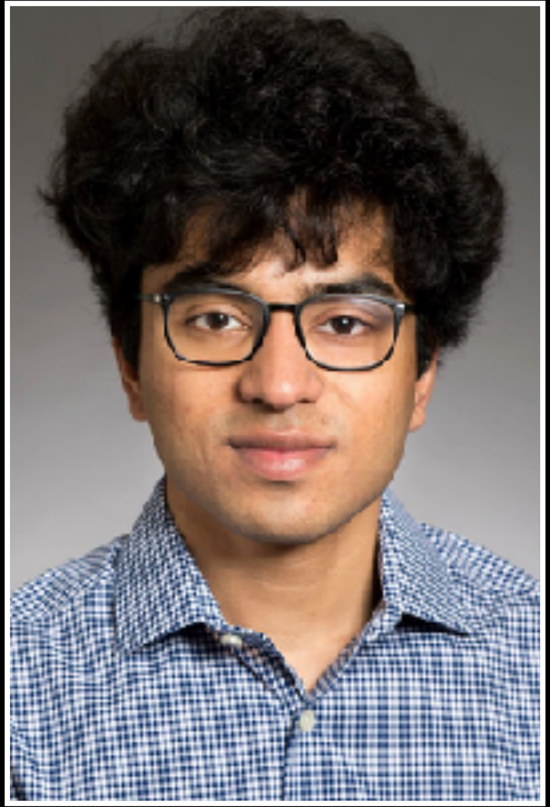
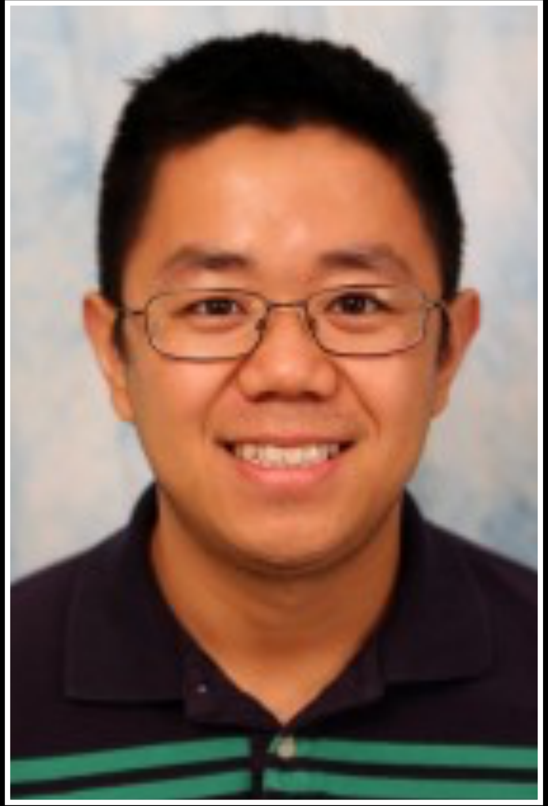
Active Perception using Light Curtains for Autonomous Driving

Webpage: http://siddancha.github.io/projects/active-perception-light-curtains


Siddharth Ancha


Yaadhav Raaj


Peiyun Hu


Srinivasa Narasimhan

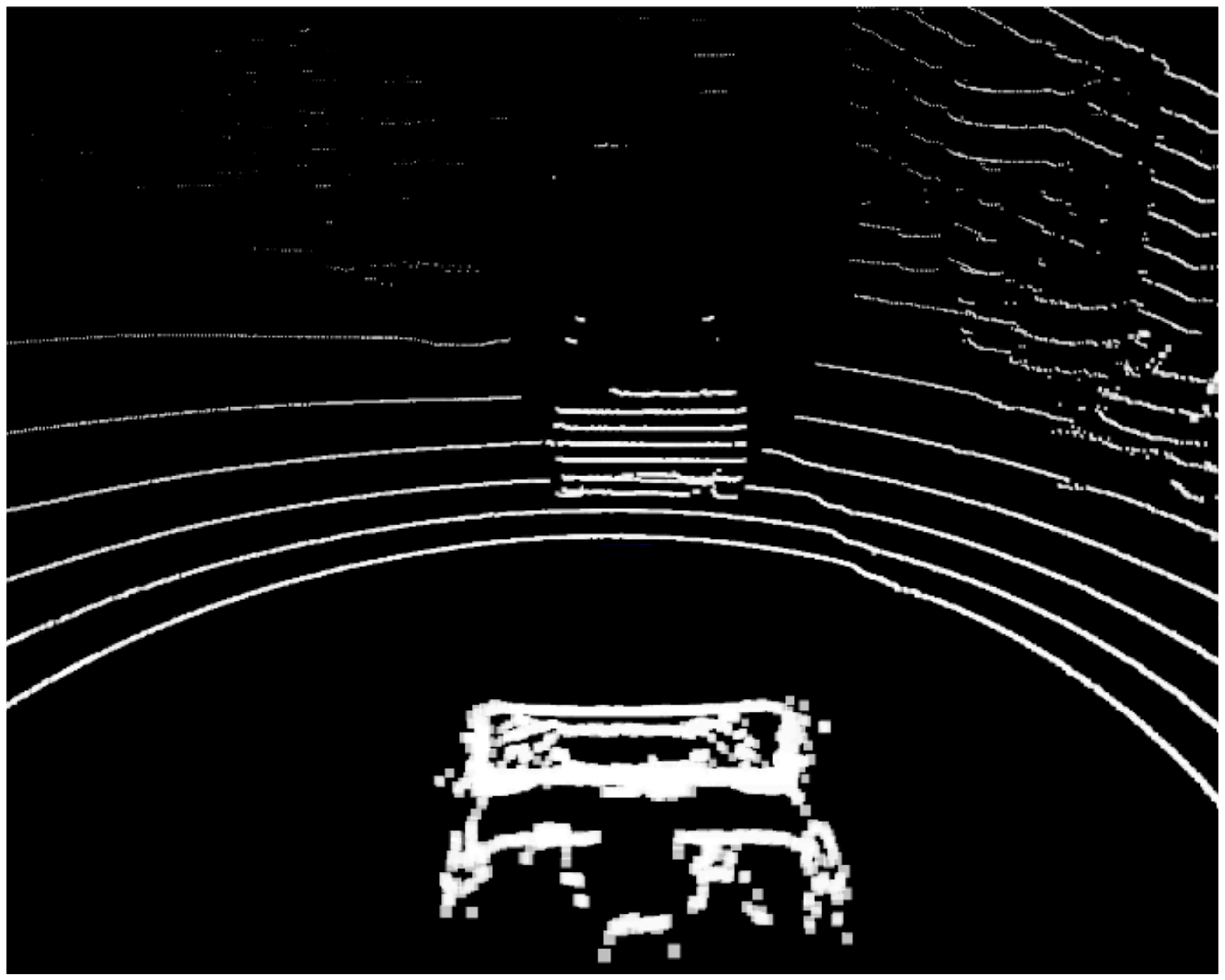

David Held

Carnegie Mellon University

THE ROBOTICS INSTITUTE

ML MACHINE LEARNING Carnegie Mellon University

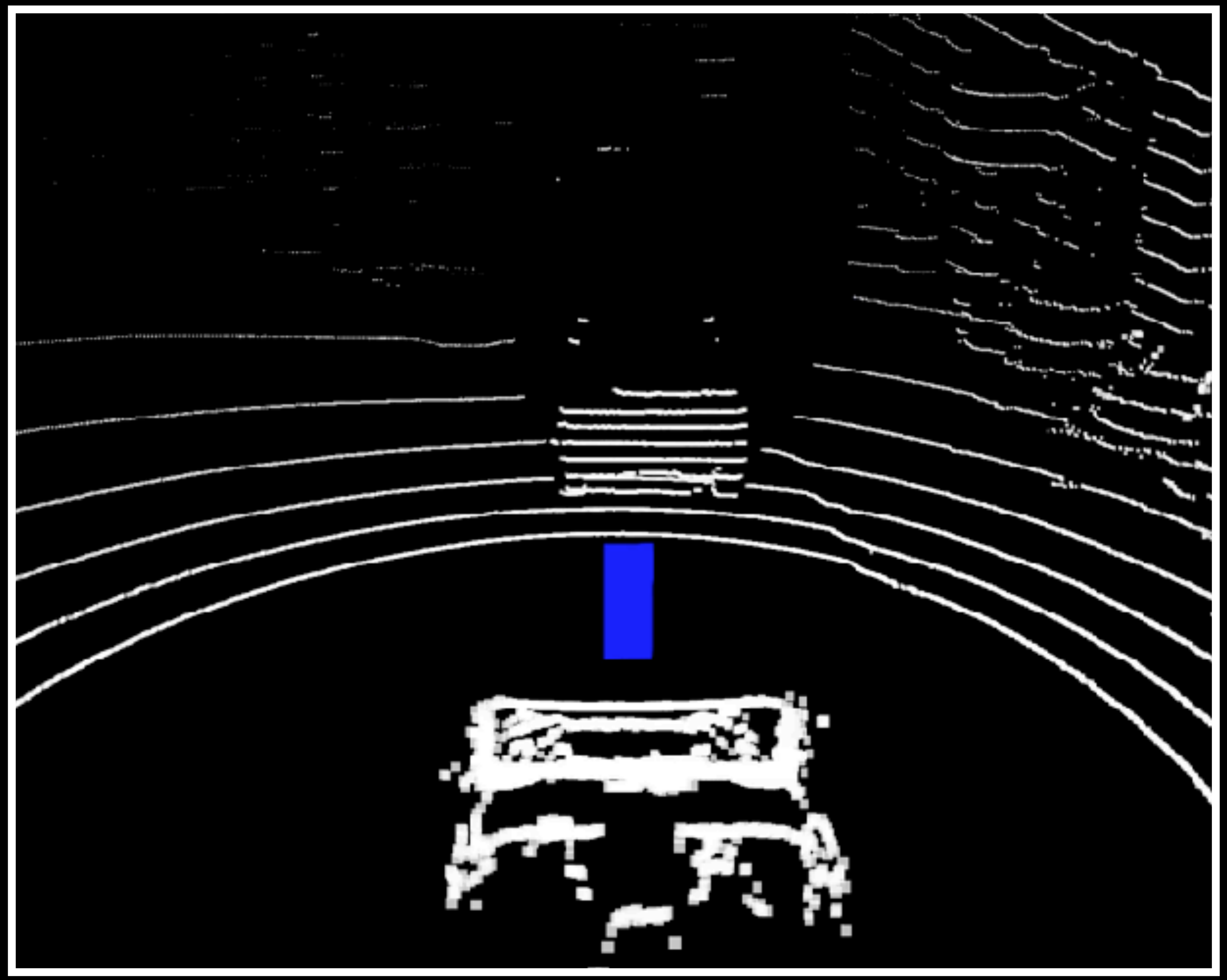
LiDARs perform fixed scans



LiDAR

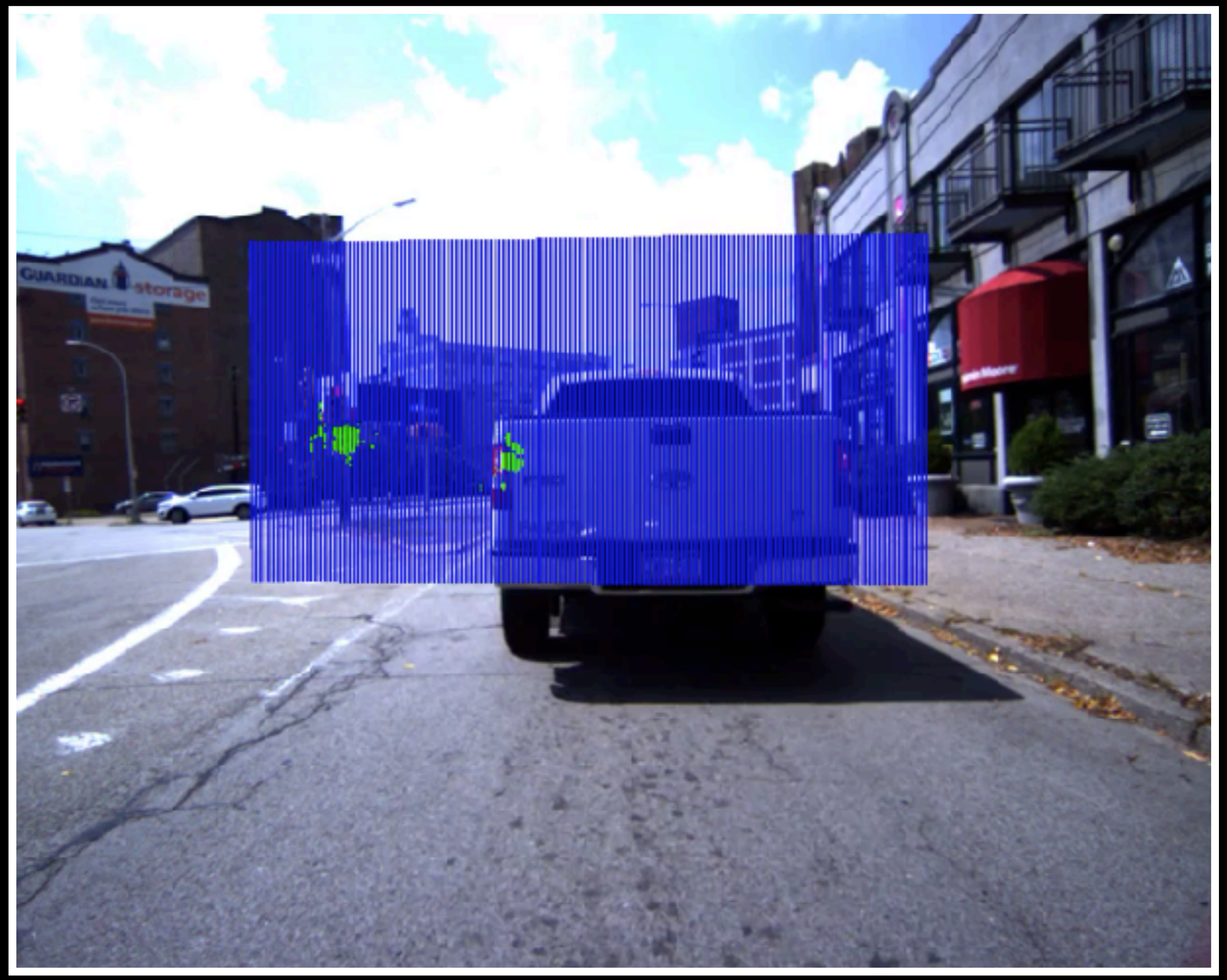
- Sparse point clouds.
- Expensive: Velodyne 64 beam LiDAR can cost > $80,000.

