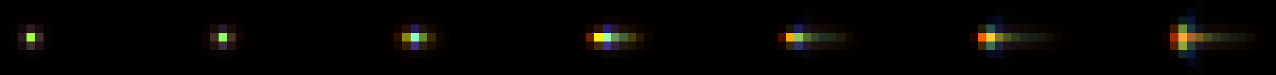
0° 2° 4° 6° 8° 10° 12°



Nominal

f/# = 4.4, Focal Length = 25.0mm

0° 2° 4° 6° 8° 10° 12°



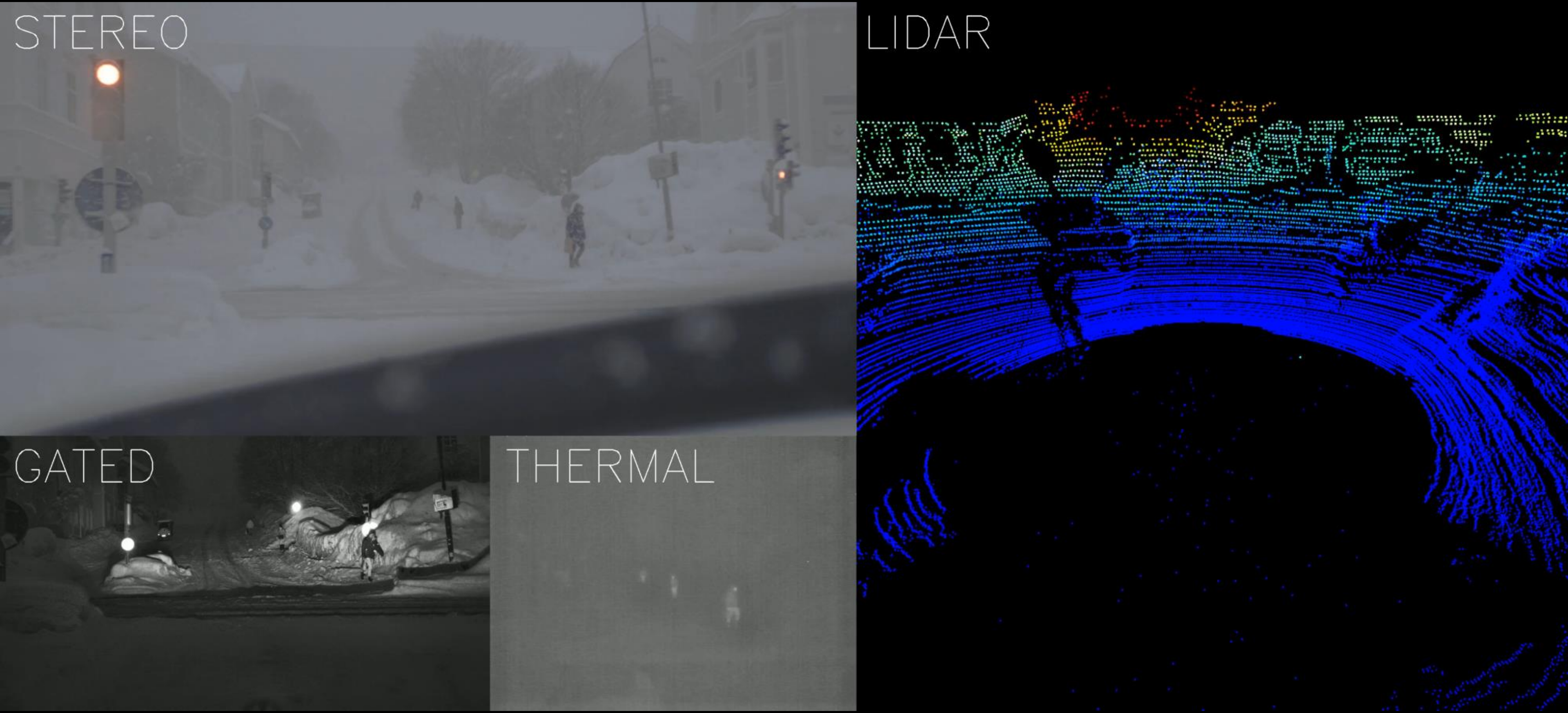
Experimental Results – Traffic Light Detection

Optimize $\mathcal{P}_{\mathrm{OPTIC}}$, $\mathcal{P}_{\mathrm{ISP}}$, and $\mathcal{P}_{\mathrm{NN}}$ to minimize $\mathcal{L}_{\mathrm{TASK}}$ = Intersection over Union loss

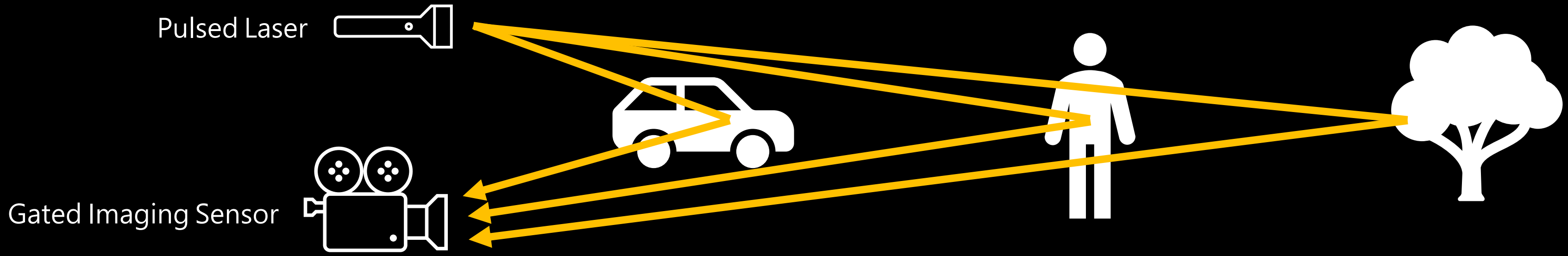
Nominal
f/# = 4.4, Focal Length = 25.0mm