Light curtains are *controllable*



LiDAR

- **Sparse point clouds.**
- **Expensive: Velodyne 64 beam LiDAR can cost > $80,000.**

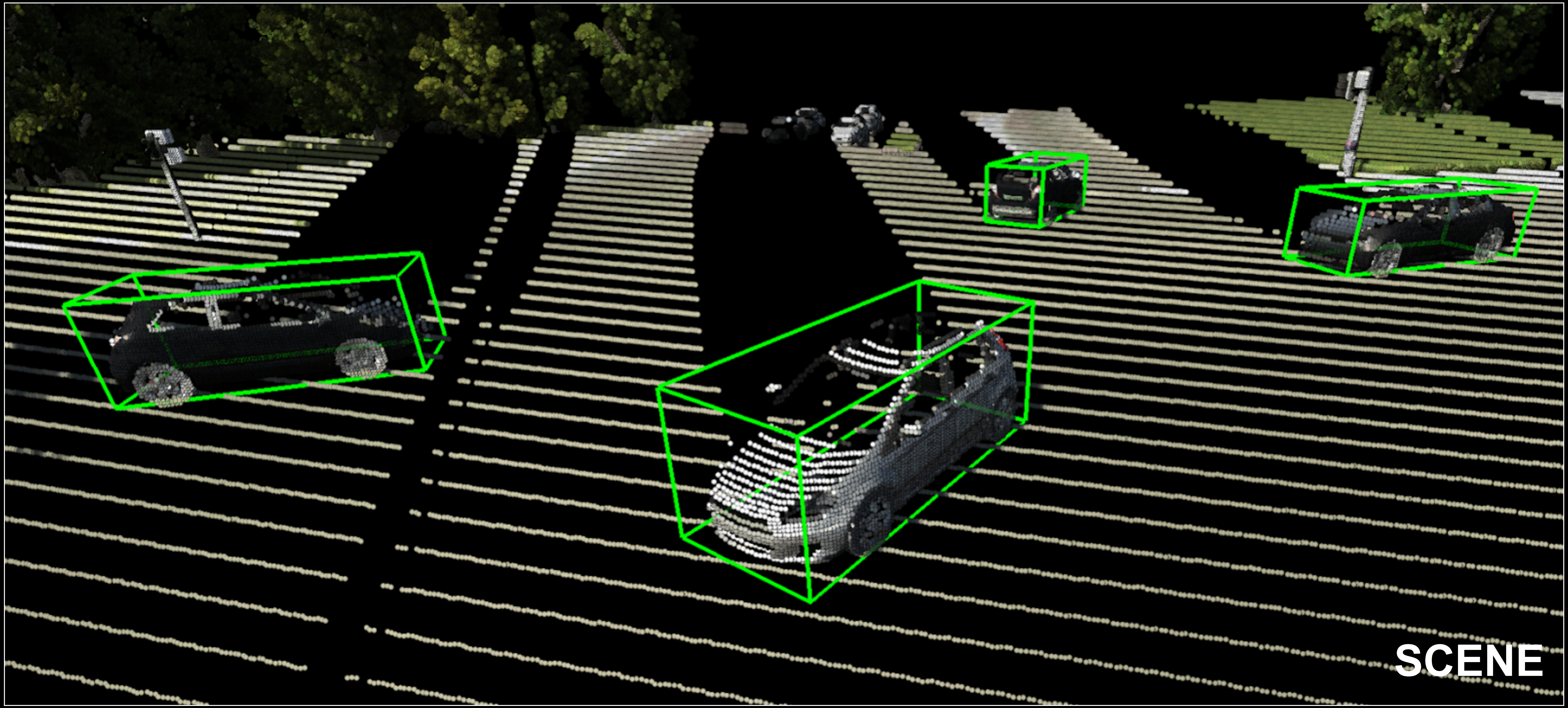


Light Curtain

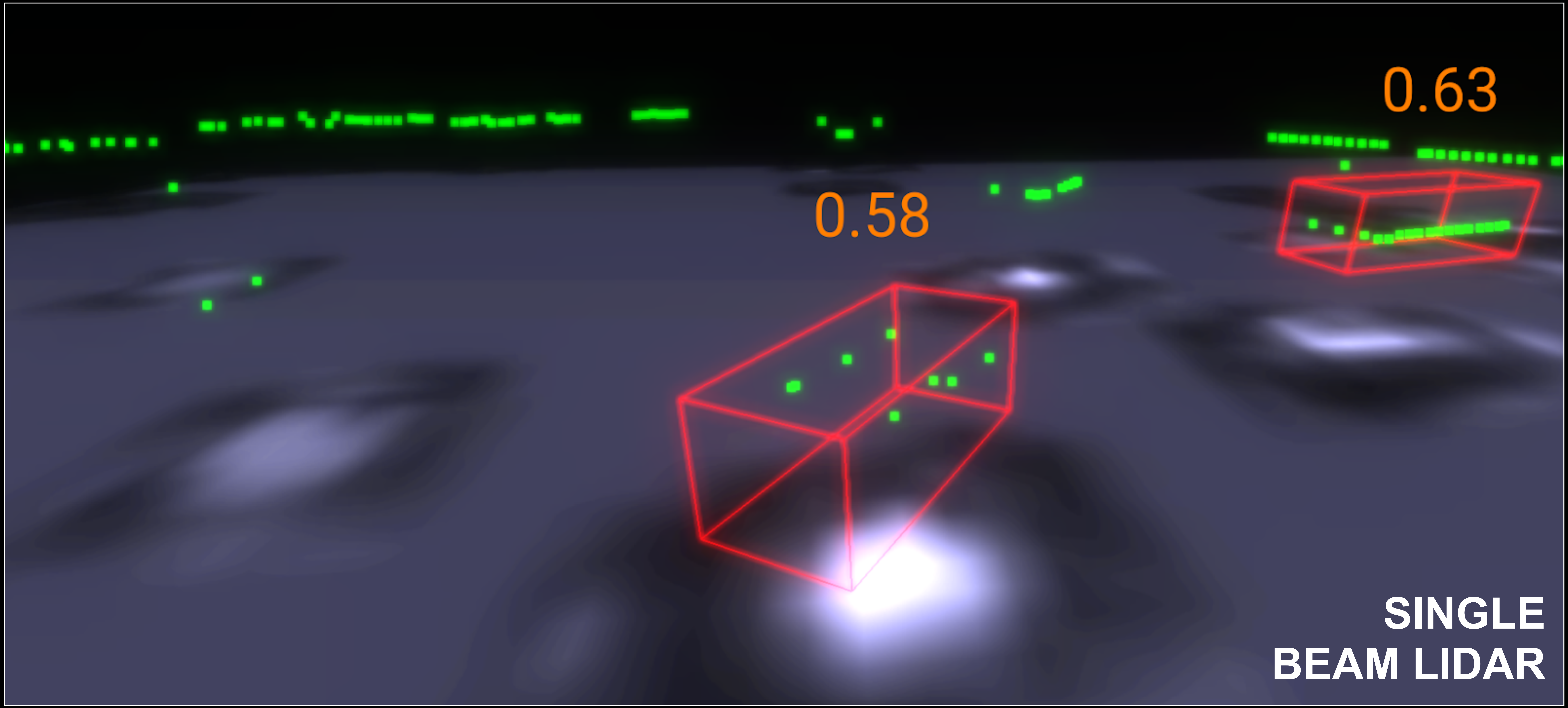
- **Dense point cloud where curtain is placed.**
- **Inexpensive:**
  Lab-built prototype costs ~$1000.

**Agile Depth Sensing Using Triangulation Light Curtains,* Bartels et. al. 2019

Active Detection using Light Curtains

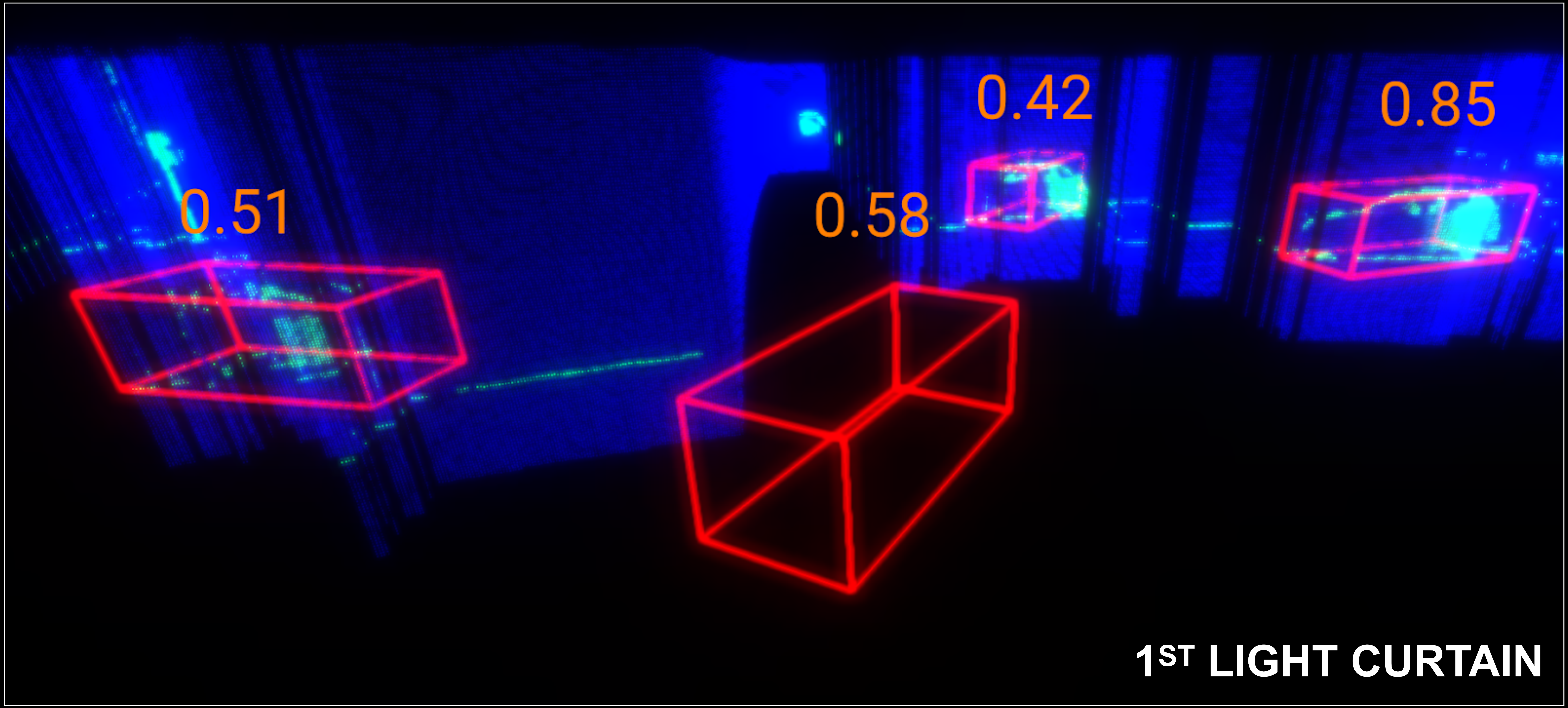


Single-beam LiDAR produces sparse points

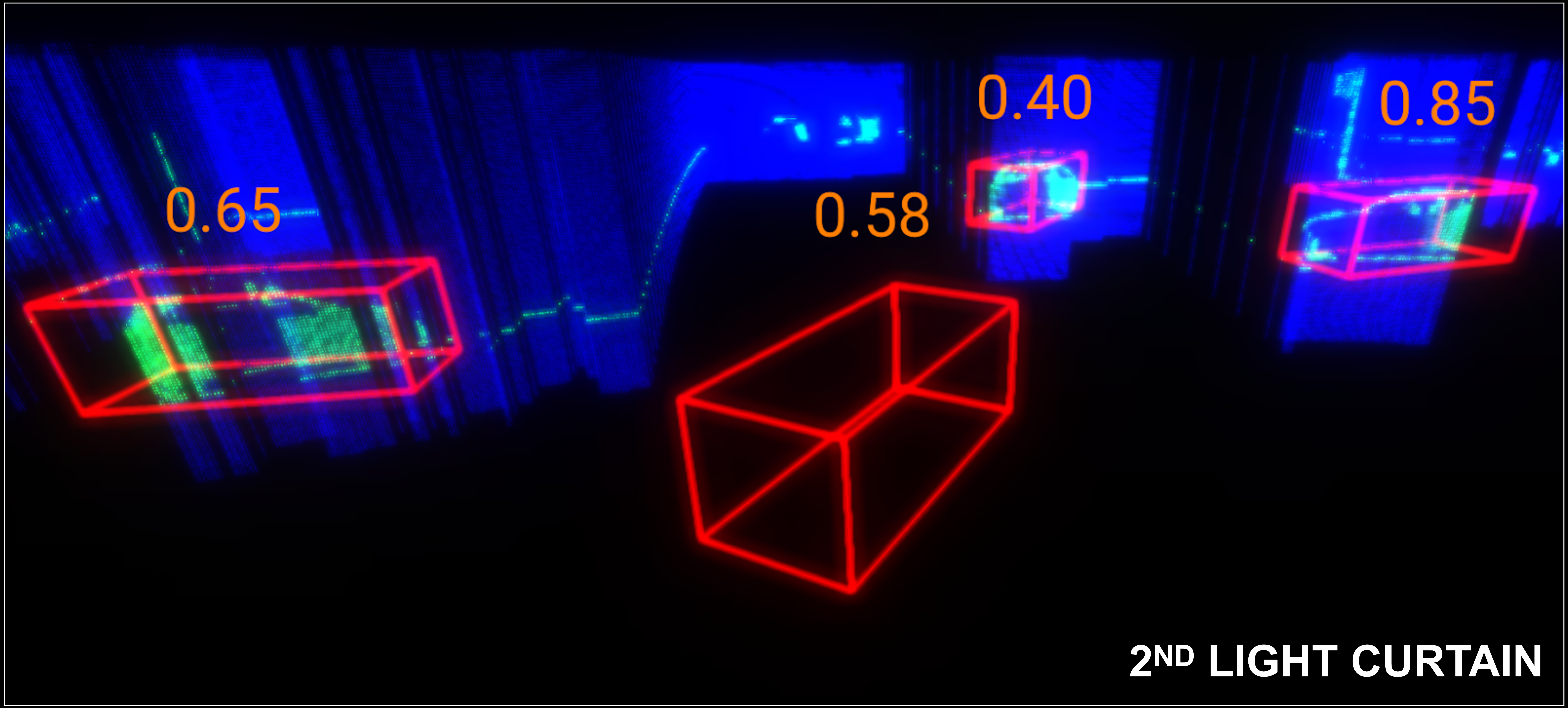


*White: more uncertain   Black: less uncertain

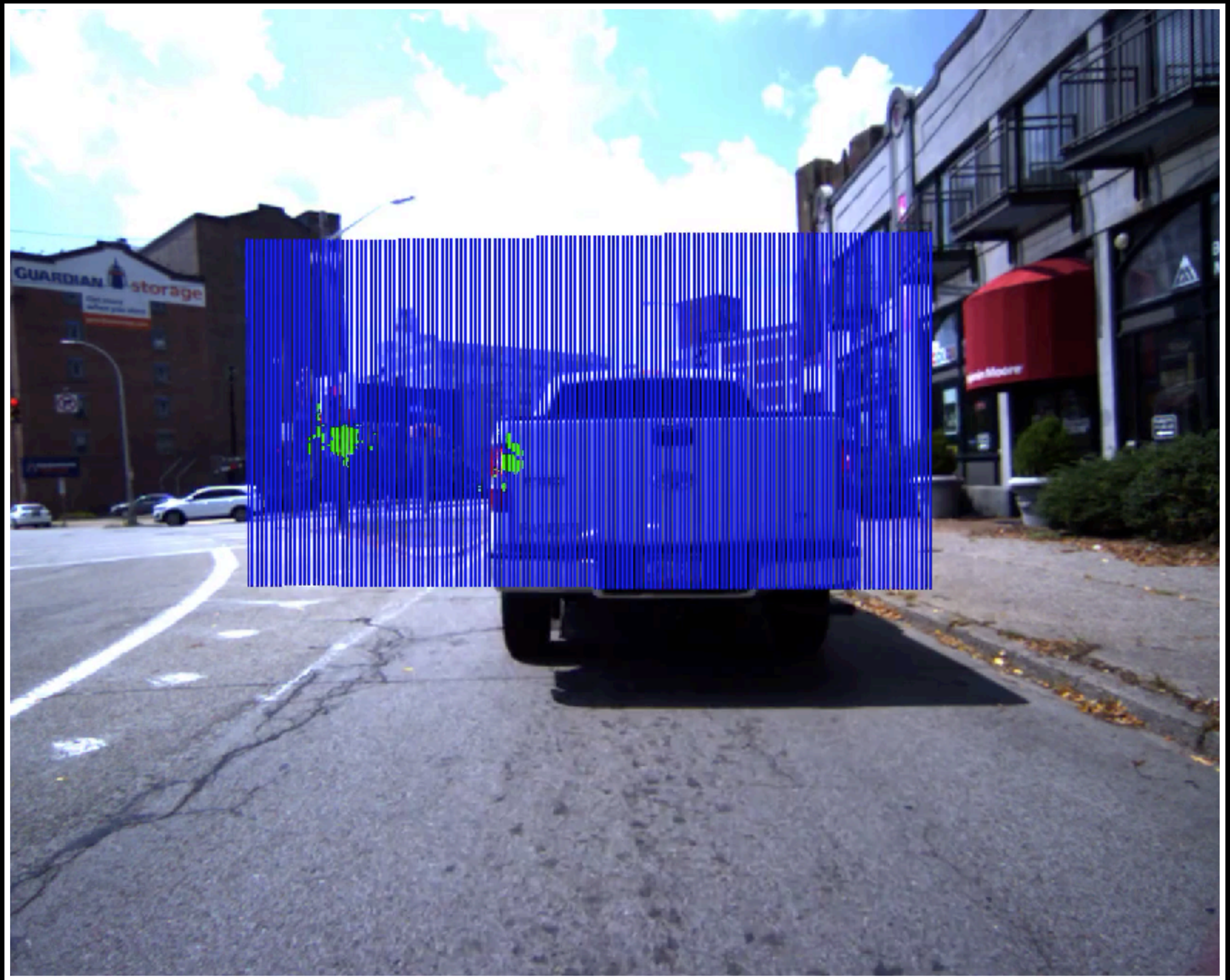
Light curtain improves detection



Light curtain improves detection



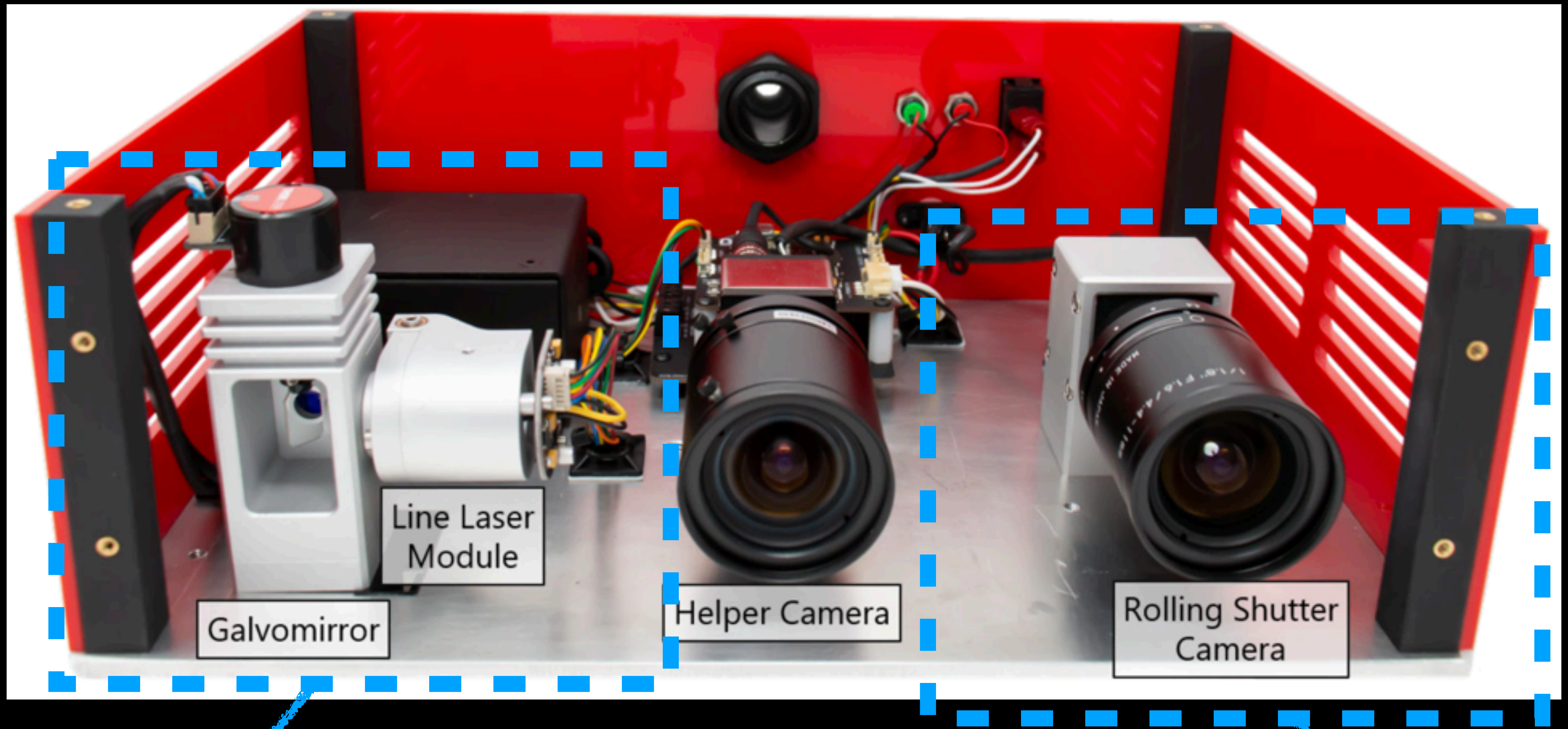
Background: what are light curtains?



“Agile Depth Sensing Using Triangulation Light Curtains”, Bartels et. al.

`http://www.cs.cmu.edu/~ILIM/agile_depth_sensing`

A light curtain has two major components

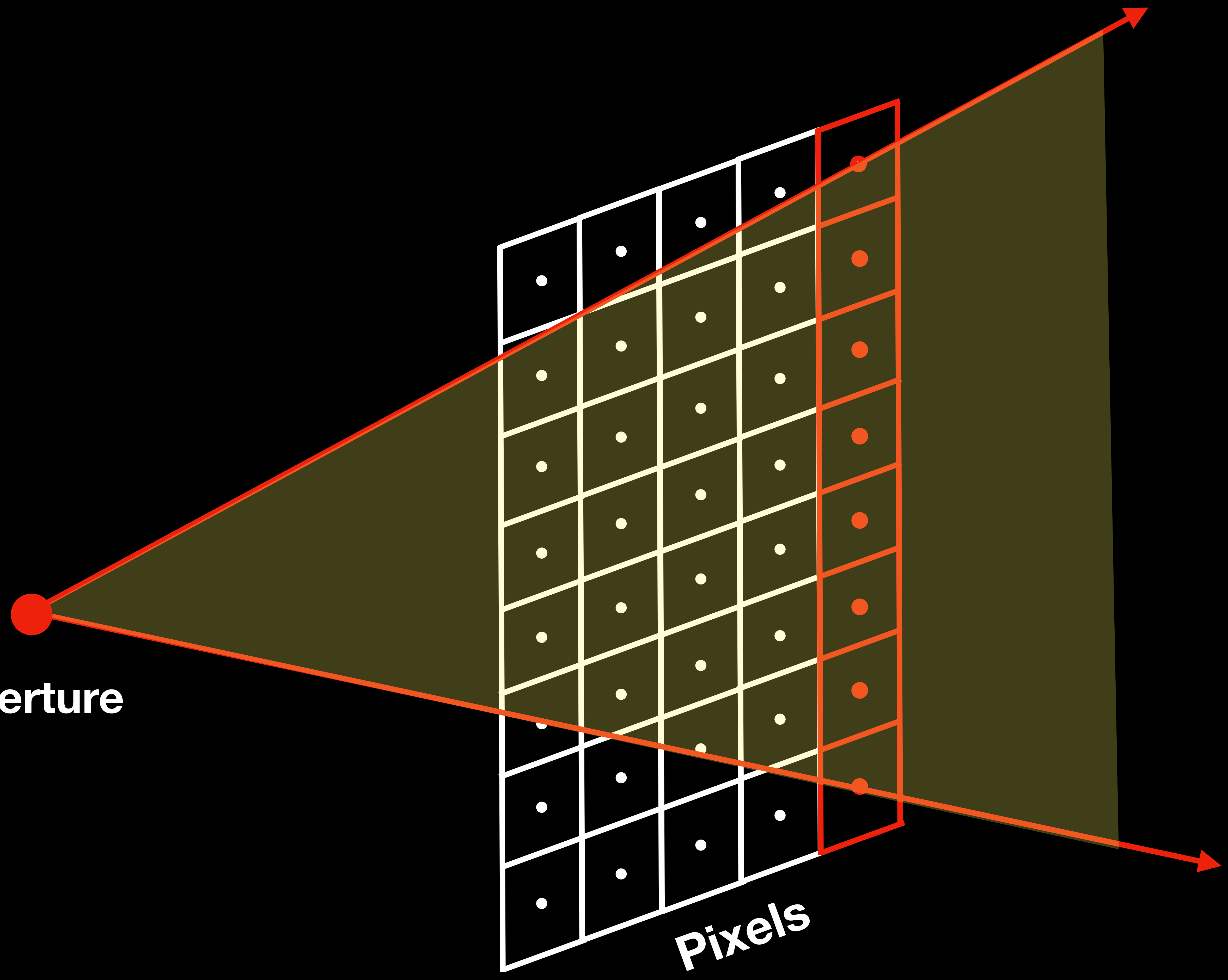


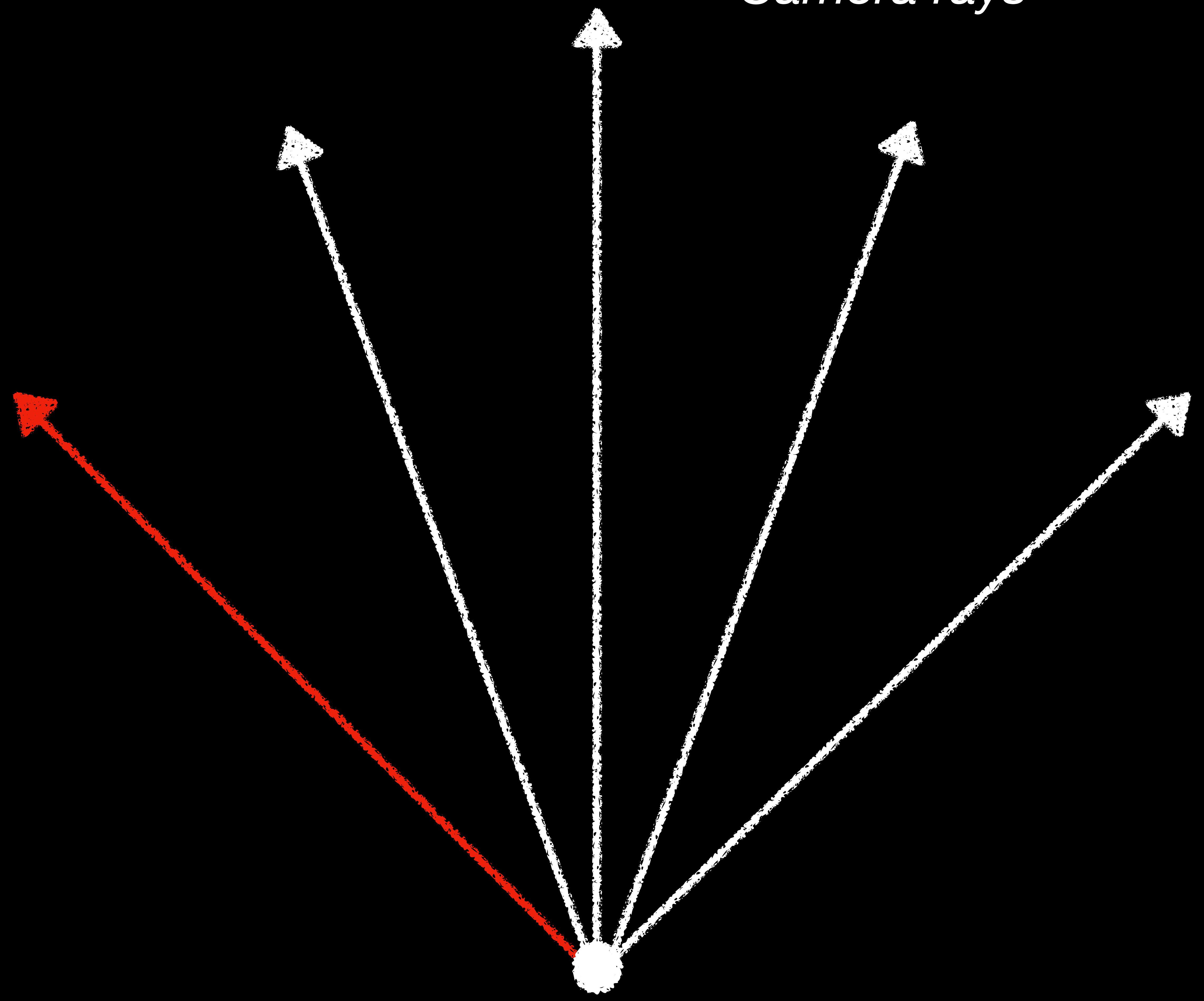
Laser **Camera**

"Agile Depth Sensing Using Triangulation Light Curtains", Bartels et. al.

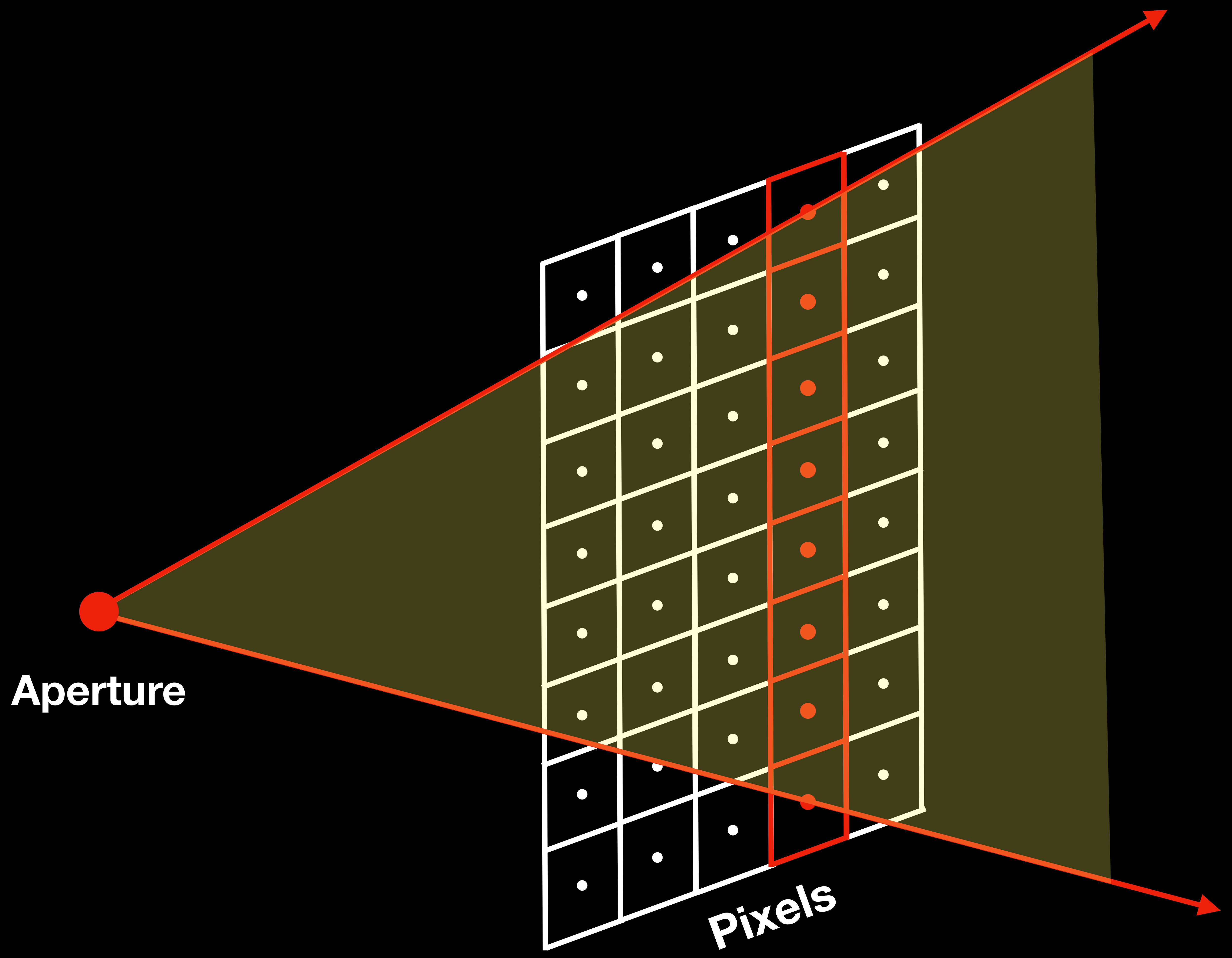
http://www.cs.cmu.edu/~ILIM/agile_depth_sensing