End-to-end Optimized
f/# = 3.3, Focal Length = 18.8mm

Robust New Sensors:
3D Detection in the Presence of Backscatter


STEREO
LIDAR
GATED
THERMAL

Gated Imaging

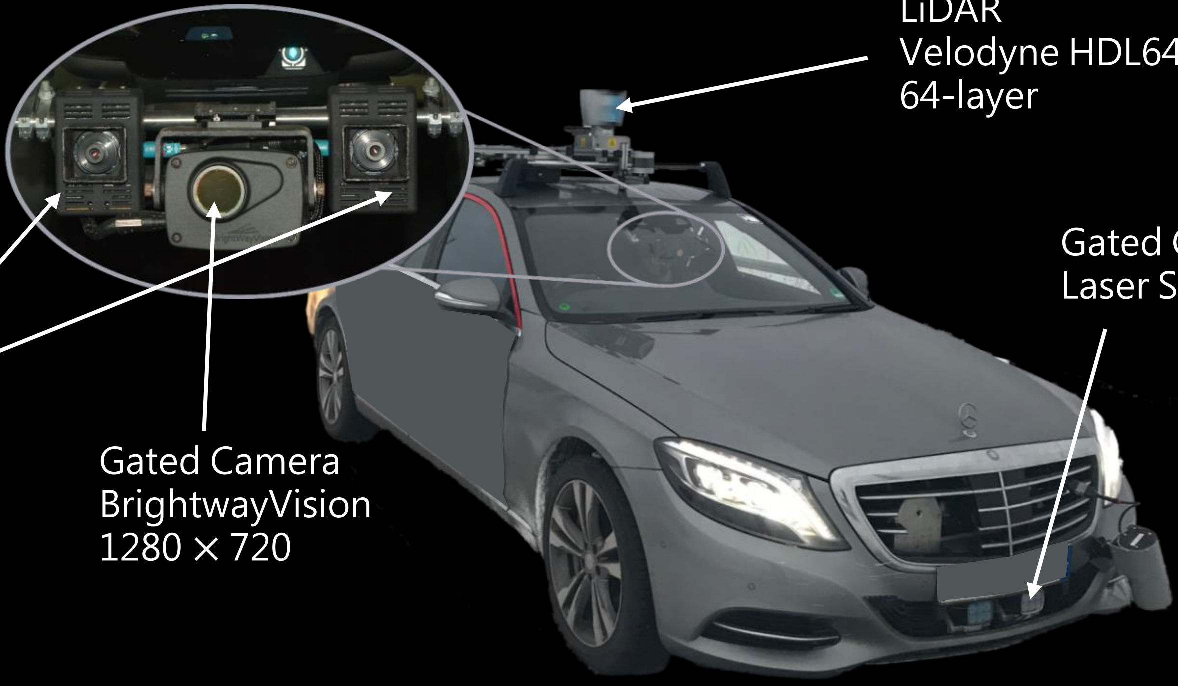


Slice 1

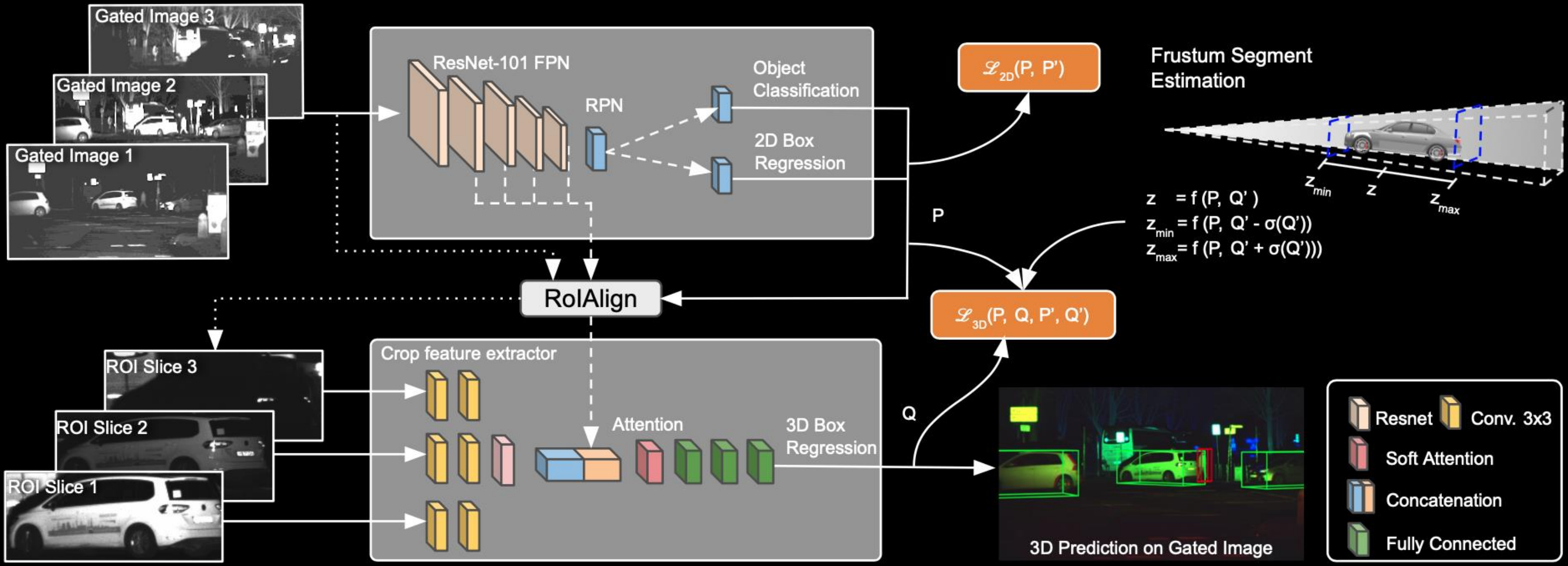
Slice 2

Slice 3

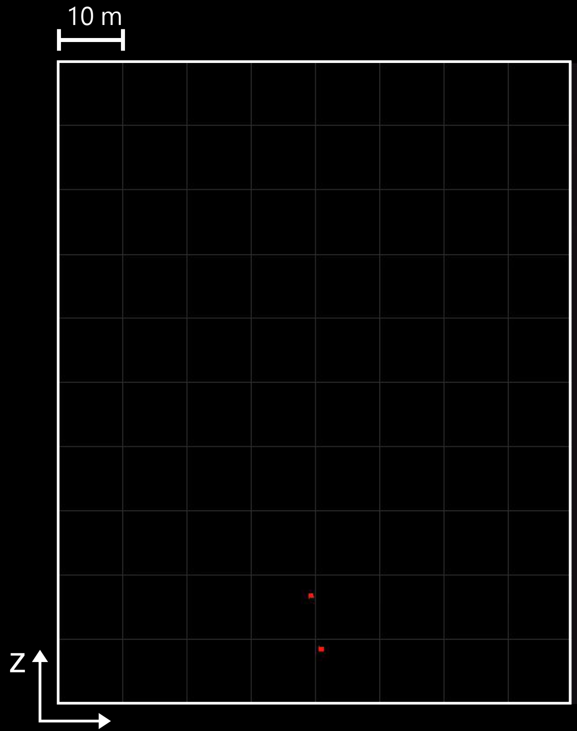
Vehicle Setup



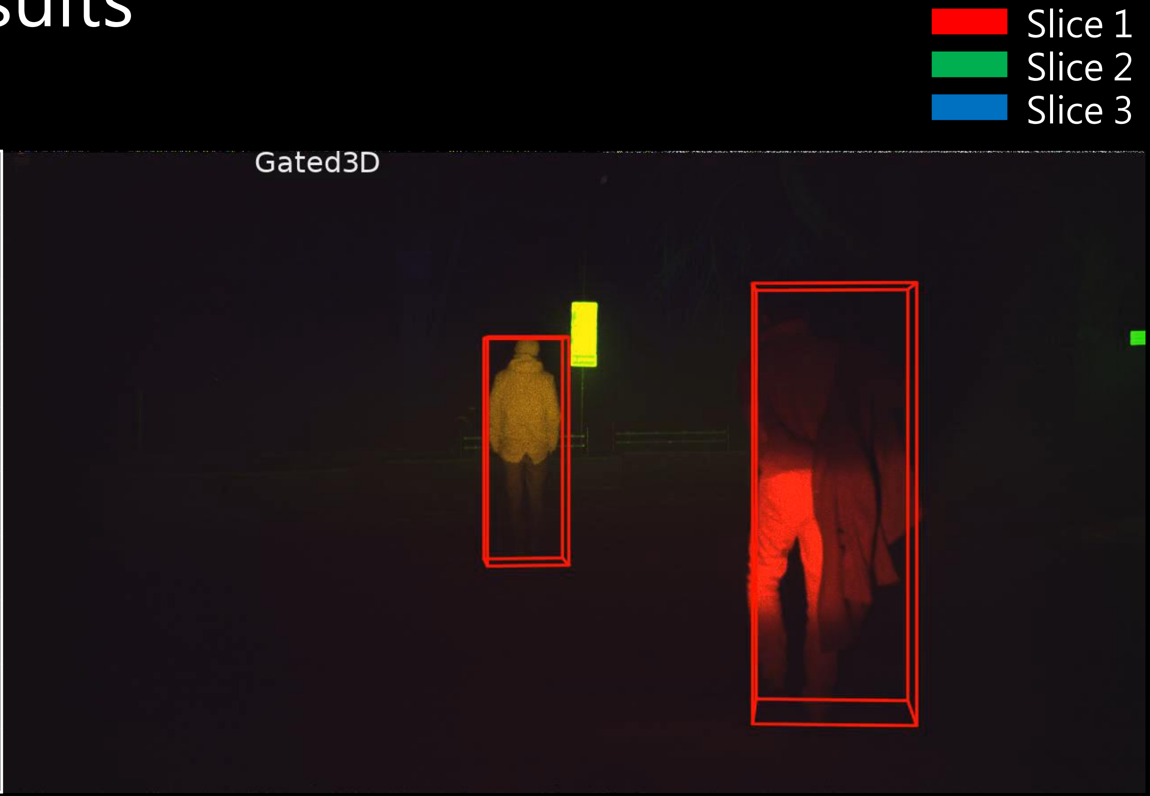
Gated3D Architecture



Qualitative Results

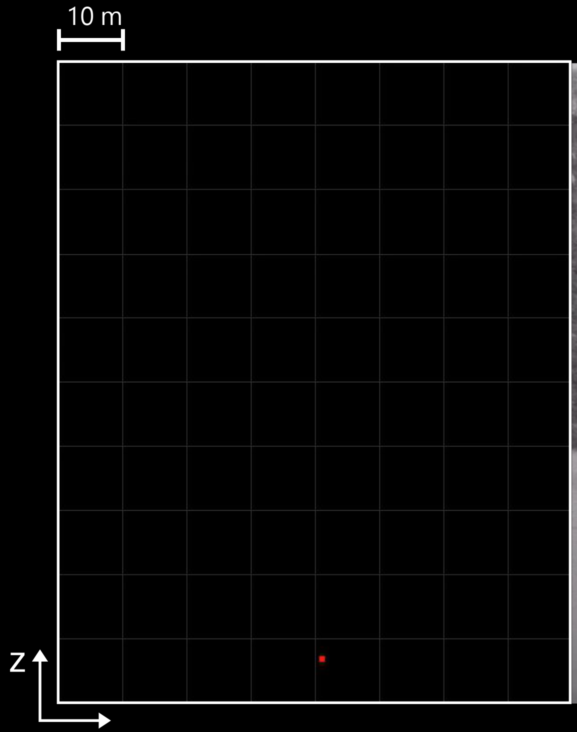


Bird's Eye View



Gated Camera View

Qualitative Results

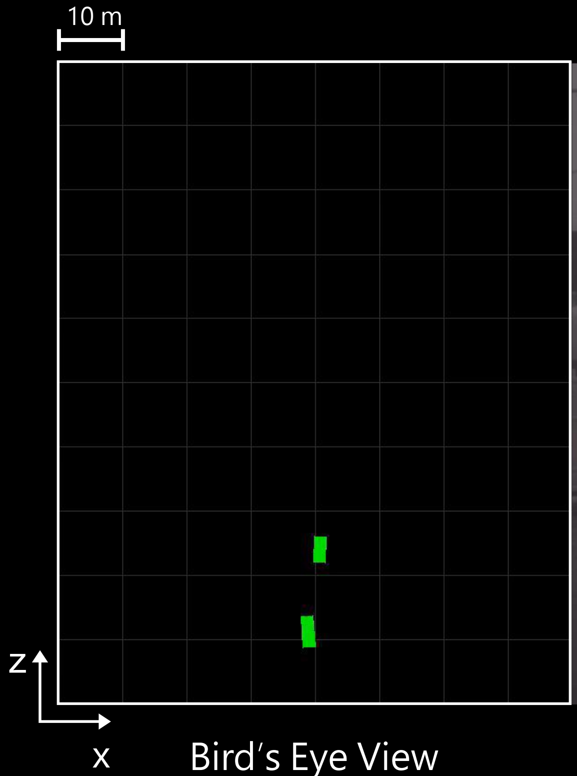


Bird's Eye View

Gated Camera View

Qualitative Results

Slice 1
Slice 2
Slice 3

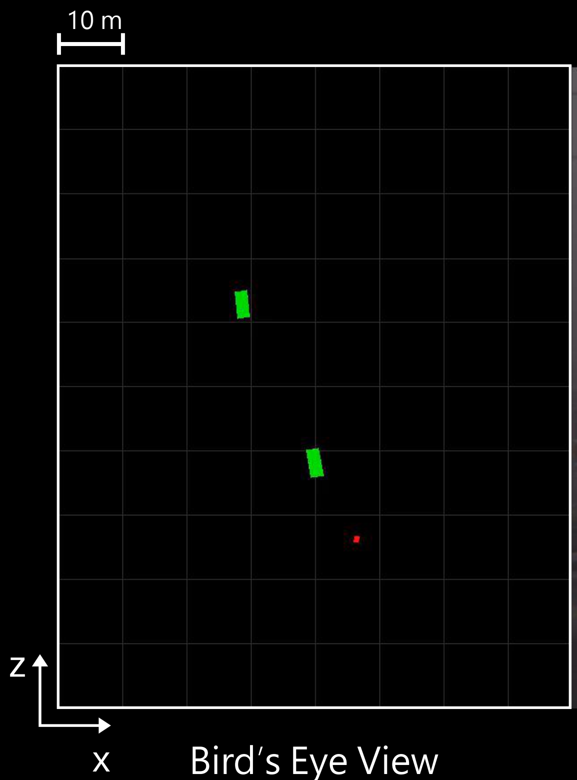