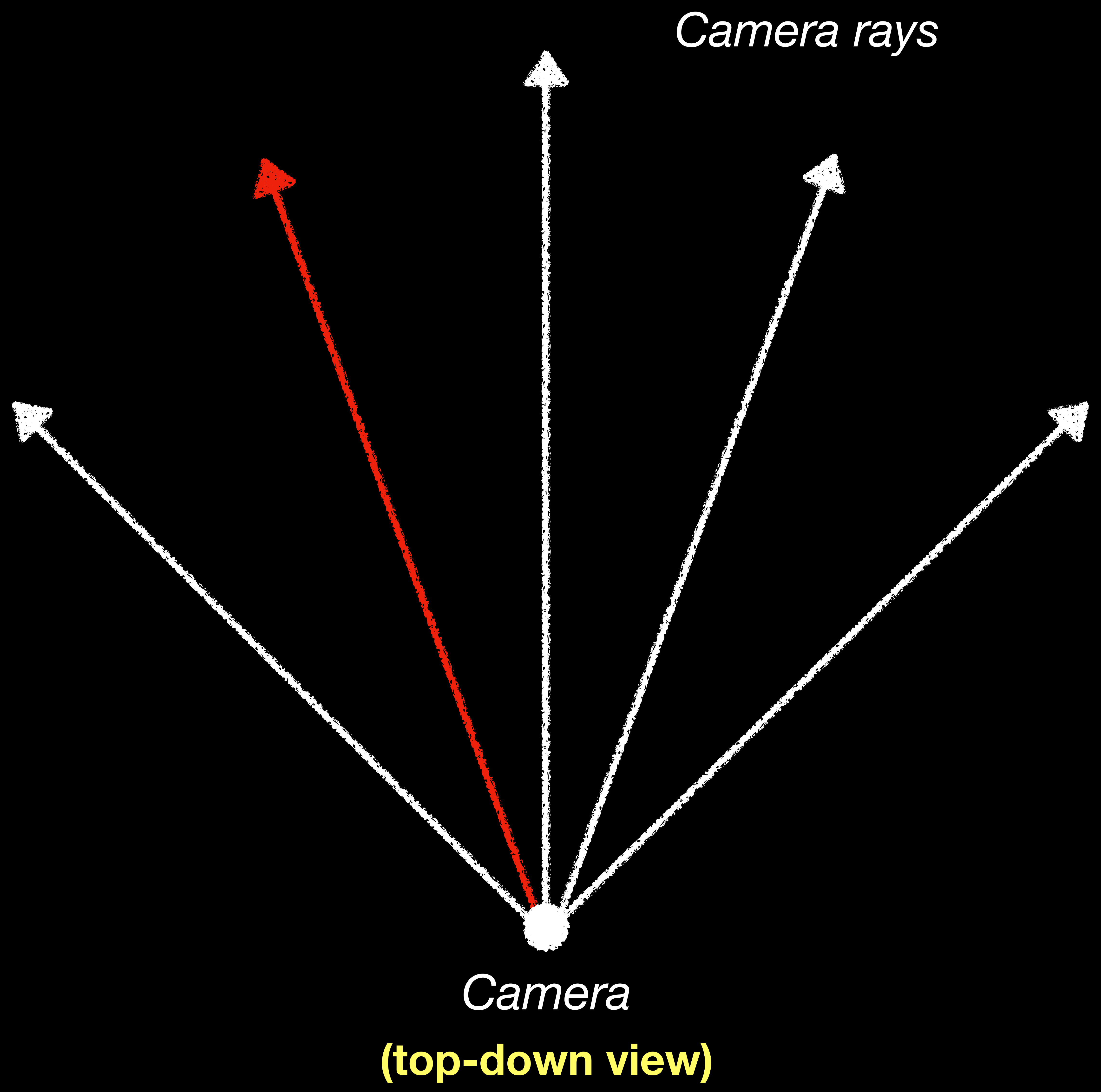
Rolling-shutter camera



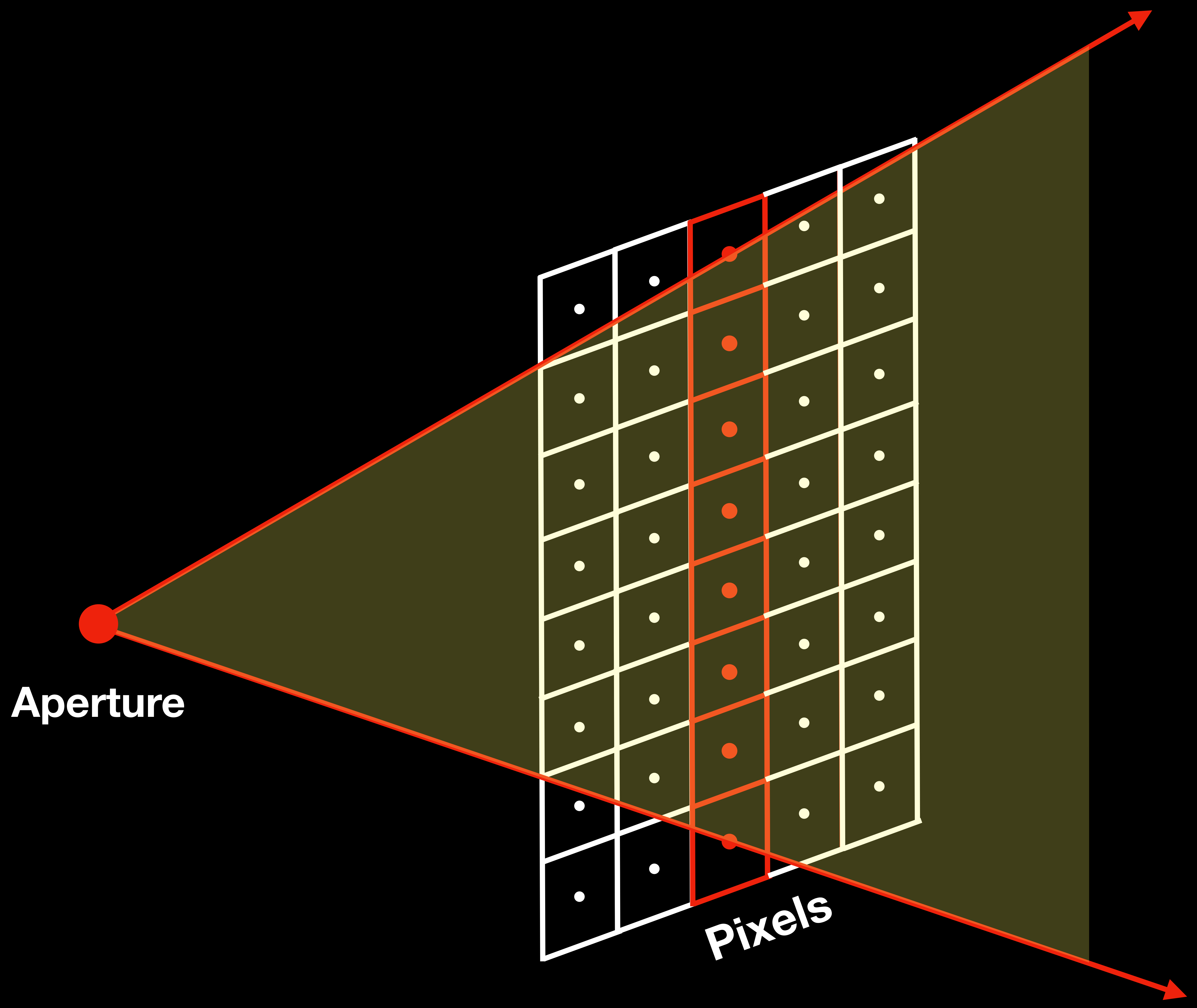


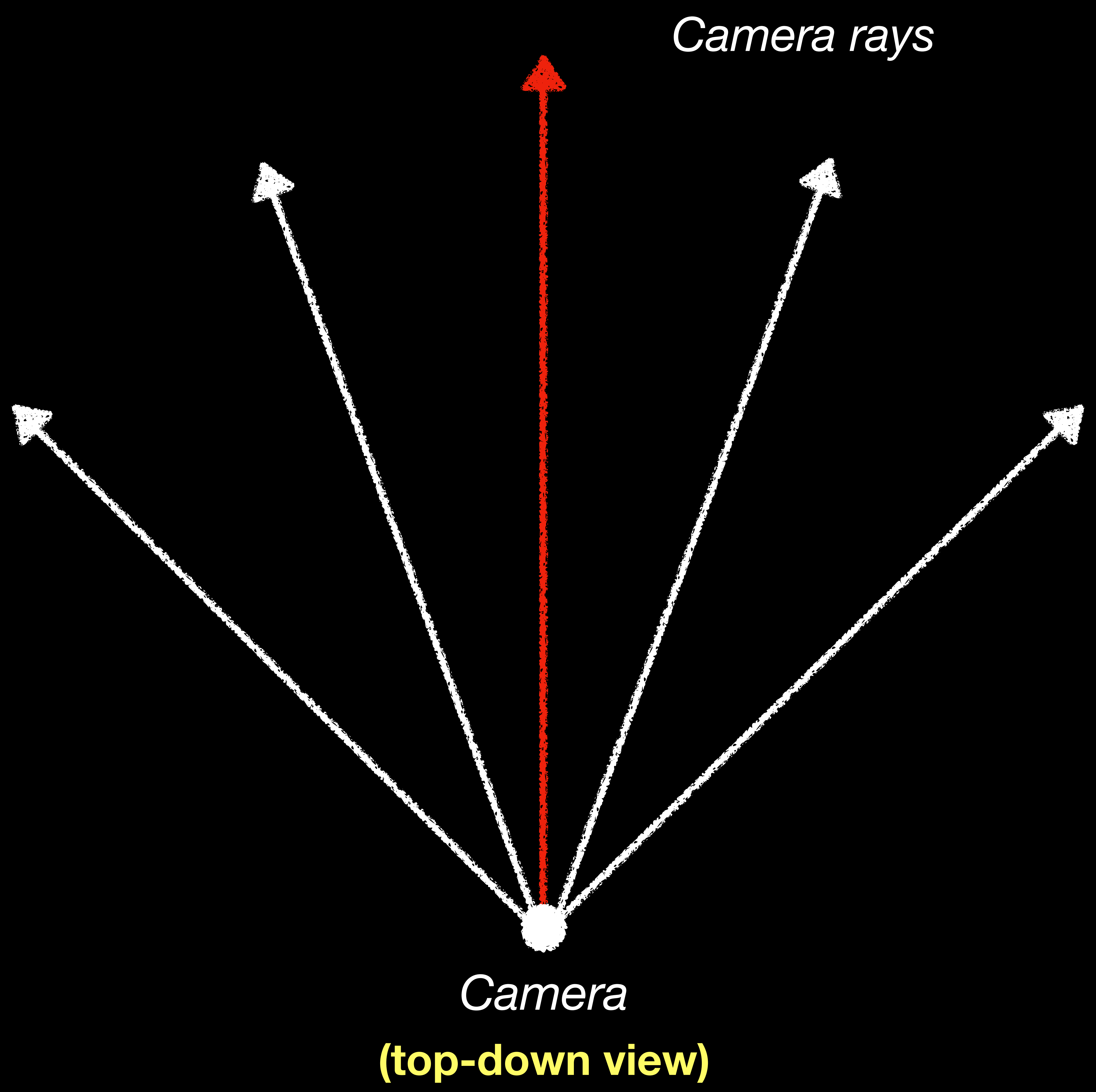
Rolling-shutter camera



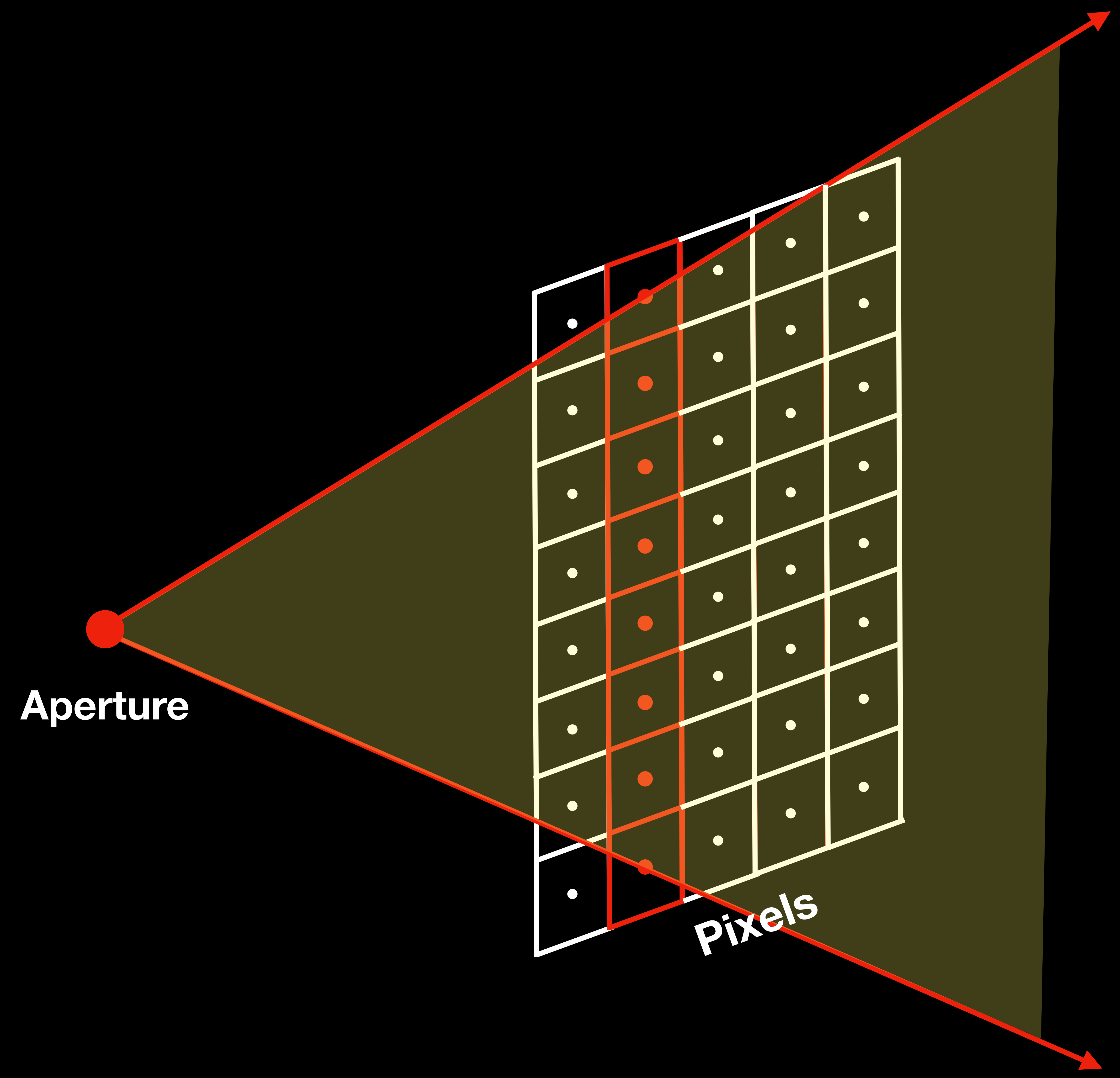


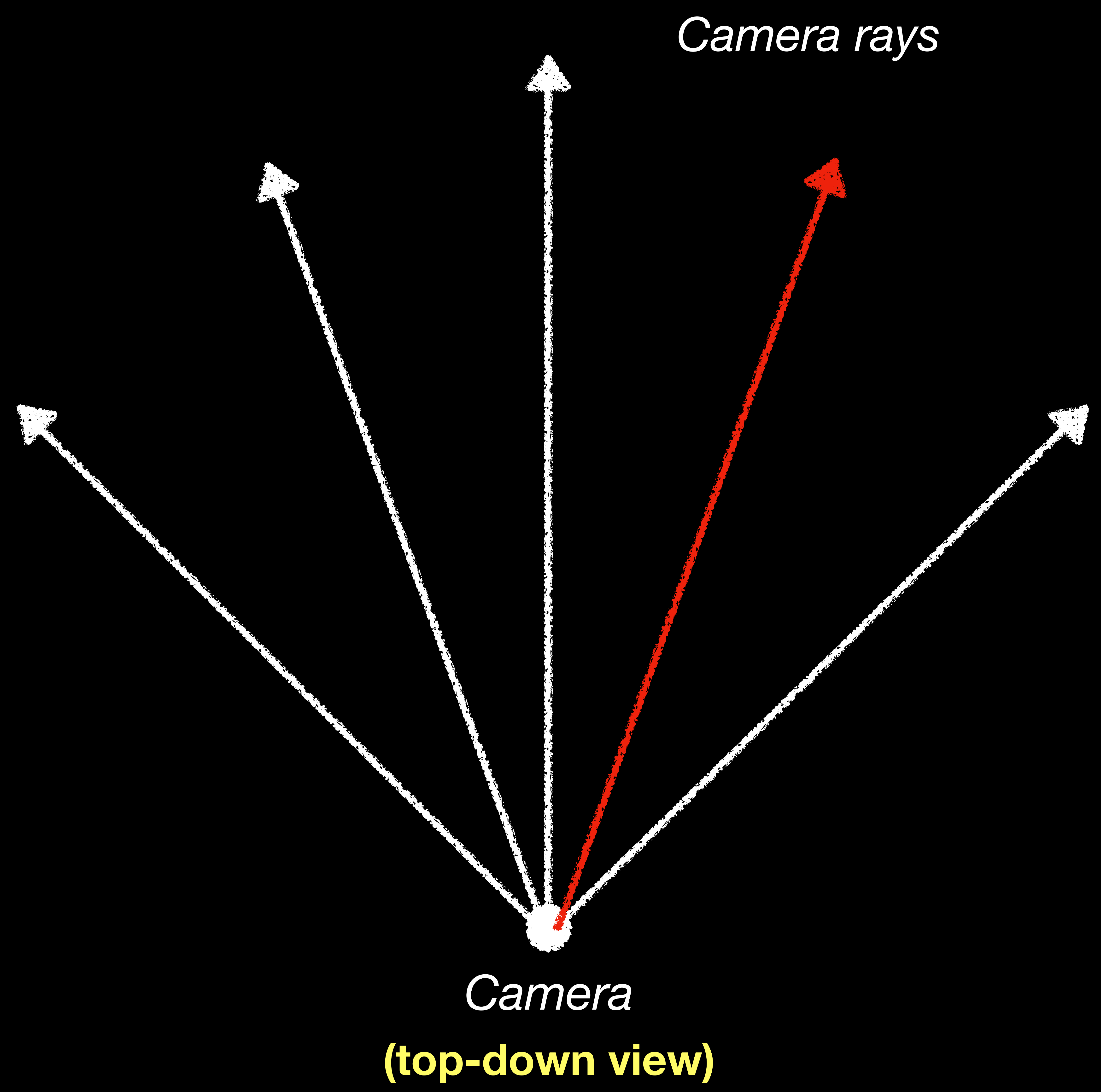
Rolling-shutter camera



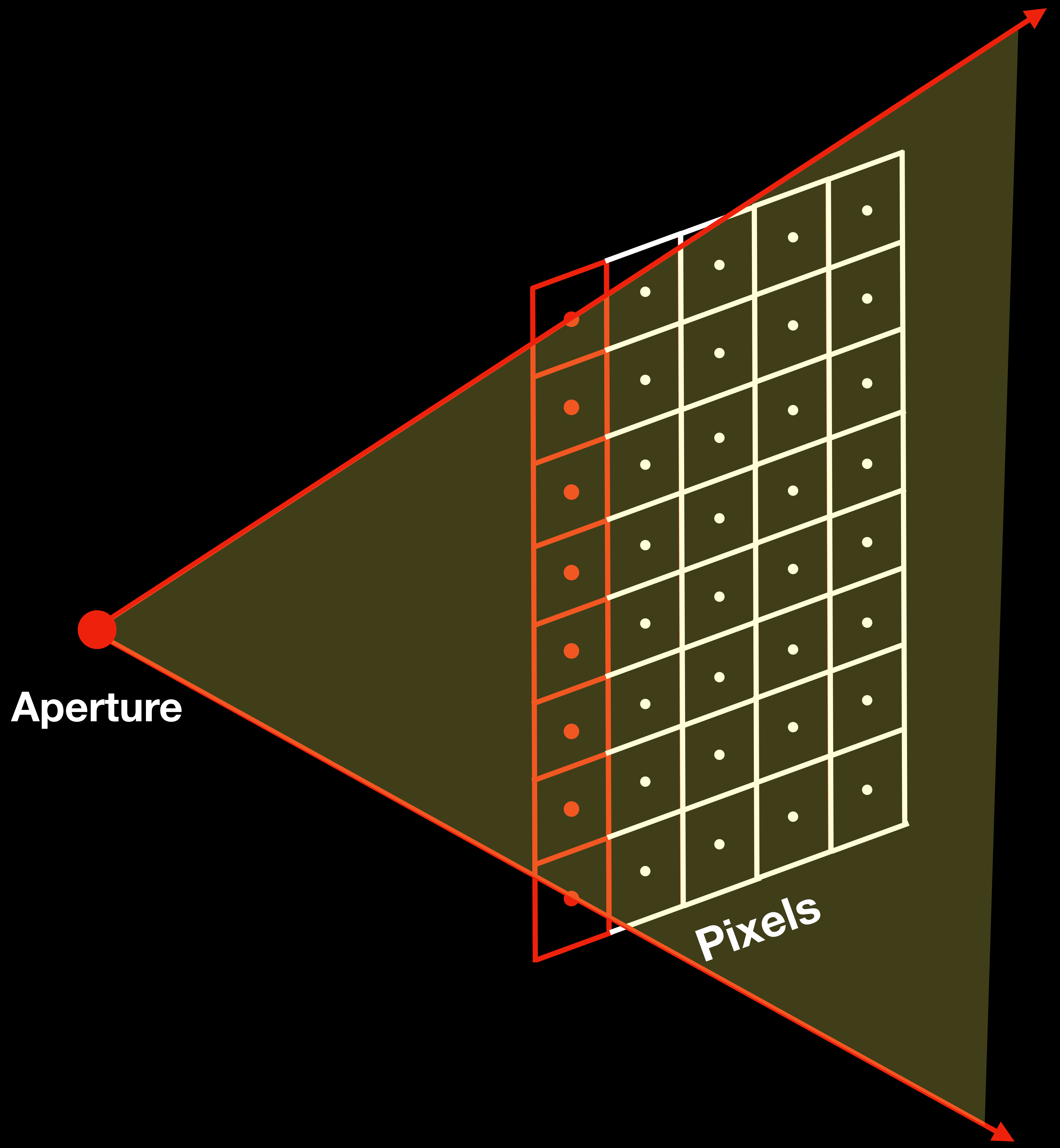


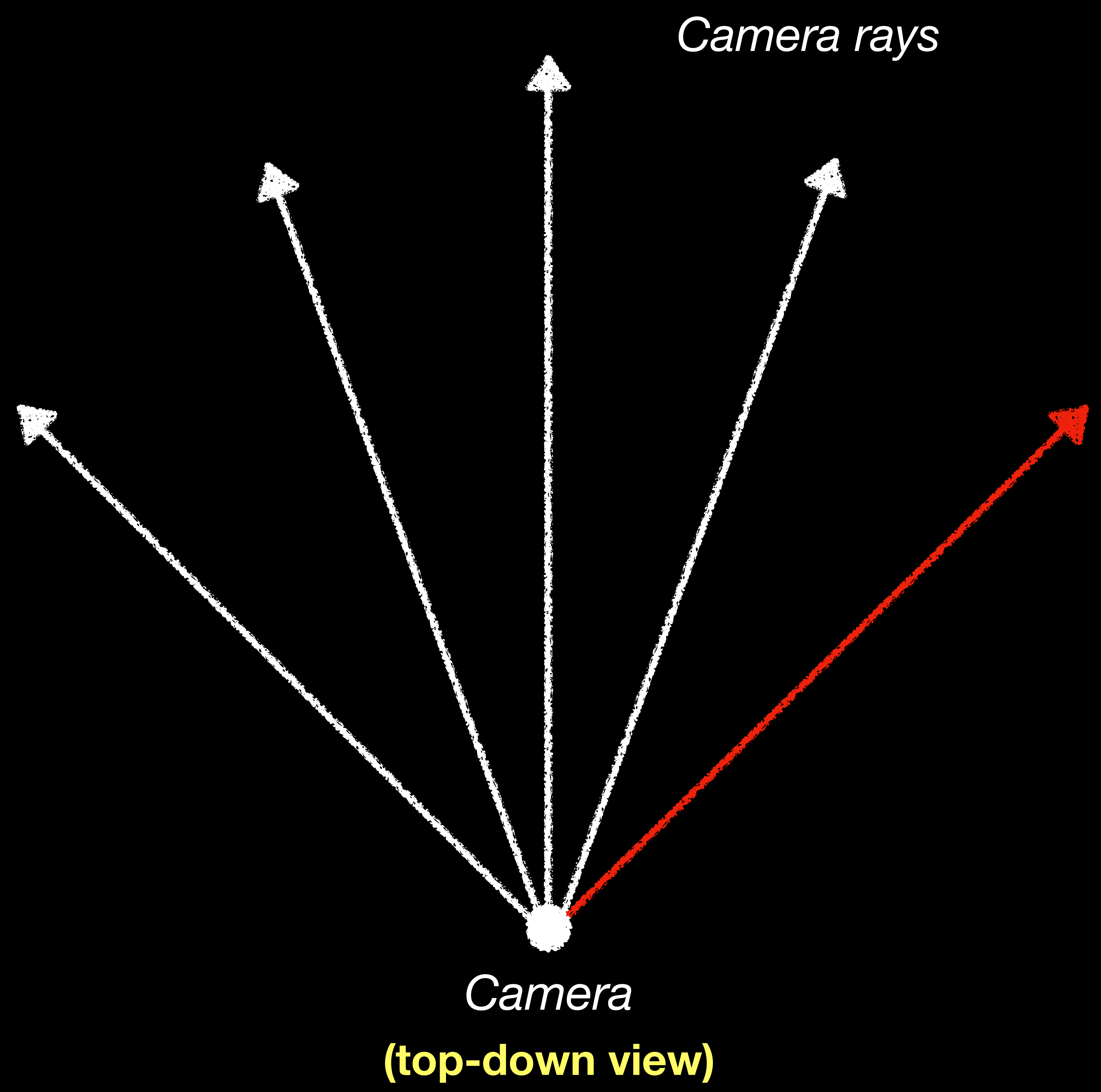
Rolling-shutter camera



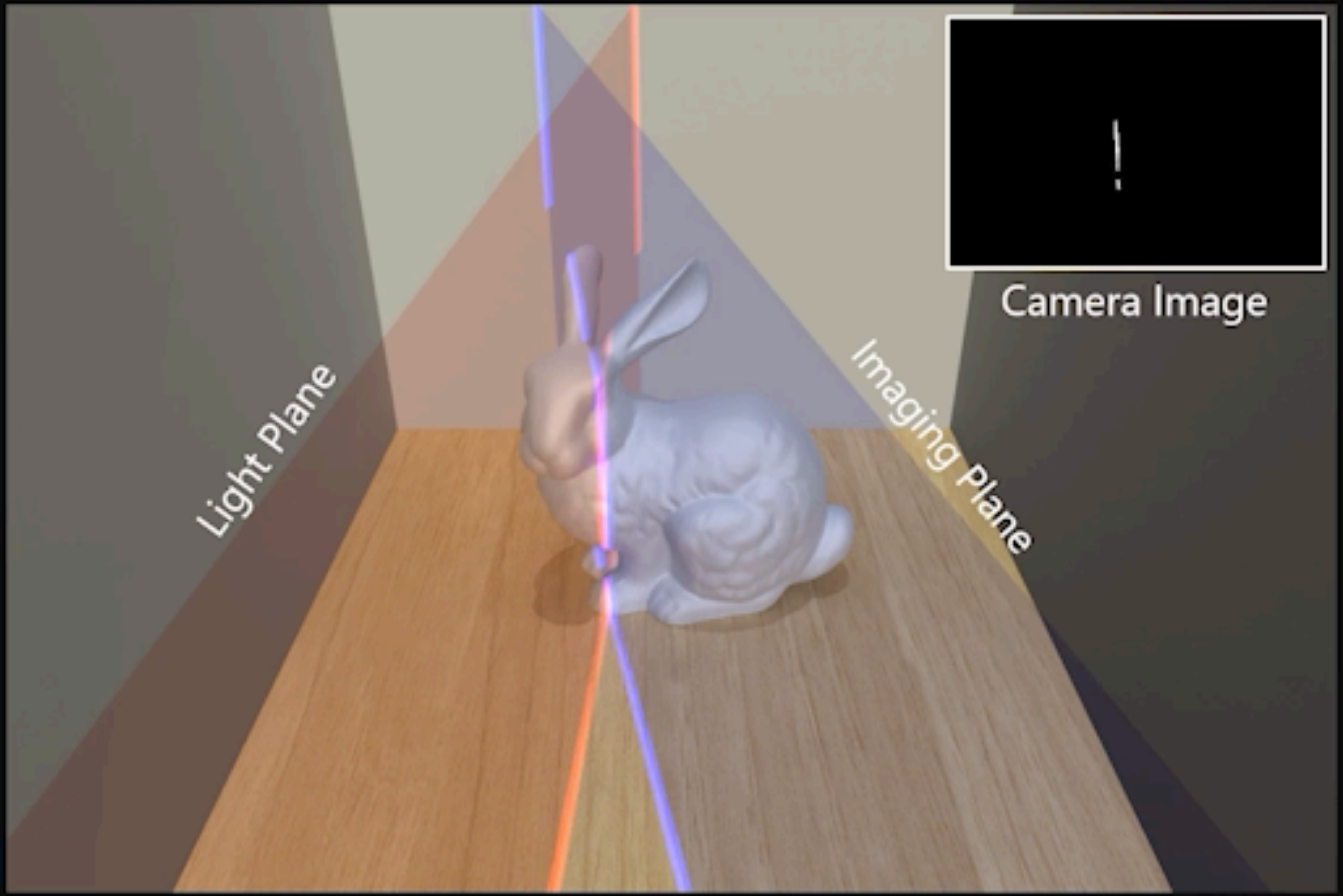


Rolling-shutter camera





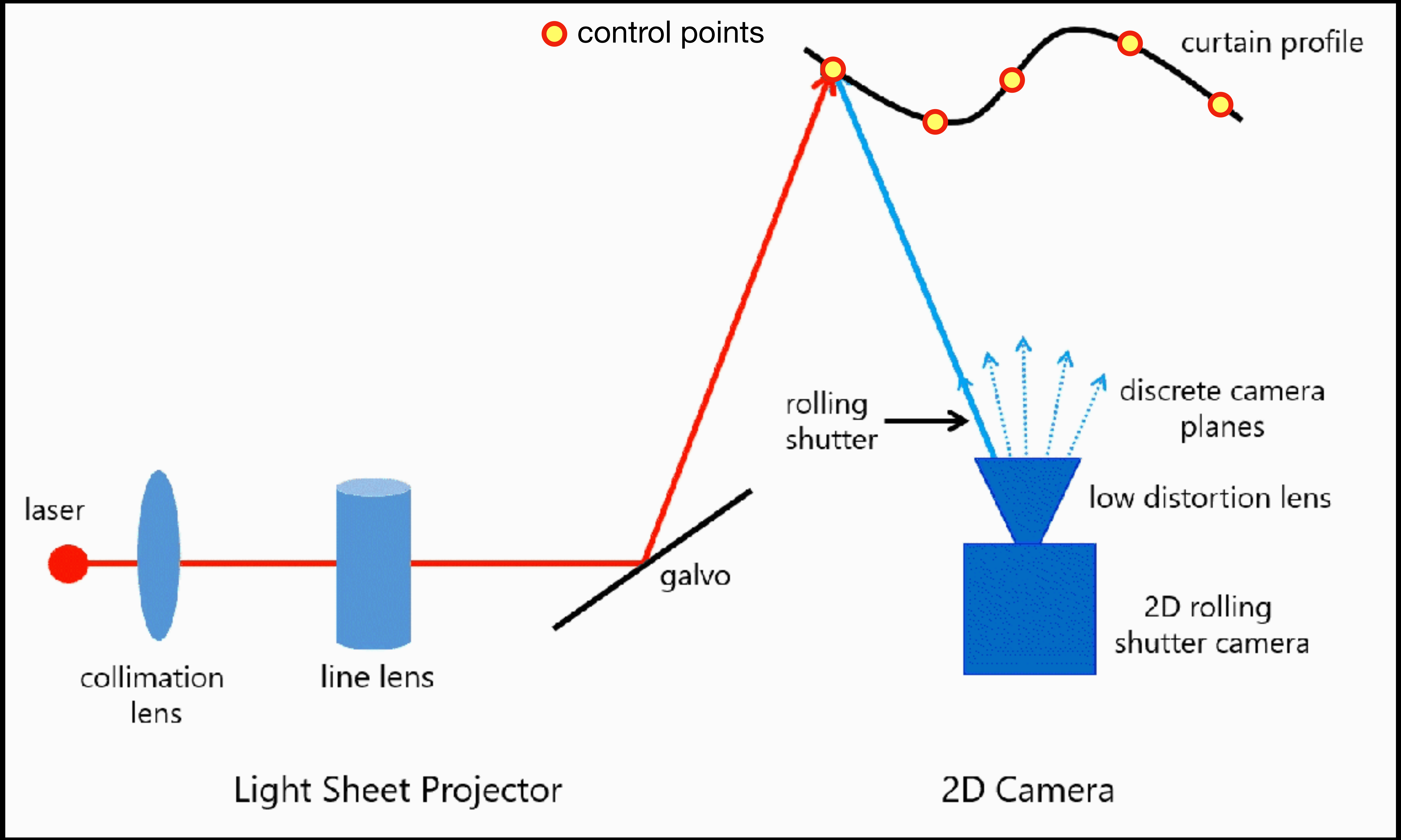
Programmable Light Curtain Principle



"Agile Depth Sensing Using Triangulation Light Curtains", Bartels et. al.

http://www.cs.cmu.edu/~ILIM/agile_depth_sensing

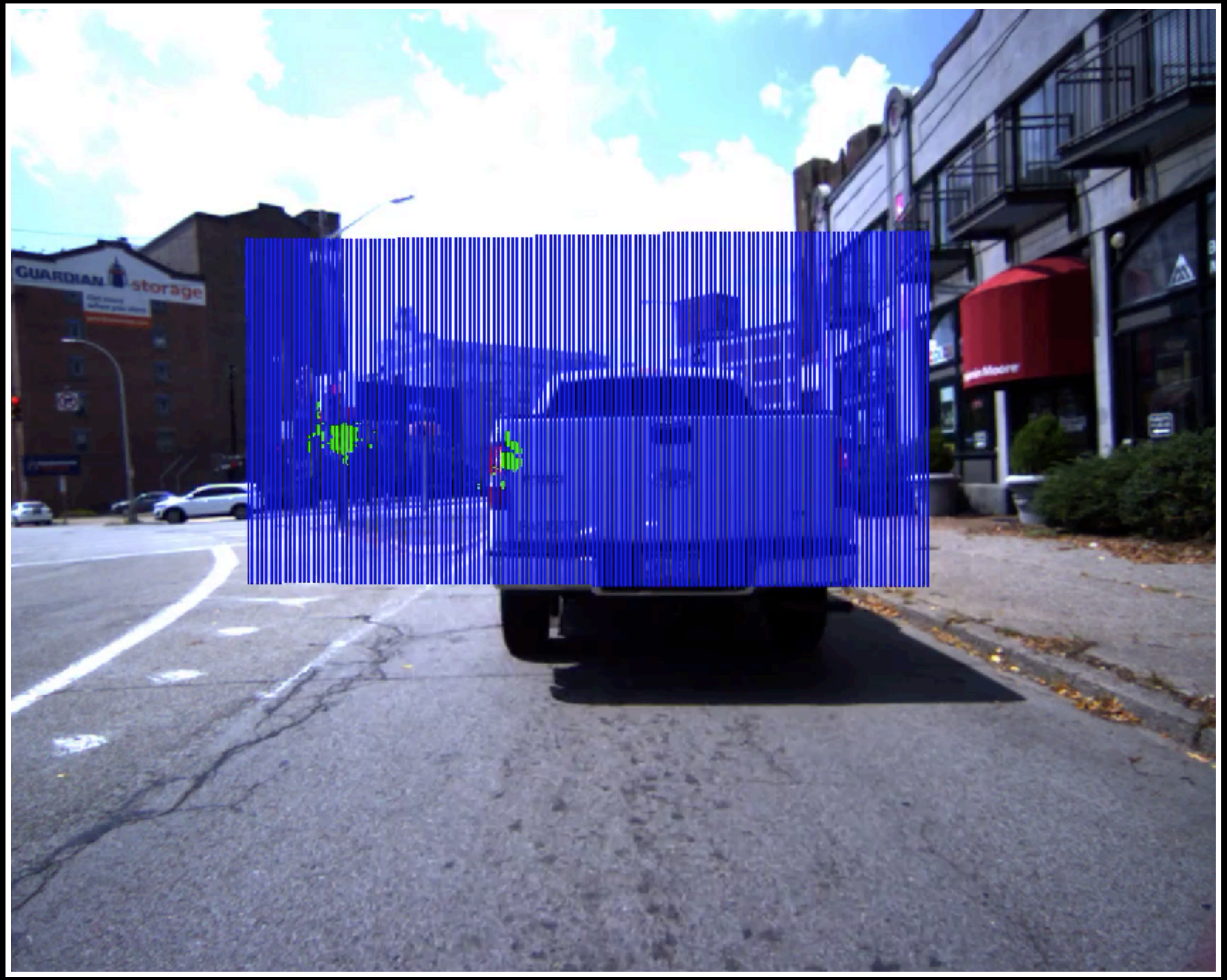
Light curtain working principle



"Agile Depth Sensing Using Triangulation Light Curtains", Bartels et. al.