Bird's Eye View

Gated Camera View

Qualitative Results

Slice 1
Slice 2
Slice 3



Bird's Eye View

Gated Camera View

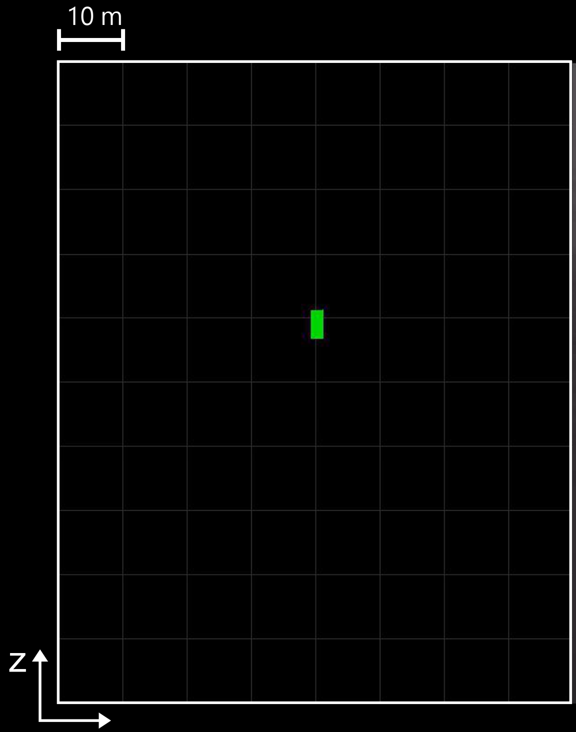
Qualitative Results

Bird's Eye View

Gated Camera View

Qualitative Results

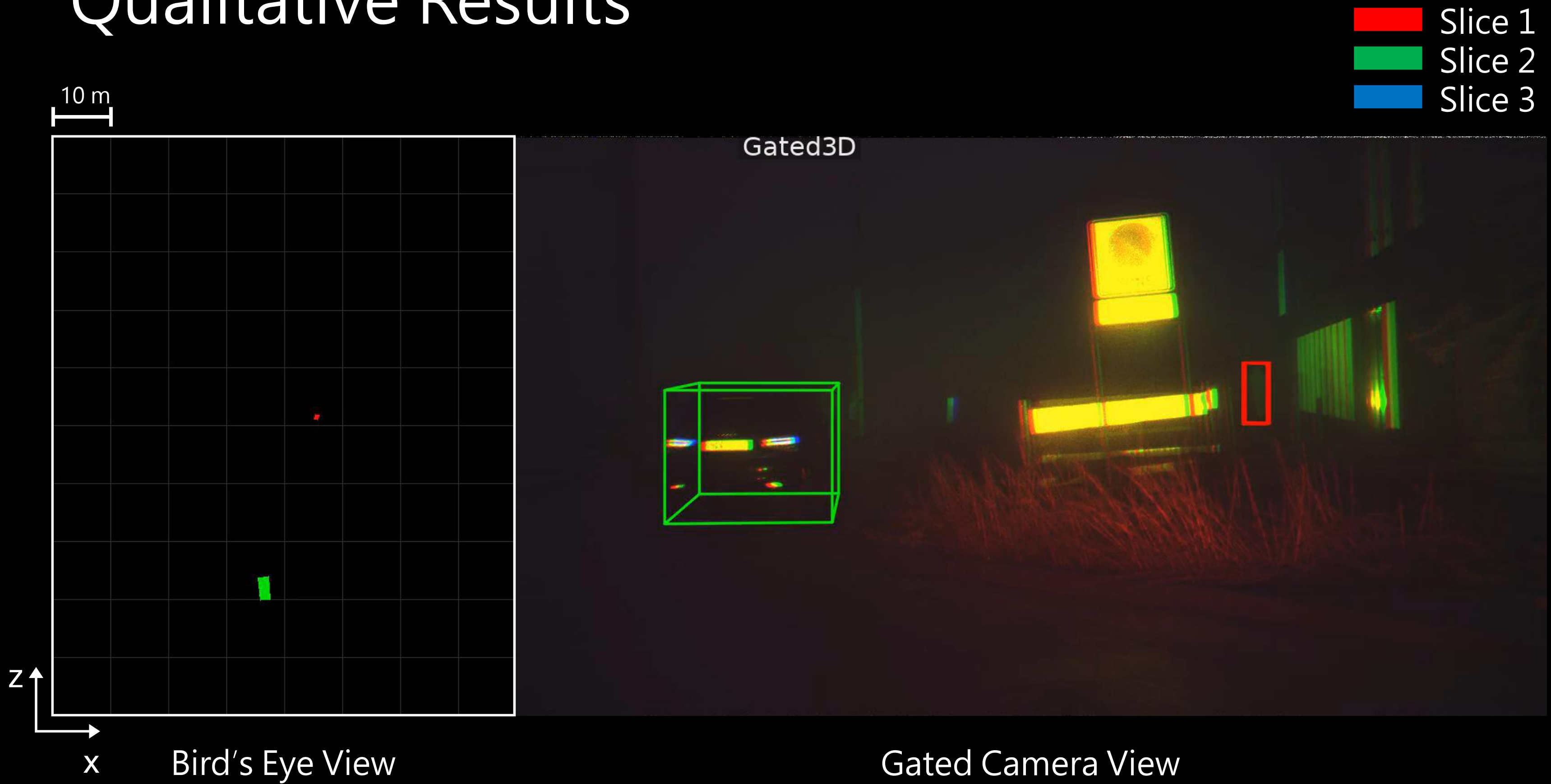
Slice 1
Slice 2
Slice 3



Bird's Eye View

Gated Camera View

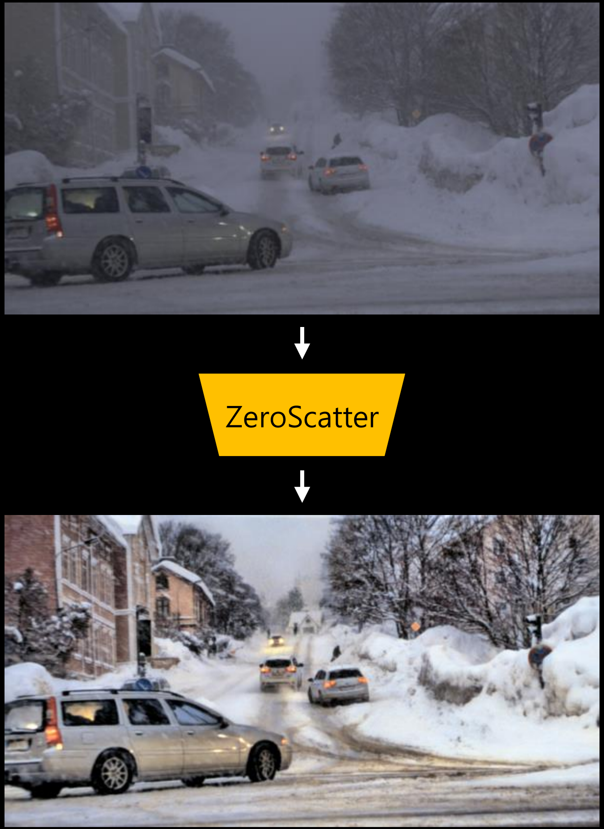
Qualitative Results



Bird's Eye View

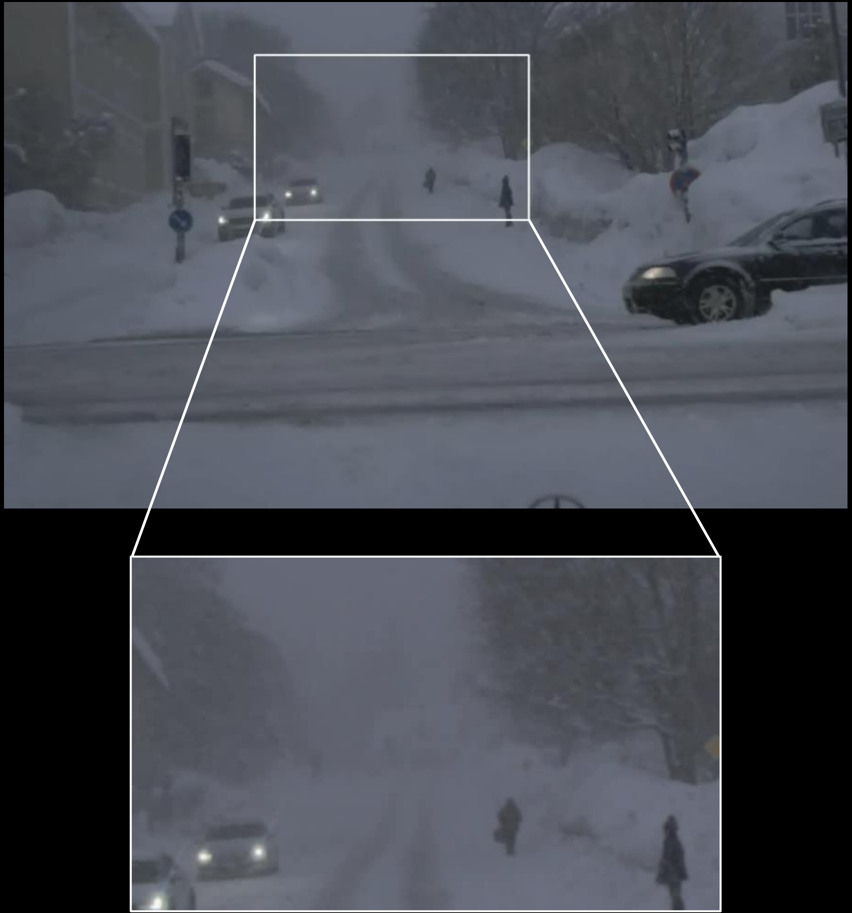
Gated Camera View

Gating for Supervision from RGB



→ map to gated image + loss

Input (Heavy Snow)



ZeroScatter Output


Input (Dense Fog)
ZeroScatter Output

Differentiate Through Scenes:

Neural Scene Graphs for Inference

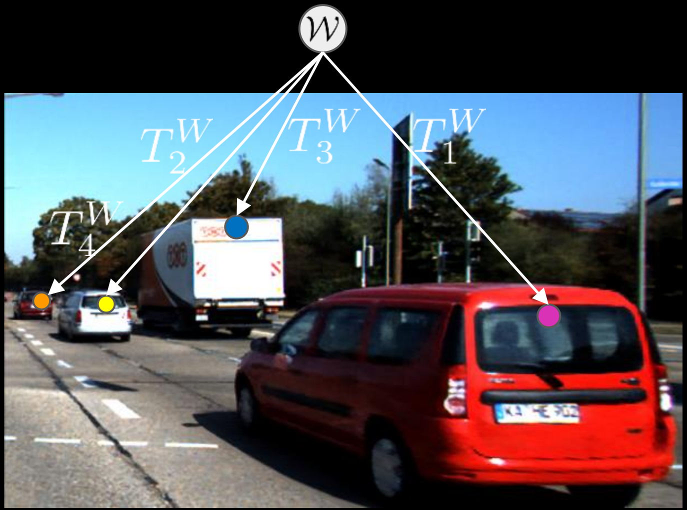
Reference

Dynamic Automotive Scene

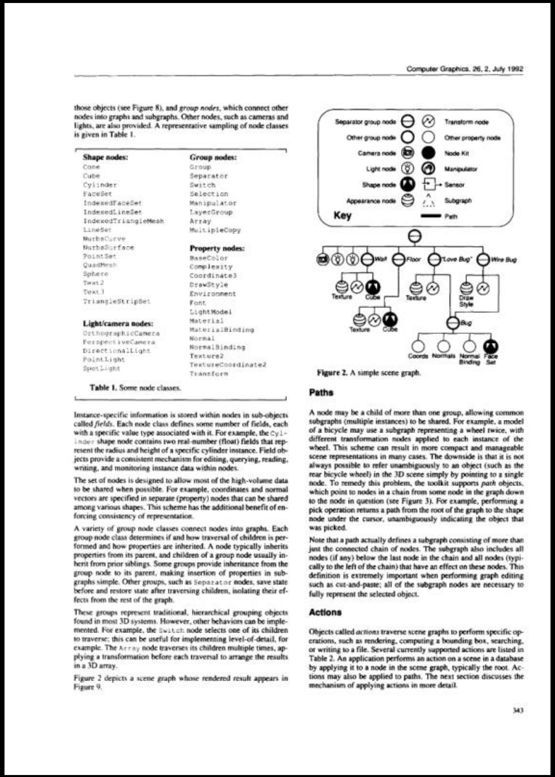
Frame T

Neural Scene Graphs

[Strauss et al., 1992]



Neural Scene Graph Representation



those objects (see Figure 8), and *group nodes*, which connect other nodes into graphs and subgraphs. Other nodes, such as cameras and lights, are also provided. A representative sampling of node classes is given in Table 1.

Shape nodes:	Group nodes:
Cone	Group
Cube	Separator
Cylinder	Switch
FaceSet	Selection
IndexedFaceSet	Manipulator
IndexedLineSet	LayerGroup
IndexedTriangleMesh	Array
LineSet	MultipleCopy
NurbsCurve	
NurbsSurface	**Property nodes:**
PointSet	BaseColor
QuadMesh	Complexity
Sphere	Coordinate3
Text2	DrawStyle
Text3	Environment
TriangleStripSet	Font
	LightModel
Light/camera nodes:	Material
OrthographicCamera	MaterialBinding
PerspectiveCamera	Normal
DirectionalLight	NormalBinding
PointLight	Texture2
SpotLight	TextureCoordinate2
	Transform

Table 1. Some node classes.

Instance-specific information is stored within nodes in sub-objects called *fields*. Each node class defines some number of fields, each with a specific value type associated with it. For example, the `Cylinder` shape node contains two real-number (float) fields that represent the radius and height of a specific cylinder instance. Field objects provide a consistent mechanism for editing, querying, reading, writing, and monitoring instance data within nodes.

The set of nodes is designed to allow most of the high-volume data to be shared when possible. For example, coordinates and normal vectors are specified in separate (property) nodes that can be shared among various shapes. This scheme has the additional benefit of enforcing consistency of representation.

A variety of group node classes connect nodes into graphs. Each group node class determines if and how traversal of children is performed and how properties are inherited. A node typically inherits properties from its parent, and children of a group node usually inherit from prior siblings. Some groups provide inheritance from the group node to its parent, making insertion of properties in subgraphs simple. Other groups, such as `Separator` nodes, save state before and restore state after traversing children, isolating their effects from the rest of the graph.

These groups represent traditional, hierarchical grouping objects found in most 3D systems. However, other behaviors can be implemented. For example, the `Switch` node selects one of its children to traverse; this can be useful for implementing level-of-detail, for example. The `Array` node traverses its children multiple times, applying a transformation before each traversal to arrange the results in a 3D array.

Figure 2 depicts a scene graph whose rendered result appears in Figure 9.

Separator group node
Transform node
Other group node
Other property node
Camera node
Node Kit
Light node
Manipulator
Shape node
Sensor
Appearance node
Subgraph
Key
Path
Wall
Floor
"Love Bug"
Wire Bug
Texture
Cube
Texture
Cube
Texture
Draw Style
Bug
Coords
Normals
Normal Binding
Face Set

Figure 2. A simple scene graph.

Paths

A node may be a child of more than one group, allowing common subgraphs (multiple instances) to be shared. For example, a model of a bicycle may use a subgraph representing a wheel twice, with different transformation nodes applied to each instance of the wheel. This scheme can result in more compact and manageable scene representations in many cases. The downside is that it is not always possible to refer unambiguously to an object (such as the rear bicycle wheel) in the 3D scene simply by pointing to a single node. To remedy this problem, the toolkit supports *path* objects, which point to nodes in a chain from some node in the graph down to the node in question (see Figure 3). For example, performing a pick operation returns a path from the root of the graph to the shape node under the cursor, unambiguously indicating the object that was picked.

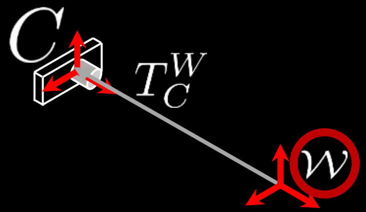
Note that a path actually defines a subgraph consisting of more than just the connected chain of nodes. The subgraph also includes all nodes (if any) below the last node in the chain and all nodes (typically to the left of the chain) that have an effect on these nodes. This definition is extremely important when performing graph editing such as cut-and-paste; all of the subgraph nodes are necessary to fully represent the selected object.

Actions

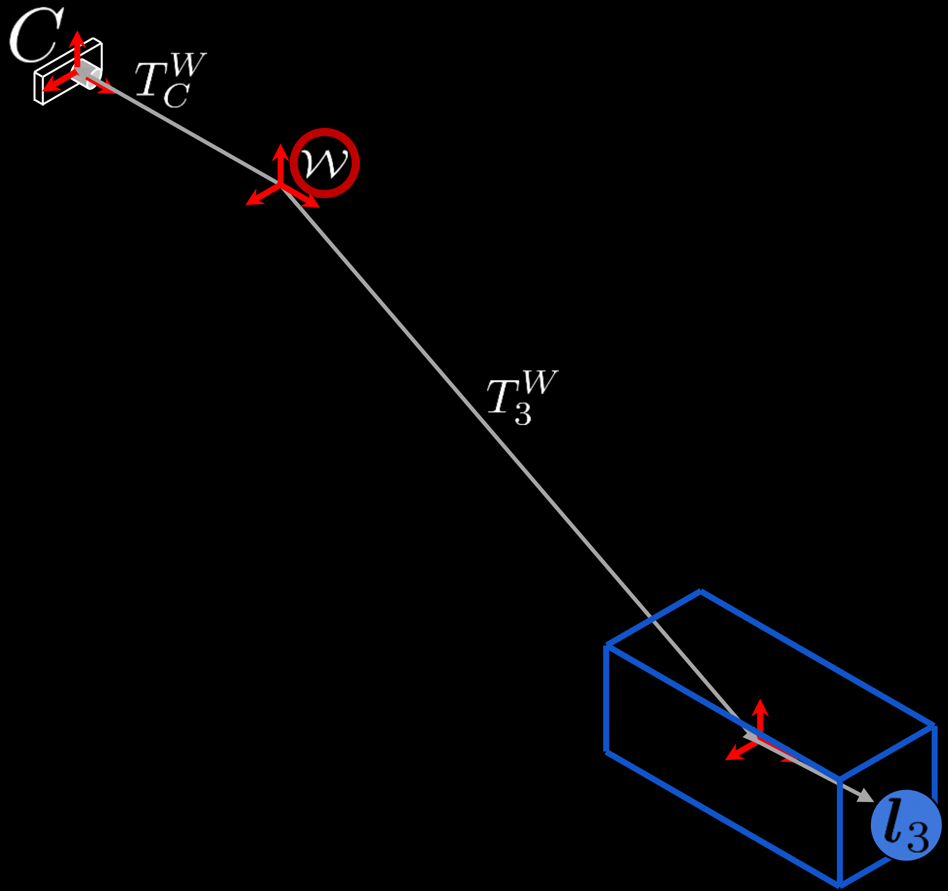
Objects called *actions* traverse scene graphs to perform specific operations, such as rendering, computing a bounding box, searching, or writing to a file. Several currently supported actions are listed in Table 2. An application performs an action on a scene in a database by applying it to a node in the scene graph, typically the root. Actions may also be applied to paths. The next section discusses the mechanism of applying actions in more detail.

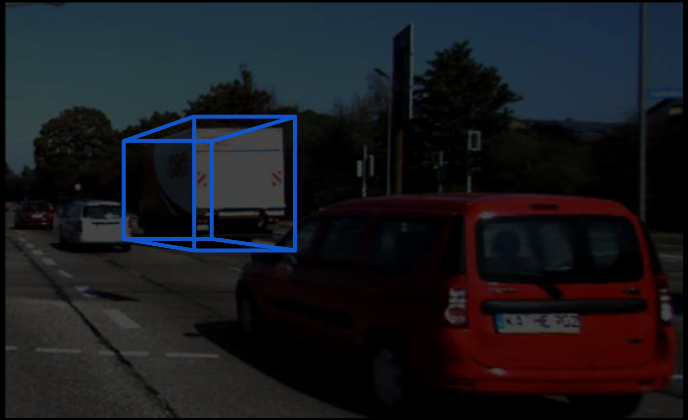
Scene Graphs in Graphics

Pinhole Camera
Observer

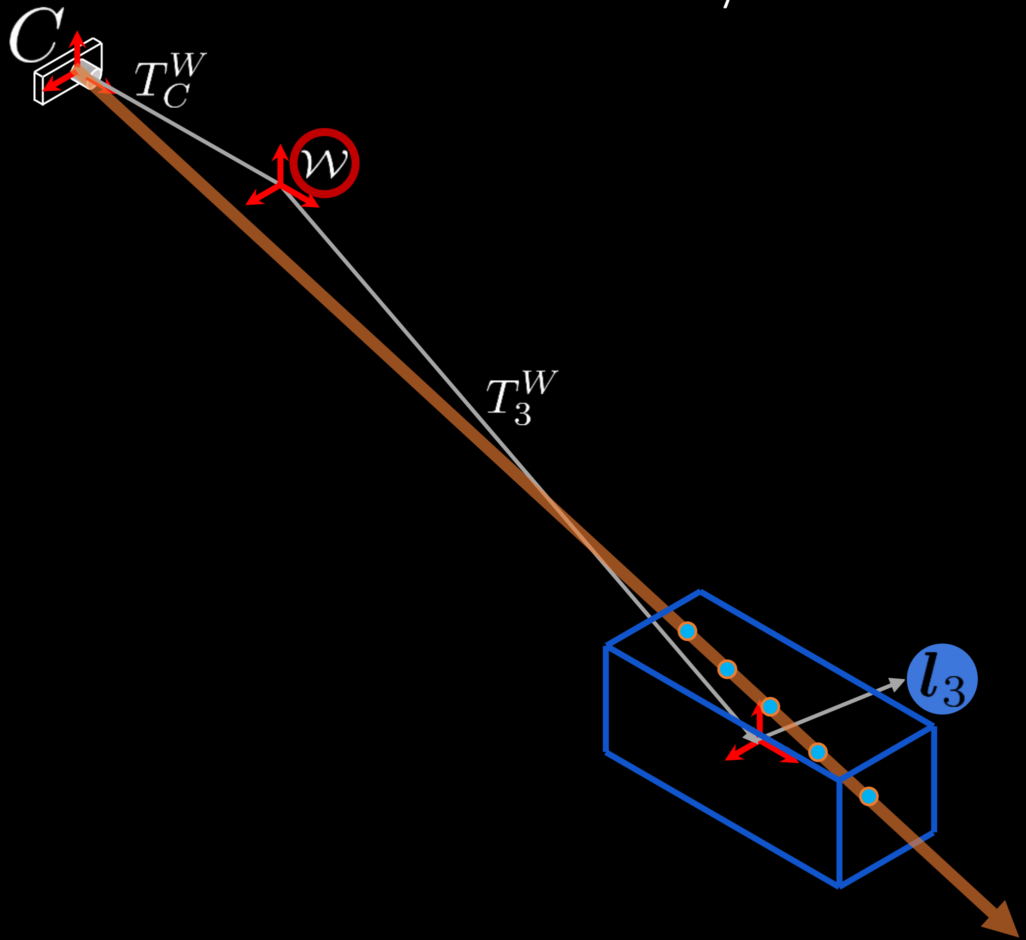


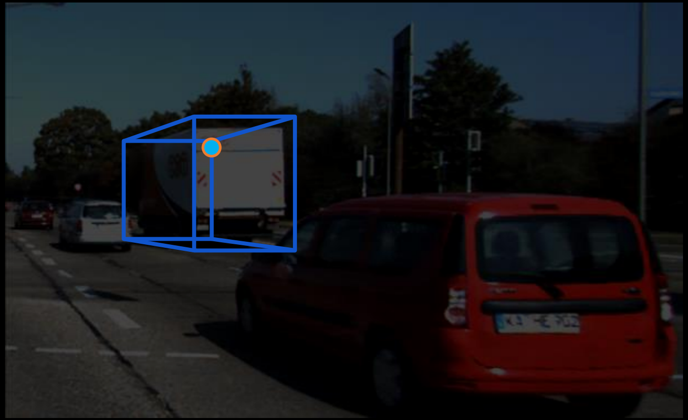
Object Bounding Boxes



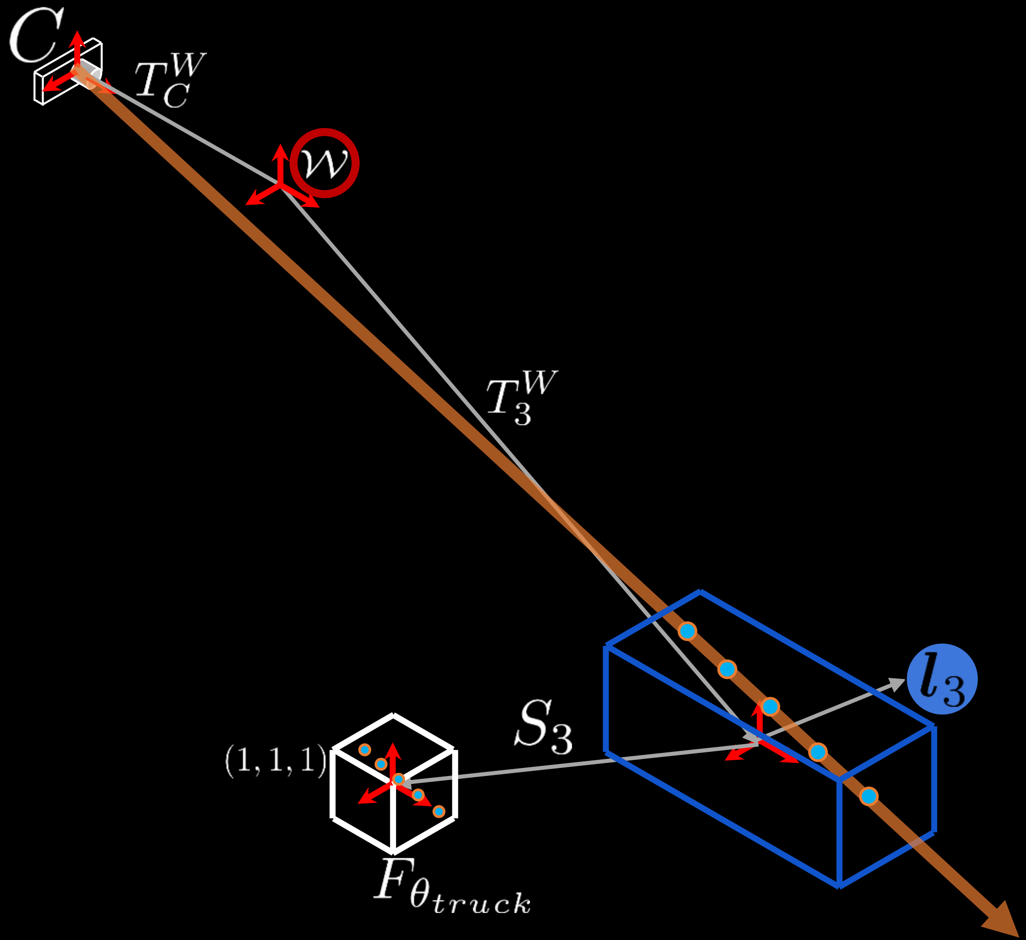


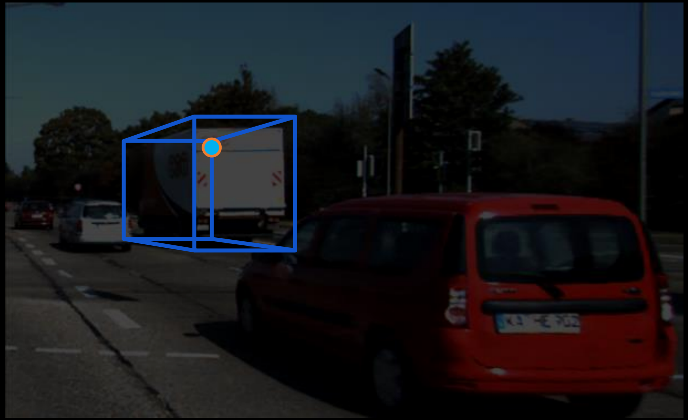
Sampling Points between Box-Ray Intersections