http://www.cs.cmu.edu/~ILIM/agile_depth_sensing

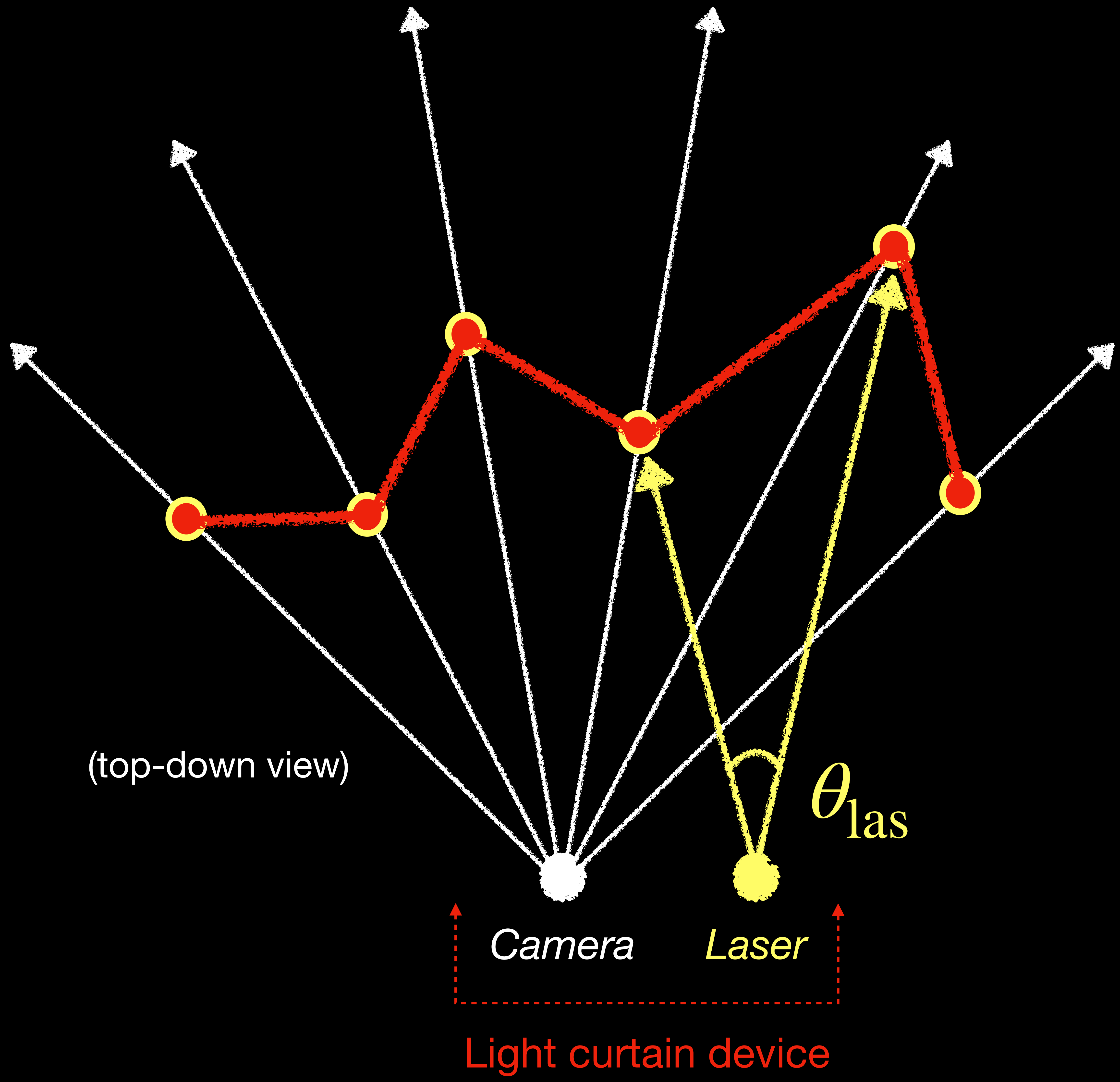
Light curtain output



"Agile Depth Sensing Using Triangulation Light Curtains", Bartels et. al.

http://www.cs.cmu.edu/~ILIM/agile_depth_sensing

Light curtain constraints



Control points

Light curtain

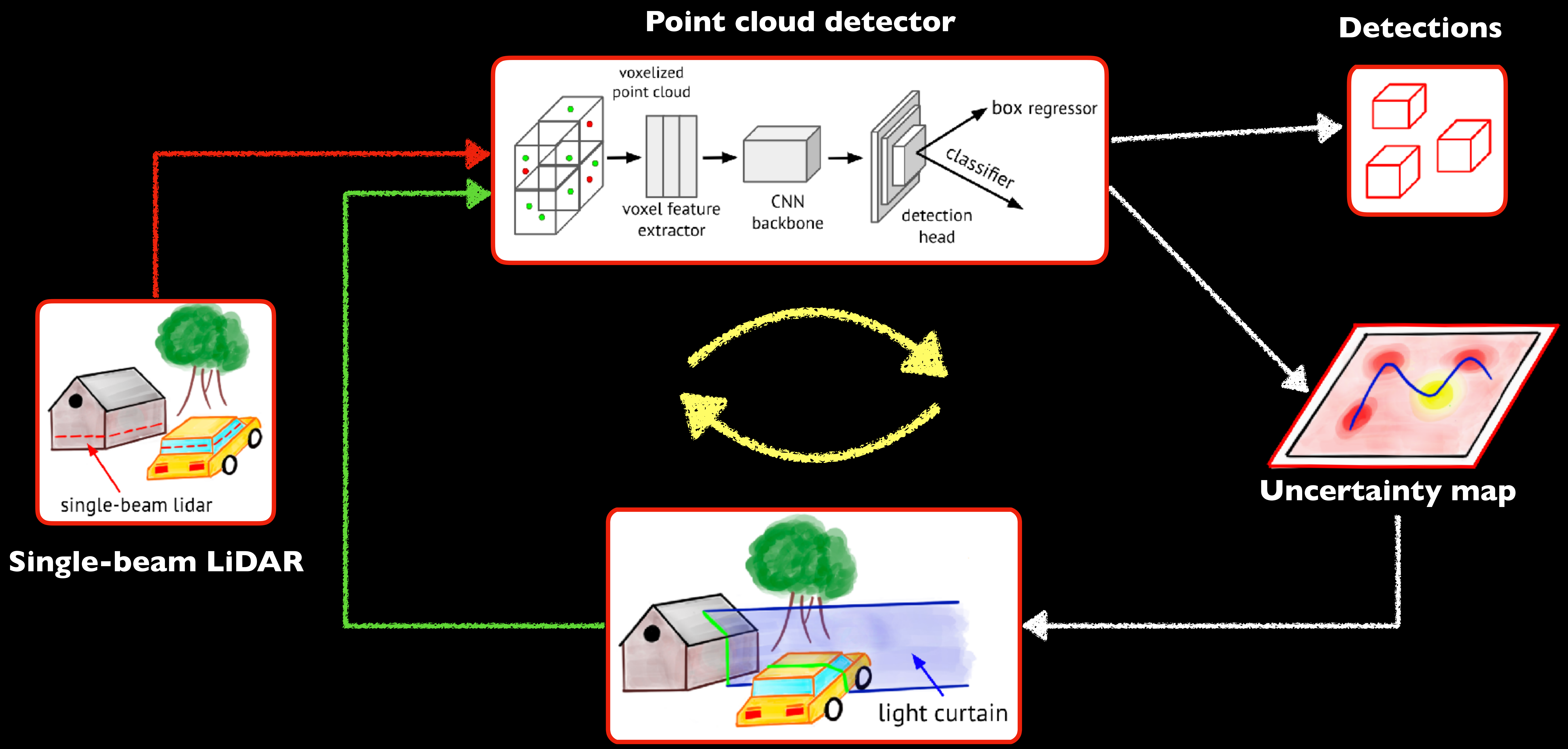
Light curtain velocity constraint

$$\theta_{las} \leq \omega_{max} \cdot \Delta t$$

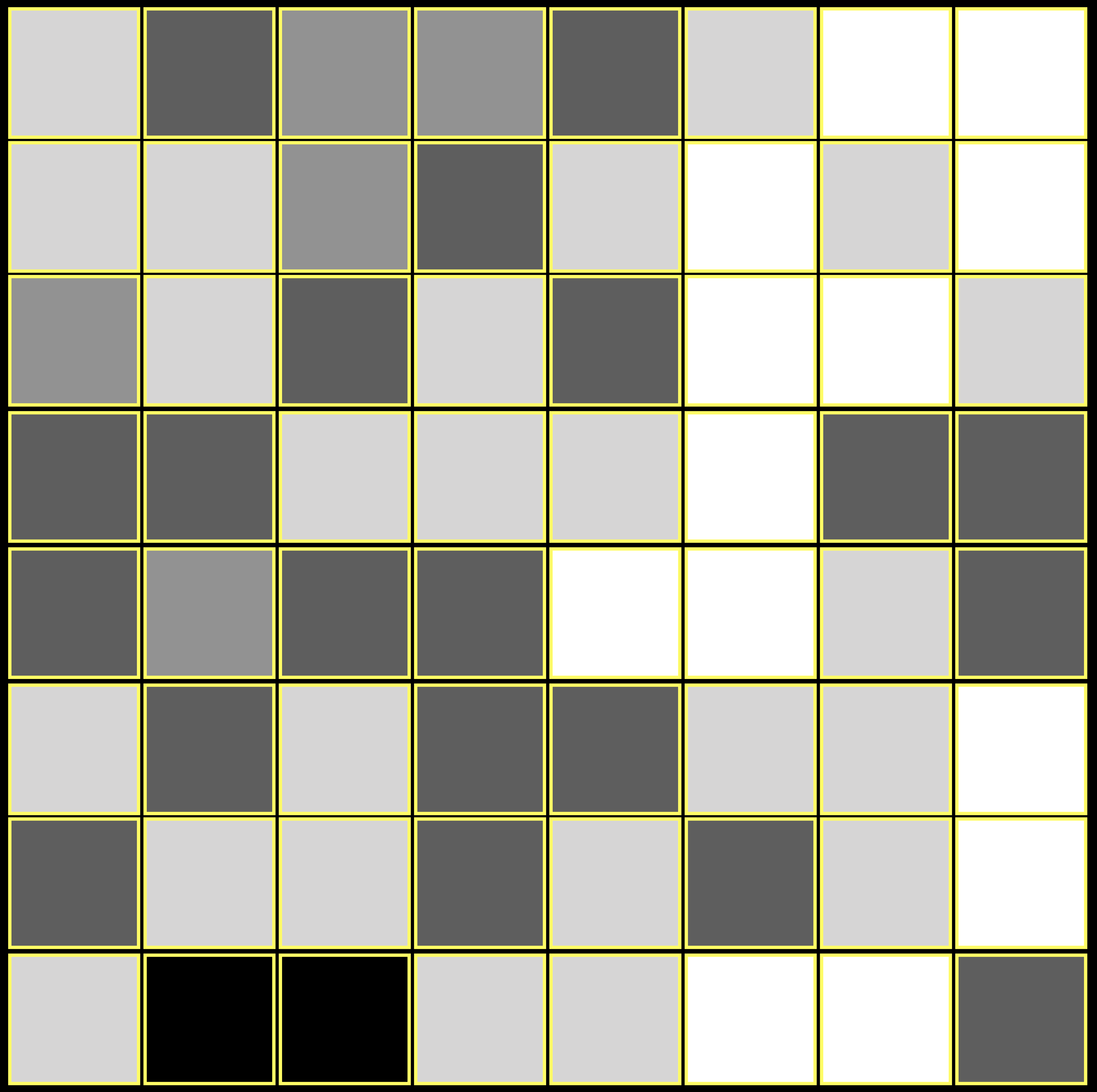
Laser max velocity limit

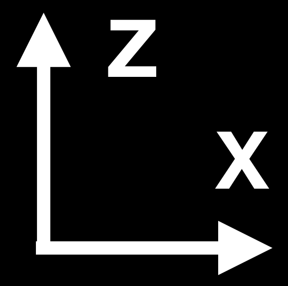
Camera sweep time between consecutive rays

Active 3D Detection Pipeline



Detector Uncertainty





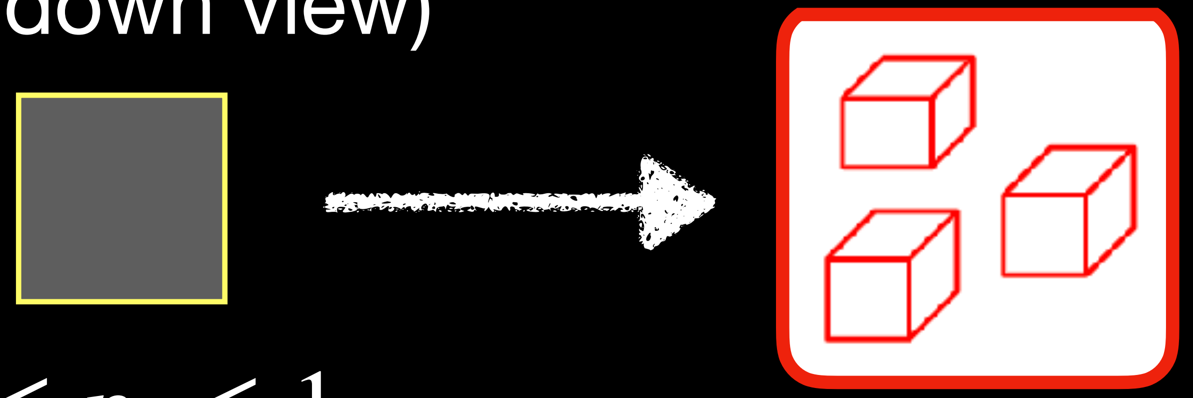
Uncertainty map
(top-down view)

Binary detection

Anchor Box
(top down view)

Detections

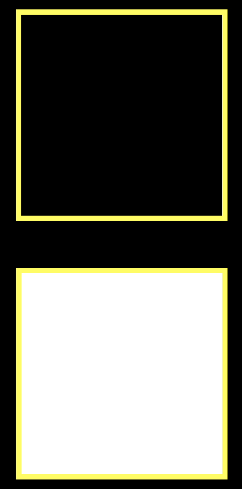
A_k



$0 \leq p_k \leq 1$

$p_k > \text{threshold}$

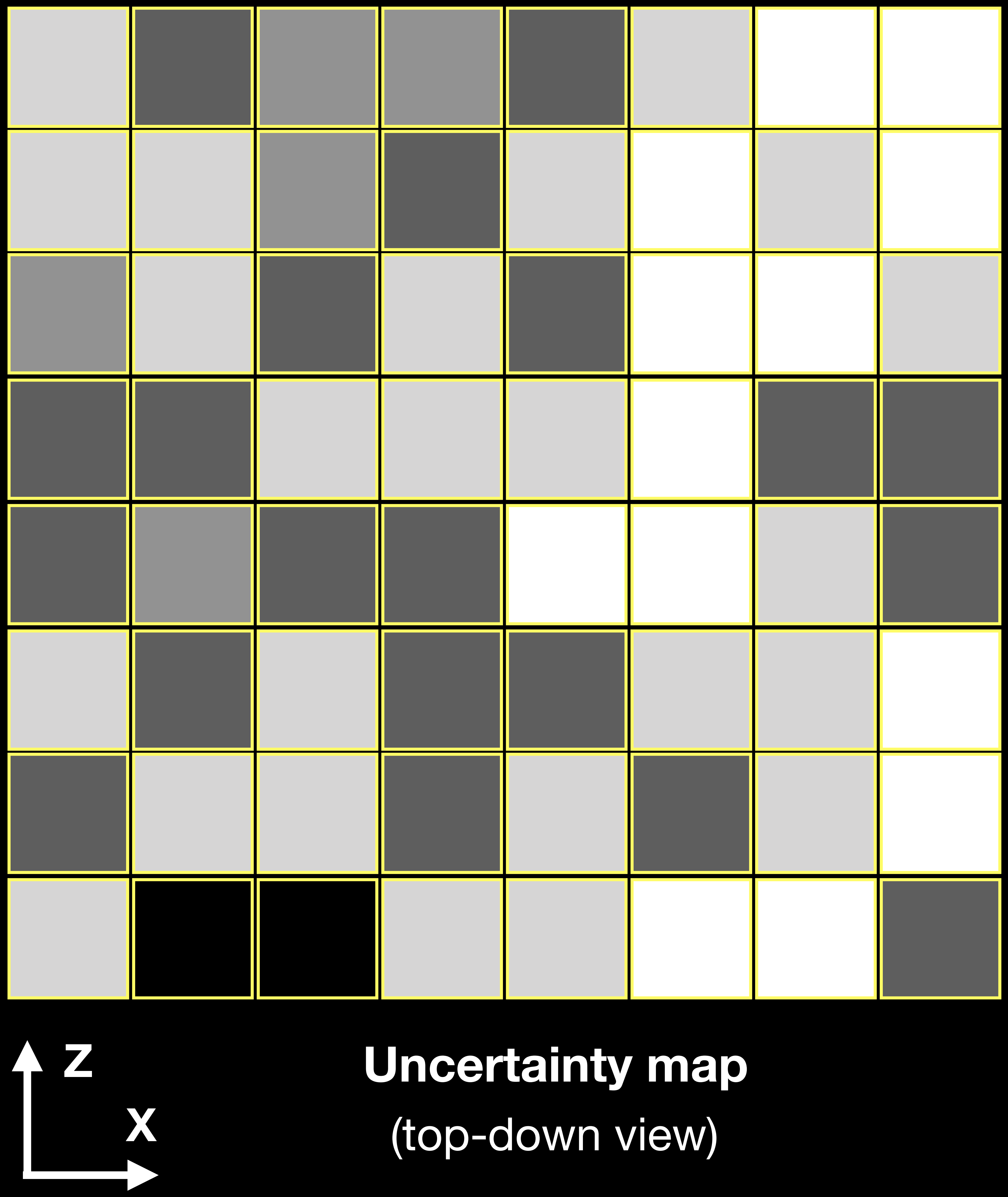
$p_k = \text{confidence}$



Lowest uncertainty
(low/high confidence)