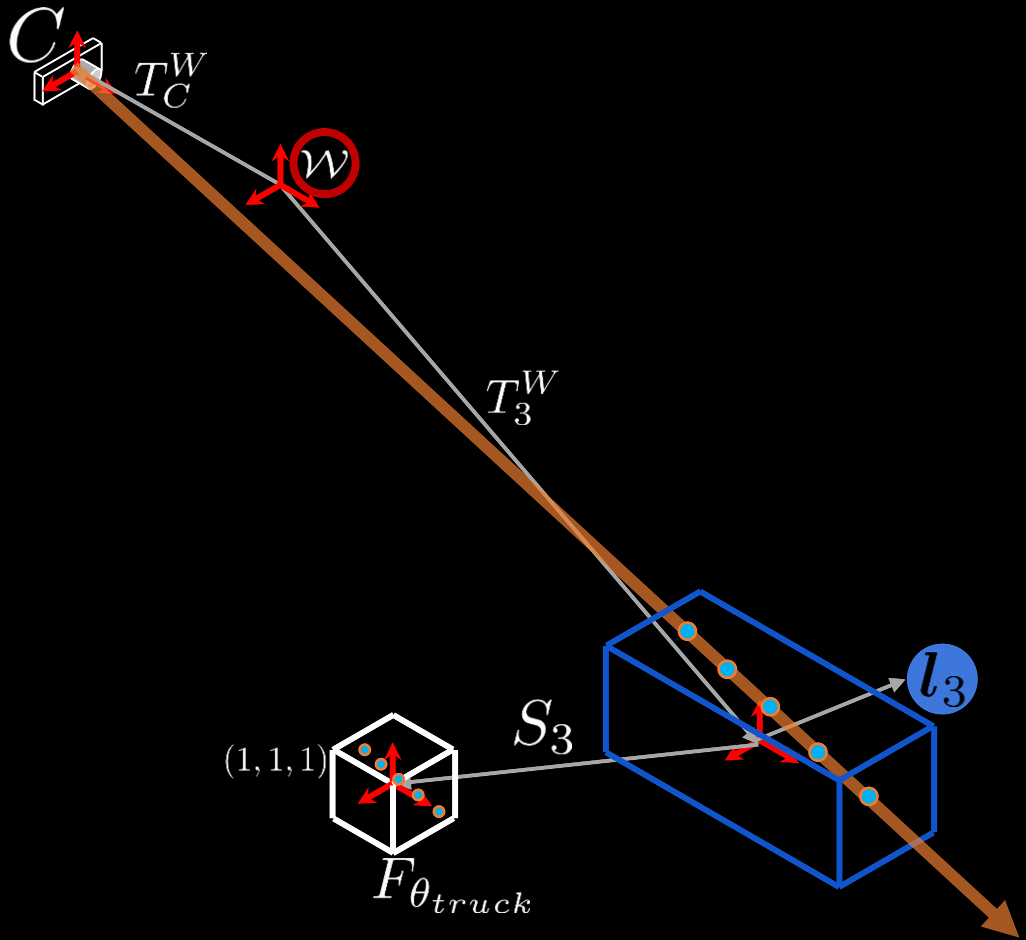


Object Radiance





Object Radiance Field



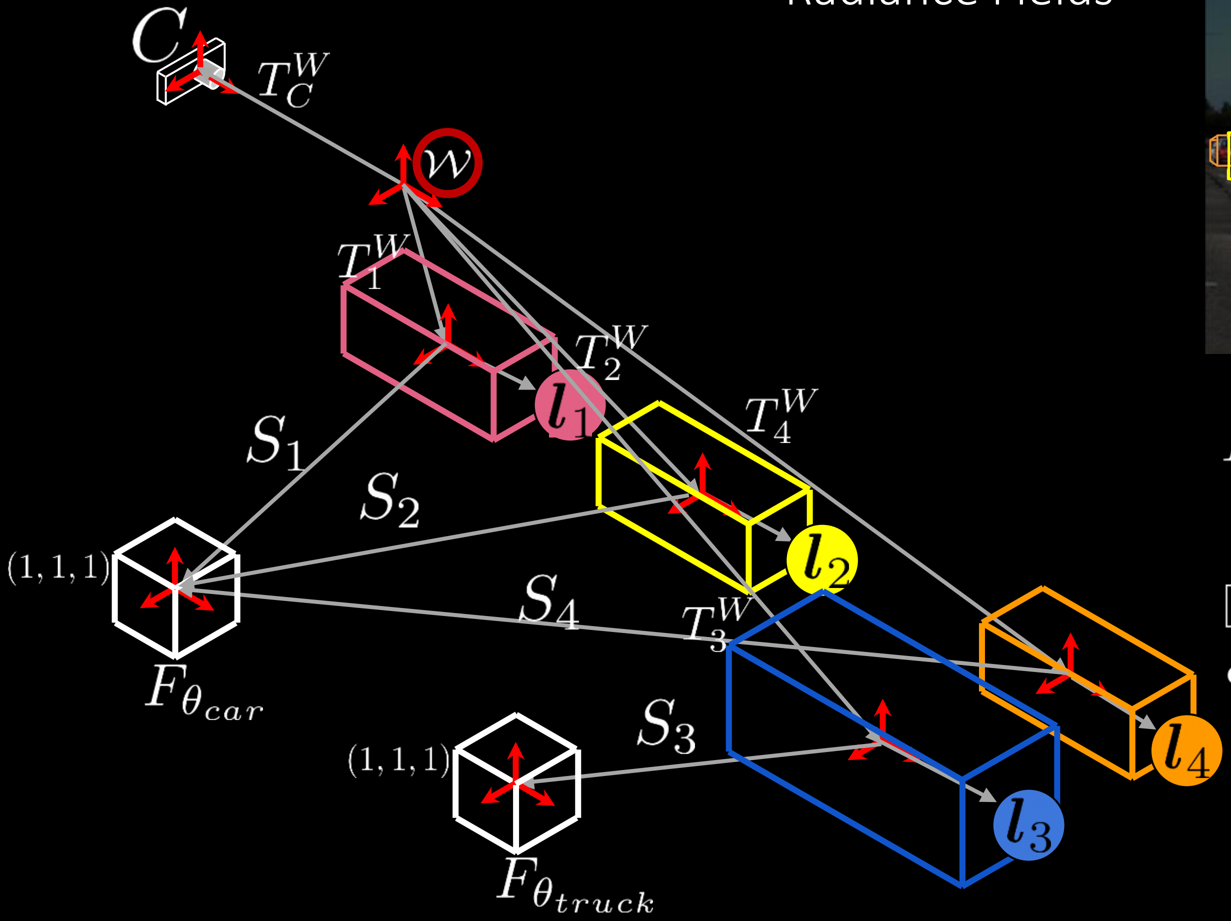
$$F_{\theta_c} : (\mathbf{x}, \mathbf{d}, \mathbf{l}_o, \mathbf{p}_o) \rightarrow (\mathbf{c}, \sigma)$$

MLP, 2 Stages:

$$[\mathbf{y}(\mathbf{x}, \mathbf{l}_o), \sigma(\mathbf{x})] = F_{\theta_{c,1}}(\gamma_x(\mathbf{x}), \mathbf{l}_o)$$

$$\mathbf{c}(\mathbf{x}, \mathbf{l}_o, \mathbf{p}_o) = F_{\theta_{c,2}}(\gamma_d(\mathbf{d}), \mathbf{y}(\mathbf{x}, \mathbf{l}_o), \mathbf{p}_o)$$

Shared Object Radiance Fields



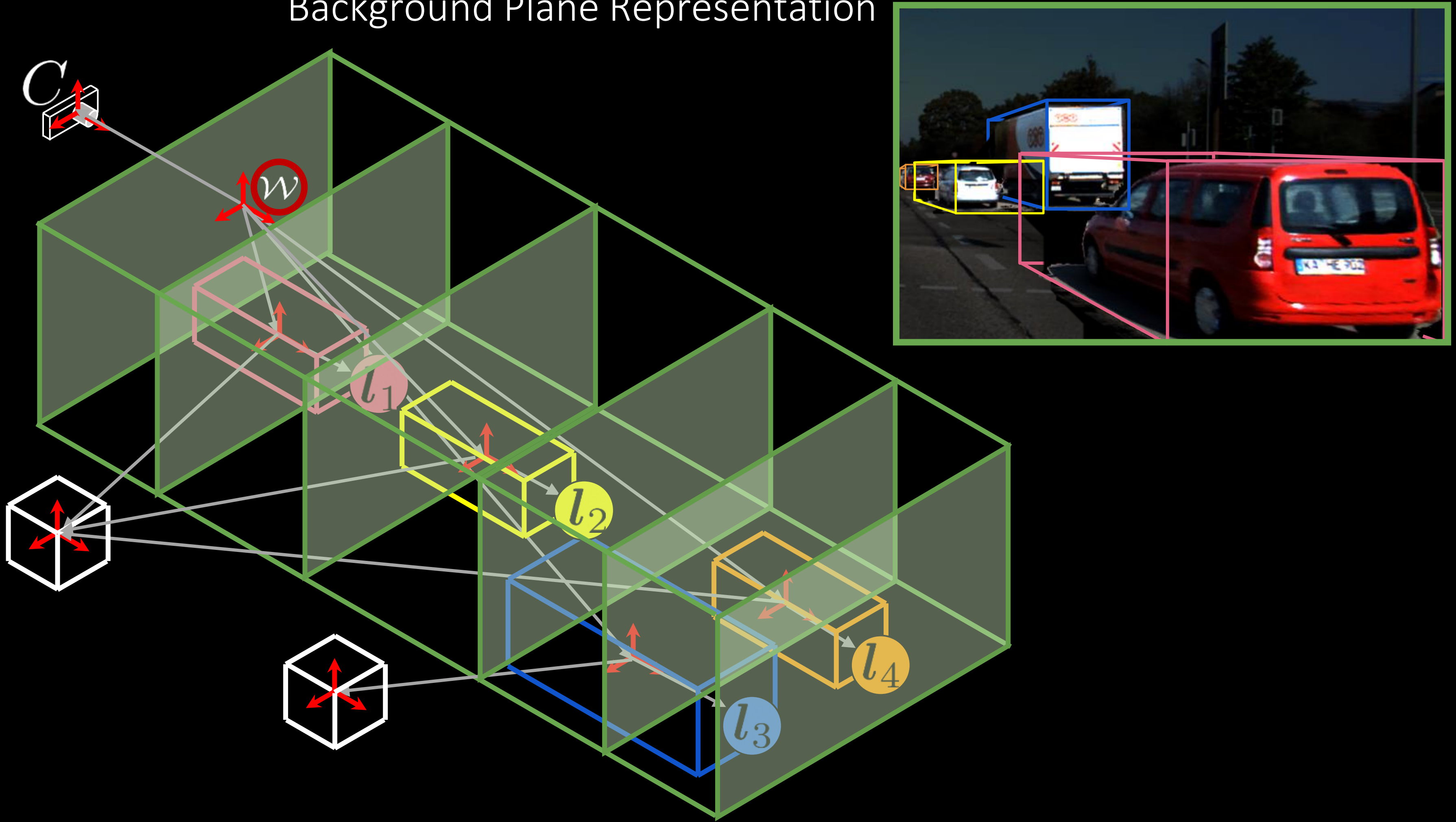
$$F_{\theta_c} : (\mathbf{x}, \mathbf{d}, \mathbf{l}_o, \mathbf{p}_o) \rightarrow (\mathbf{c}, \sigma)$$

MLP, 2 Stages:

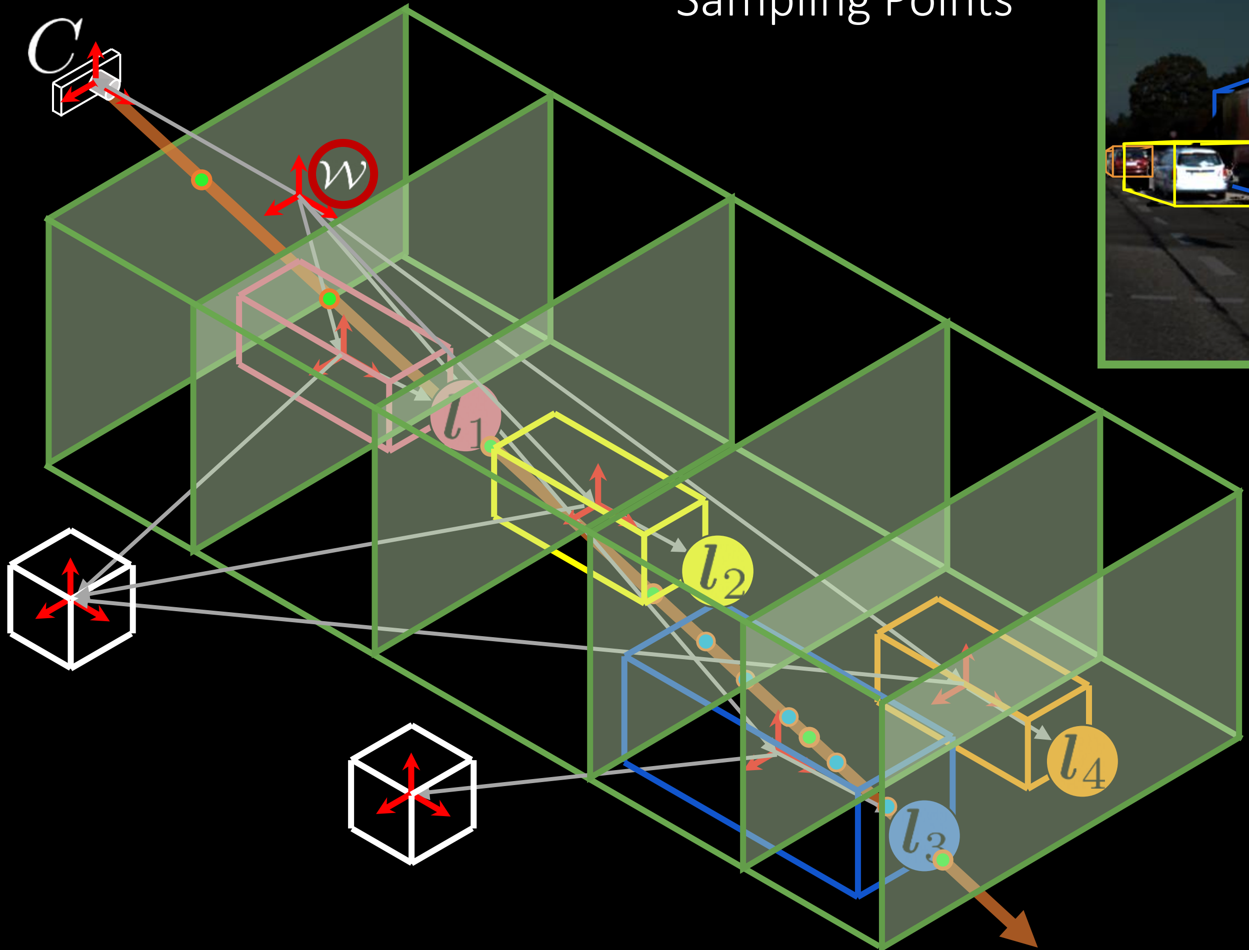
$$[\mathbf{y}(\mathbf{x}, \mathbf{l}_o), \sigma(\mathbf{x})] = F_{\theta_{c,1}}(\gamma_x(\mathbf{x}), \mathbf{l}_o)$$

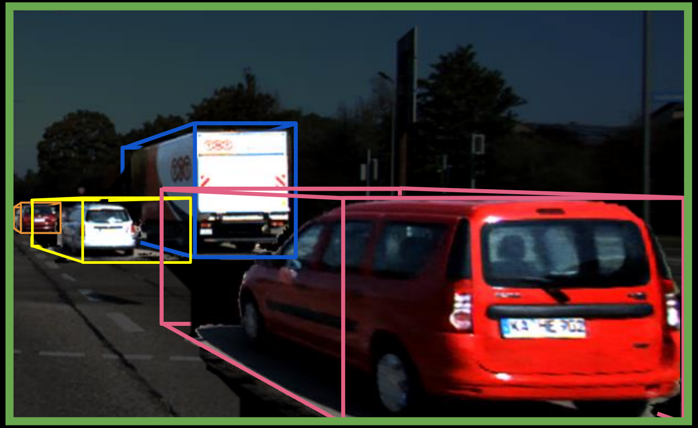
$$\mathbf{c}(\mathbf{x}, \mathbf{l}_o, \mathbf{p}_o) = F_{\theta_{c,2}}(\gamma_d(\mathbf{d}), \mathbf{y}(\mathbf{x}, \mathbf{l}_o), \mathbf{p}_o)$$

Background Plane Representation

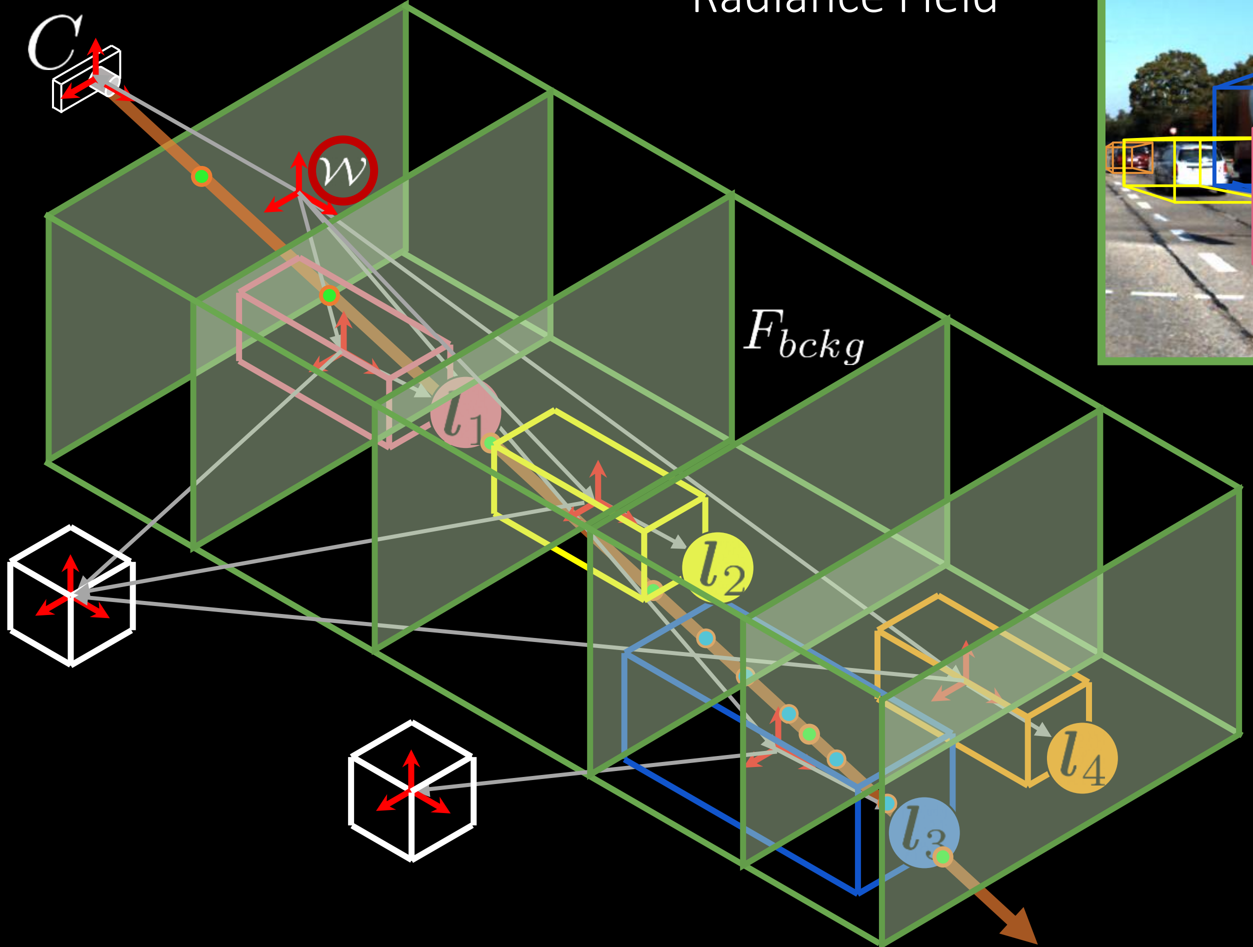


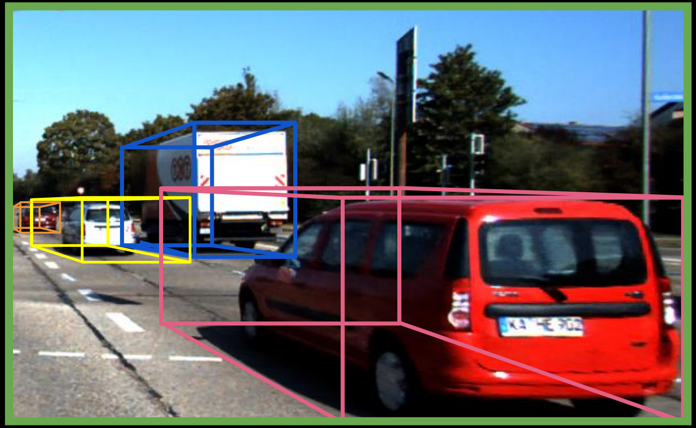
Background Sampling Points





Background Radiance Field





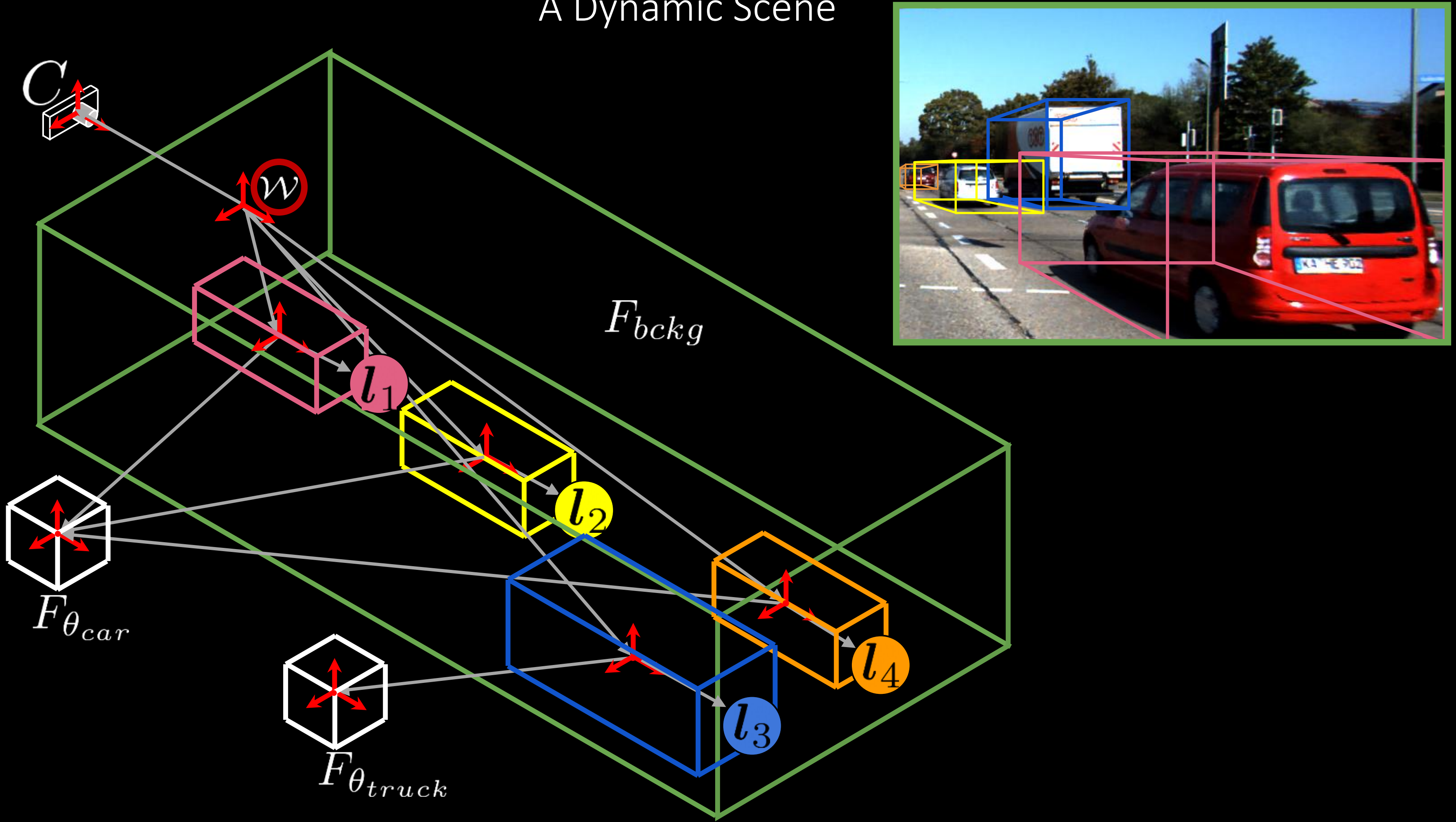
$$F_{\theta_{bckg}} : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