Highest uncertainty
(medium confidence)

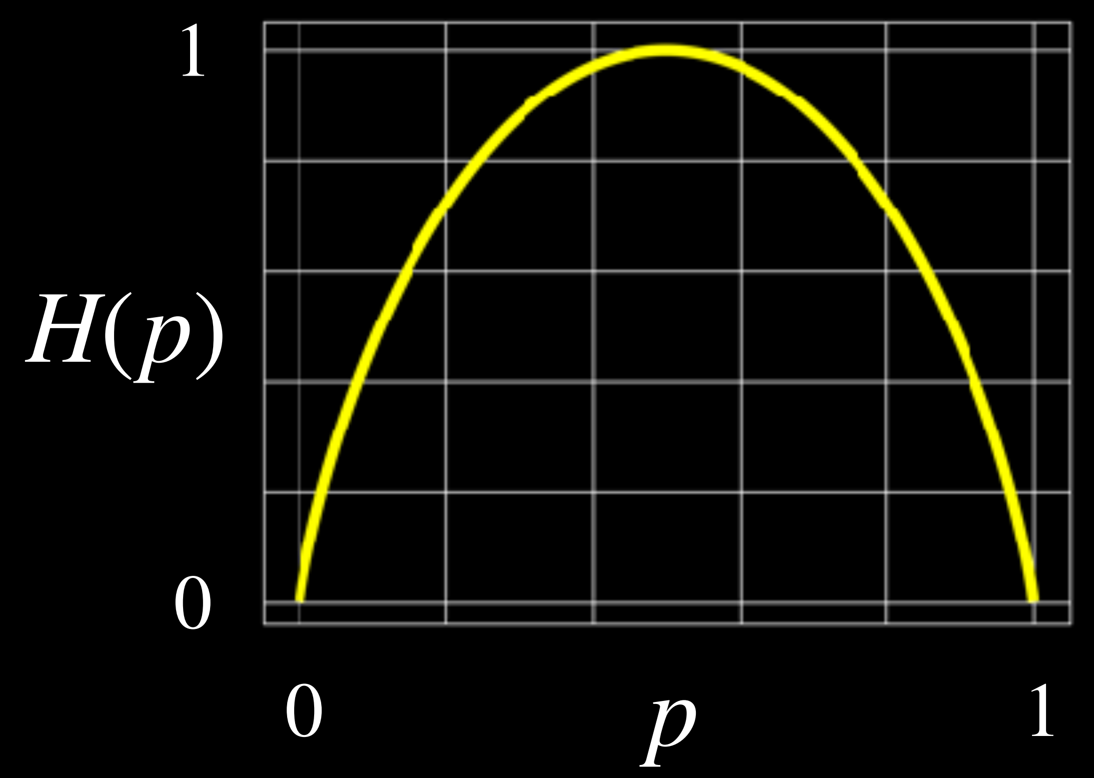
Confidence → Uncertainty



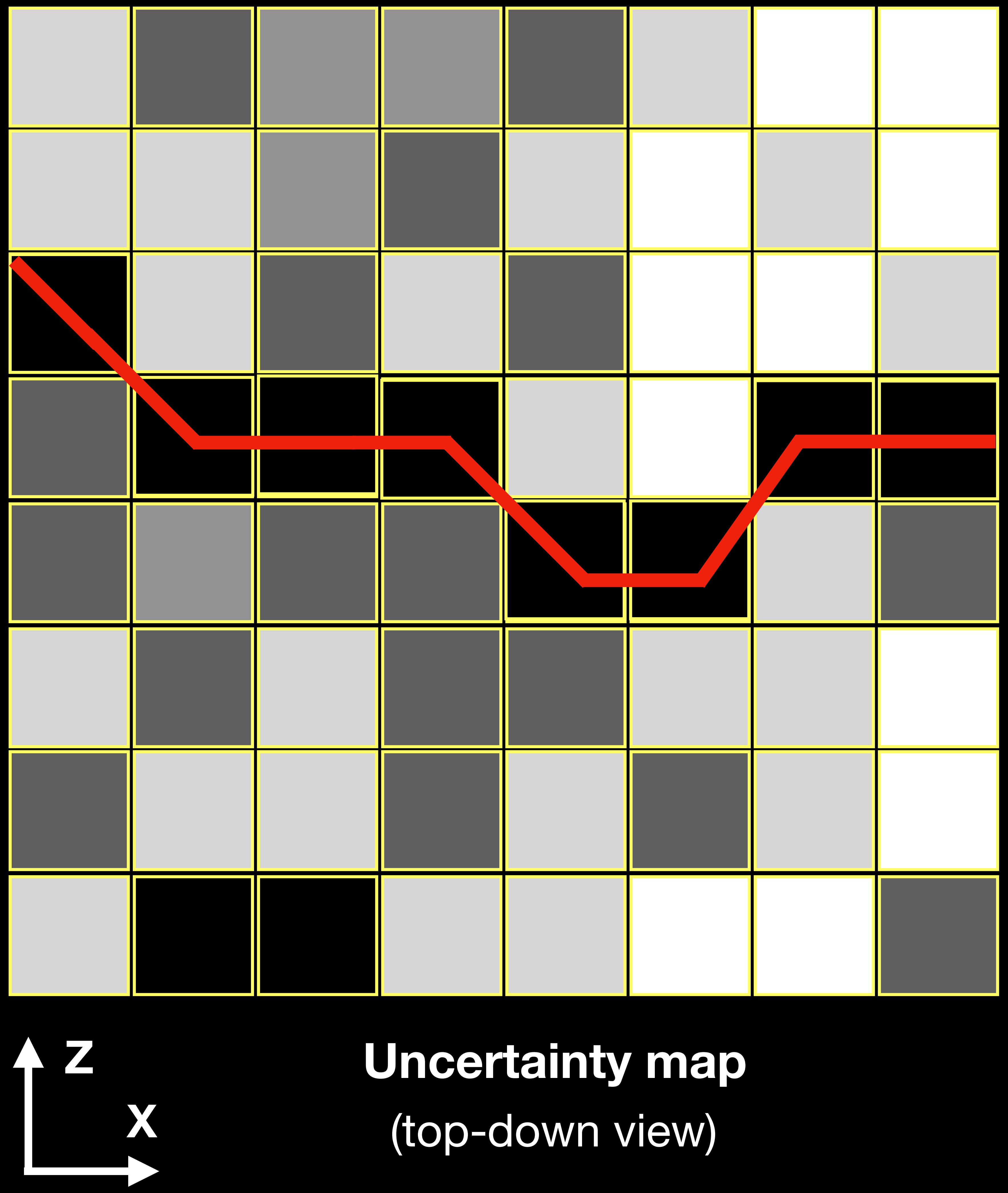
Uncertainty map
(top-down view)

Binary entropy

$$H(p) = -p\log_2 p - (1-p)\ \log_2(1-p)$$



Maximizing Information Gain



Uncertainty map
(top-down view)

Assumption 1

probabilities at each anchor locations are independent

$$H(p_{1:k}) = \sum_k H(p_k)$$

Assumption 2

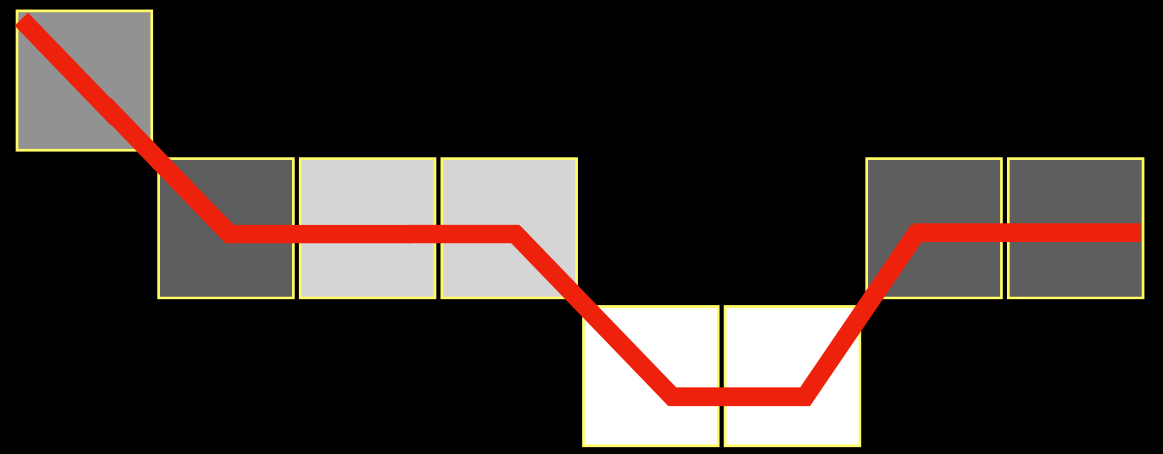
a light curtain resolves uncertainties fully and locally

Information Gain

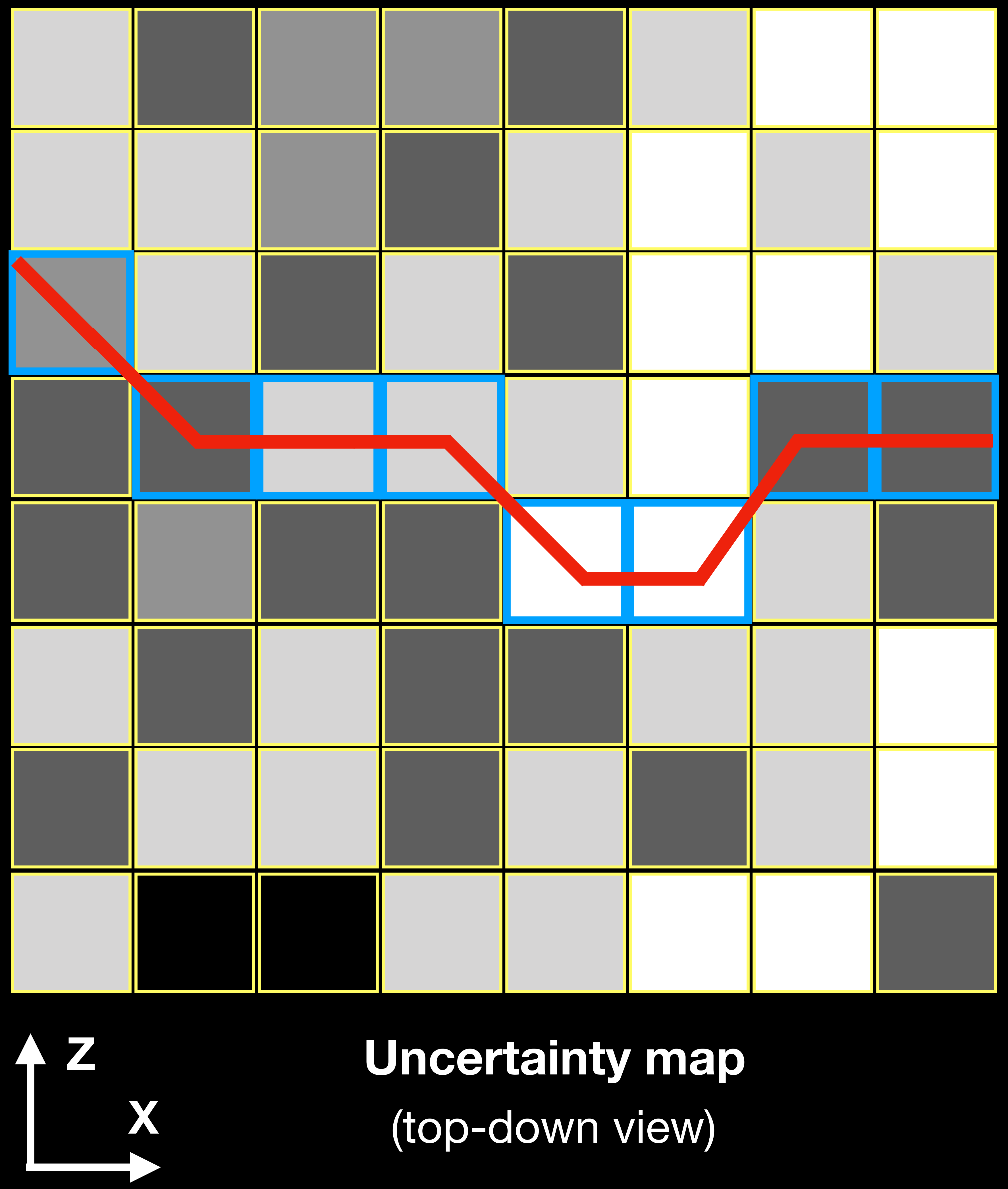
$= H$ after placement - H before placement

$$= \sum_k H(p_k^t) - \sum_k H(p_k^{t-1})$$

$= \sum_k H(p_k)$, where k-th anchor lies under the curtain

$= \sum$ ()

Maximization Objective

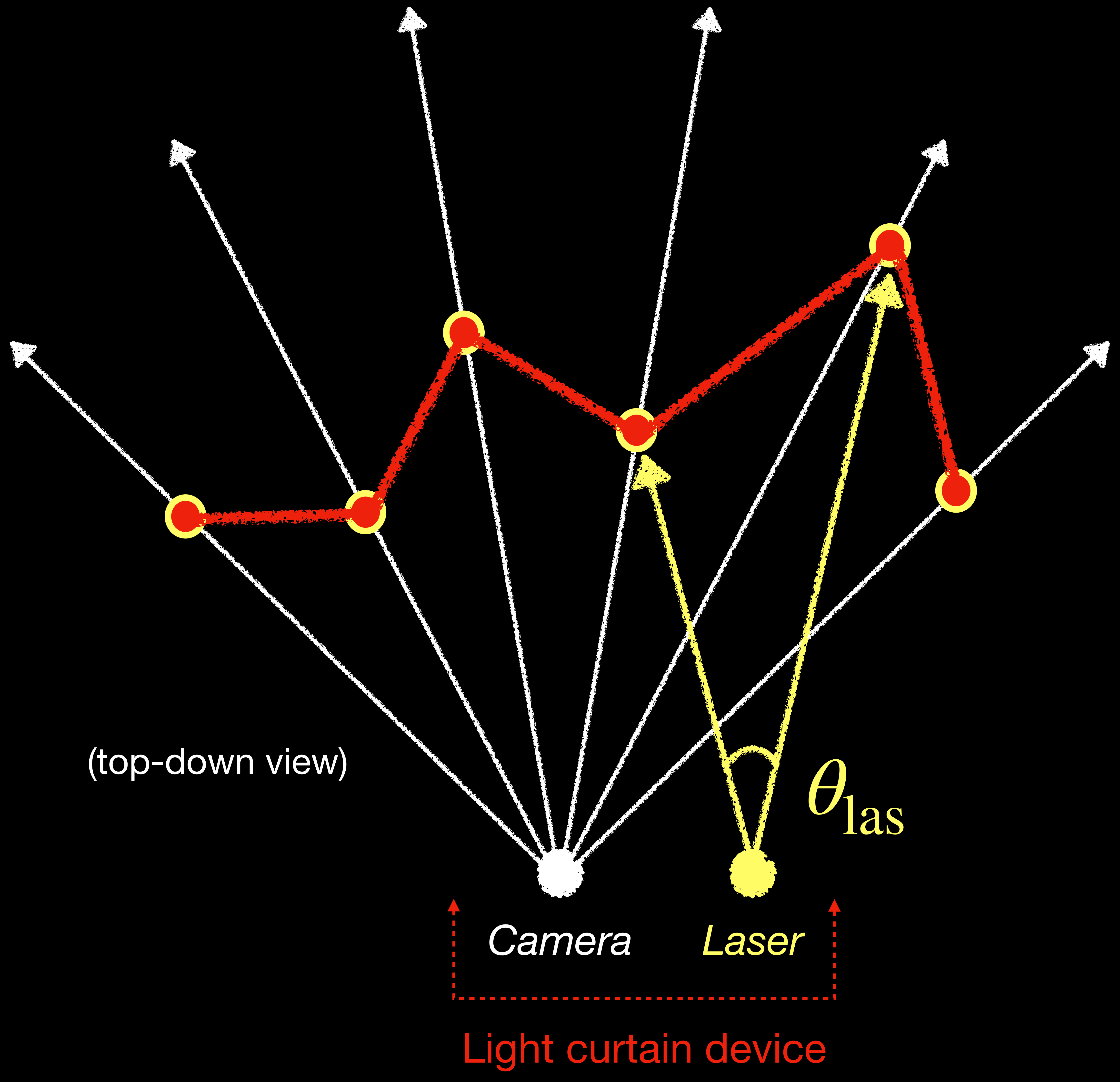


Uncertainty map
(top-down view)

Objective to place light curtains:

Maximize the sum of binary entropies of regions covered by the light curtain!