MLP, 2 Stages:

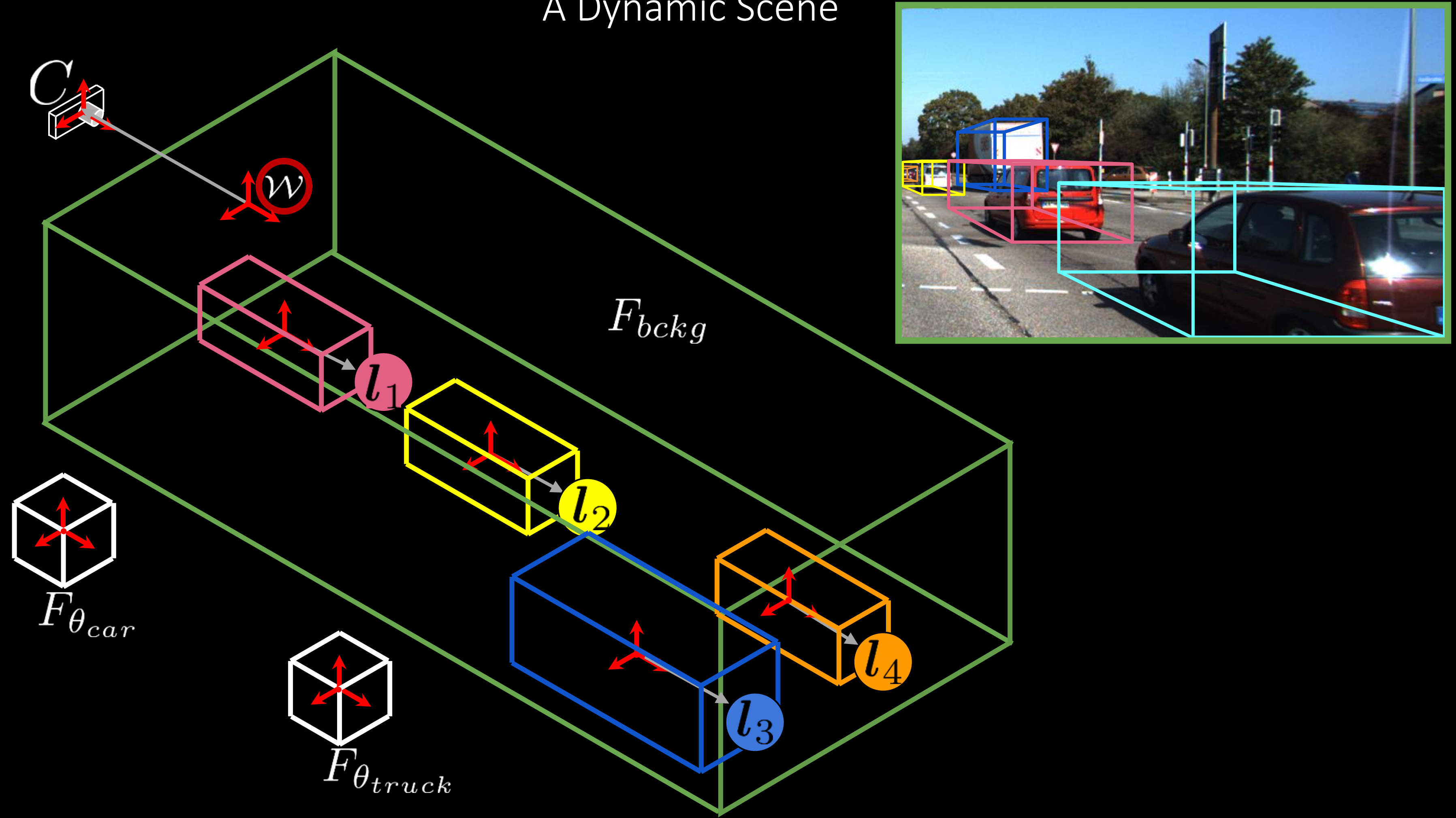
$$[\mathbf{y}(\mathbf{x}), \sigma(\mathbf{x})] = F_{\theta_{bckg,1}}(\gamma_x(\mathbf{x}))$$

$$\mathbf{c}(\mathbf{x}) = F_{\theta_{bckg,2}}(\gamma_d(\mathbf{d}), \mathbf{y}(\mathbf{x}))$$

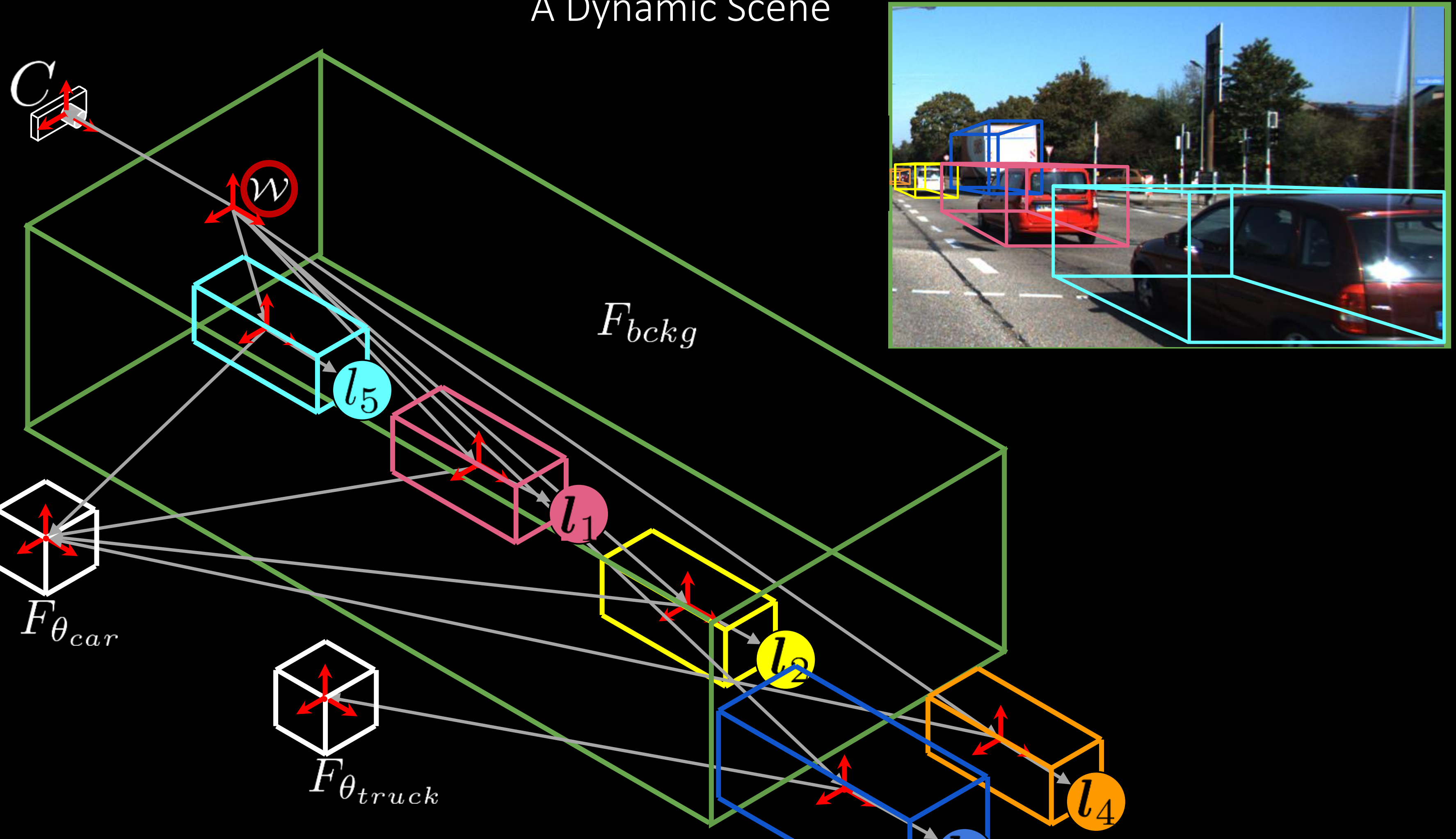
A Dynamic Scene



A Dynamic Scene



A Dynamic Scene



Scene Manipulation

Rotation

Translation

Scene Manipulation – Global Illumination Effects

Scene Manipulation – Global Illumination Effects

Camera Movement

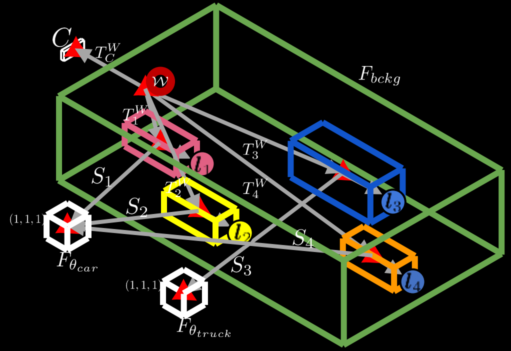
Reconstruction of Dynamic Scene

Objects fixed at t = 0.25

Objects fixed at t = 0.5

Objects fixed at t = 0.75

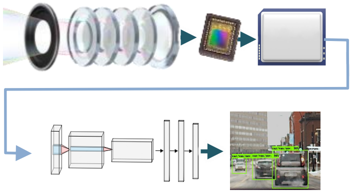
Neural Scene Graphs for Inference



Object Detection via Inverse Rendering

Neural Representation

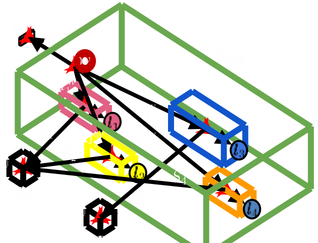
Robust Computational Imaging and Vision without Labeling



End-to-end Cameras

Novel Robust Sensors

Vision in Scattering Media



Scene Representations
for Inference

light.princeton.edu

Computational imaging for
robustness without supervision:

- **Robust perception, fusion** and **depth**
- **"Super-human vision"**
- **Co-design** of sensors + algorithms