Light curtain parametrization



- Control points
- Light curtain

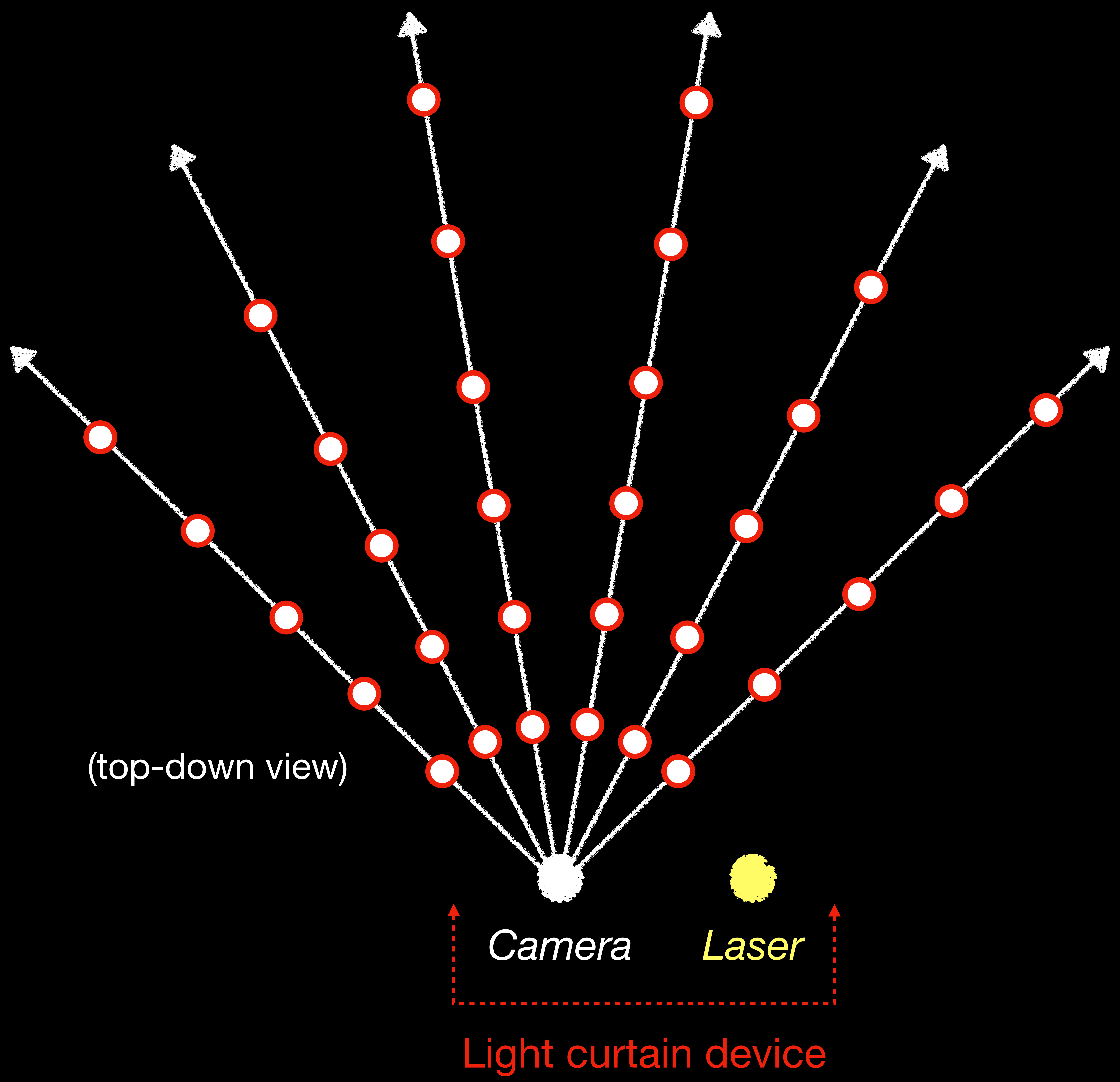
Light curtain constraint

$$\theta_{\text{las}} \leq \omega_{\text{max}} \cdot \Delta t$$

Laser max velocity limit

Camera sweep time between consecutive rays

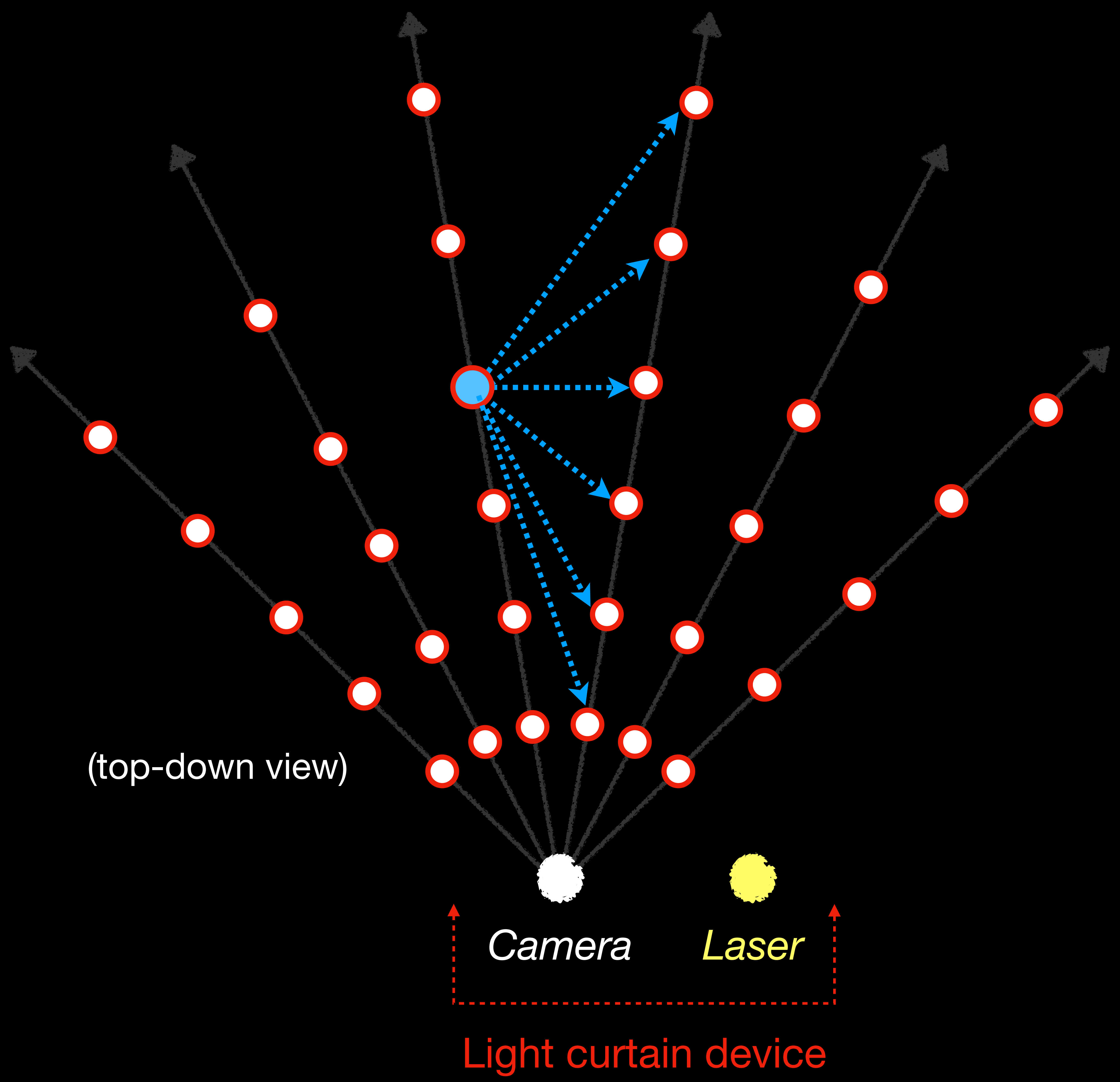
Constructing a *Constraint Graph*



Step 1 ● *Graph node*

Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Constructing a *Constraint Graph*



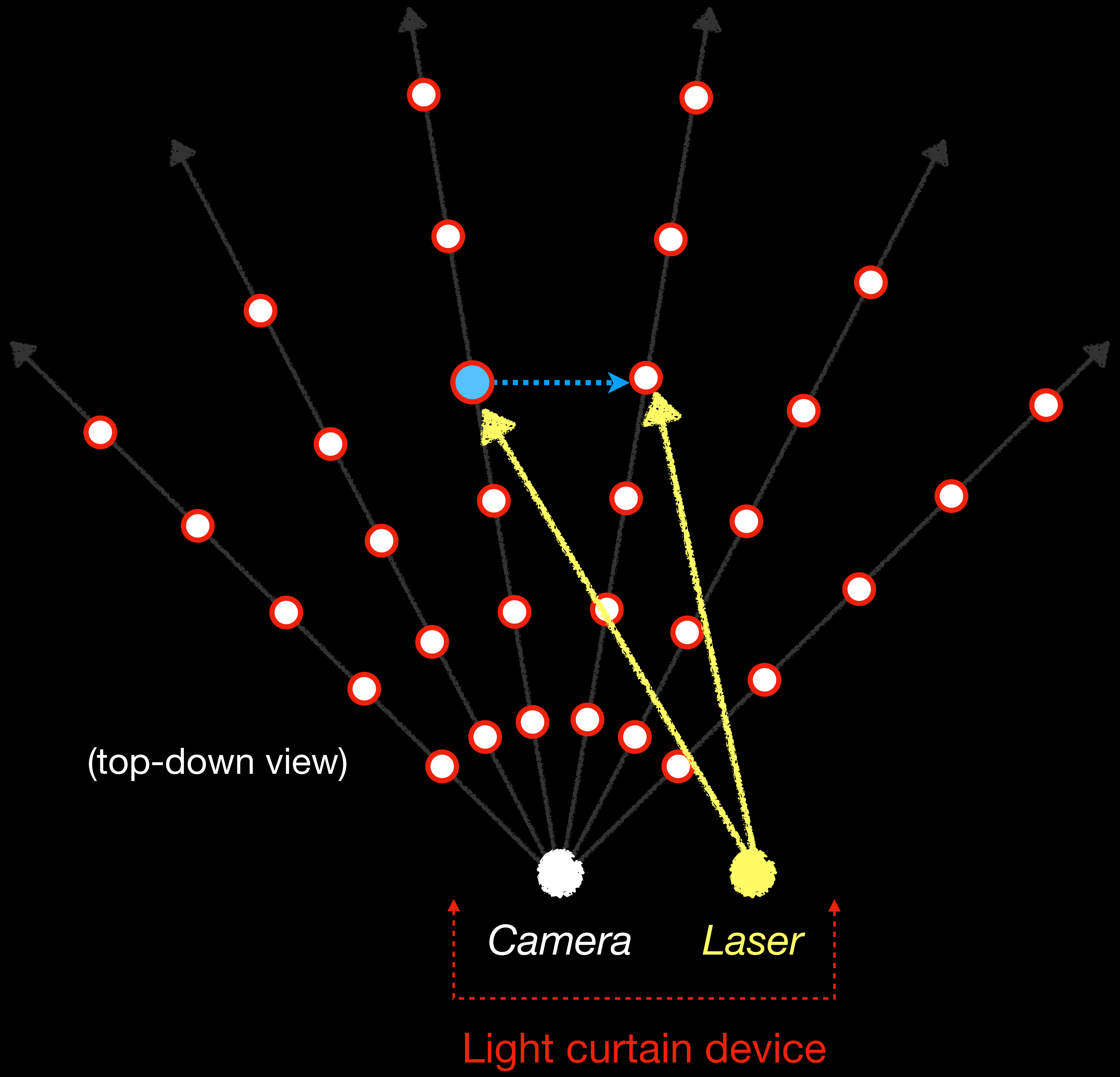
Step 1 ● *Graph node*

Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2

For each node, consider all nodes on the next ray.

Constructing a *Constraint Graph*



Step 1 *Graph node*

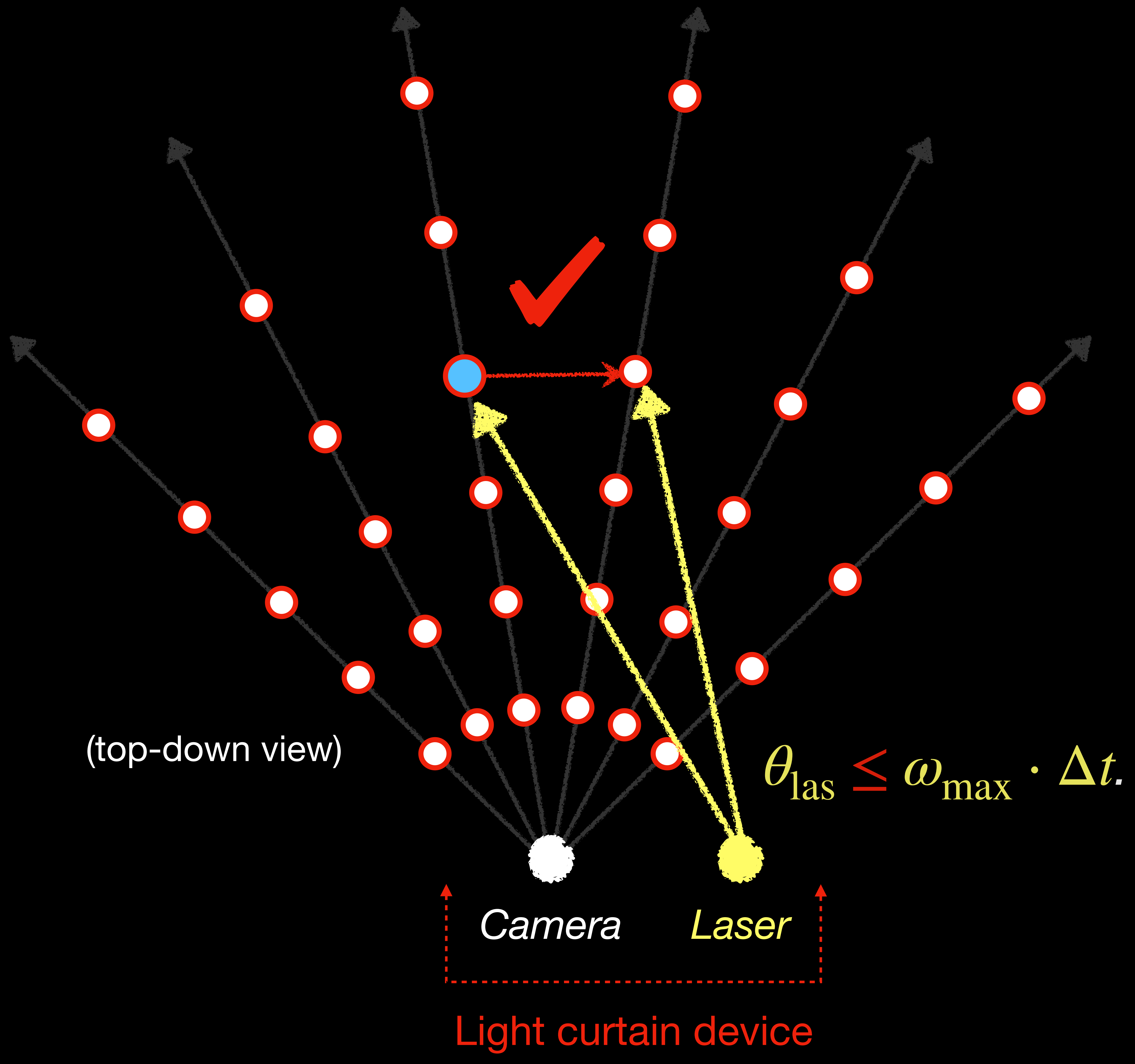
Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{\mathrm{las}} \leq \omega_{\mathrm{max}} \cdot \Delta t$ *as edges.*

Constructing a *Constraint Graph*



Step 1 — Graph node

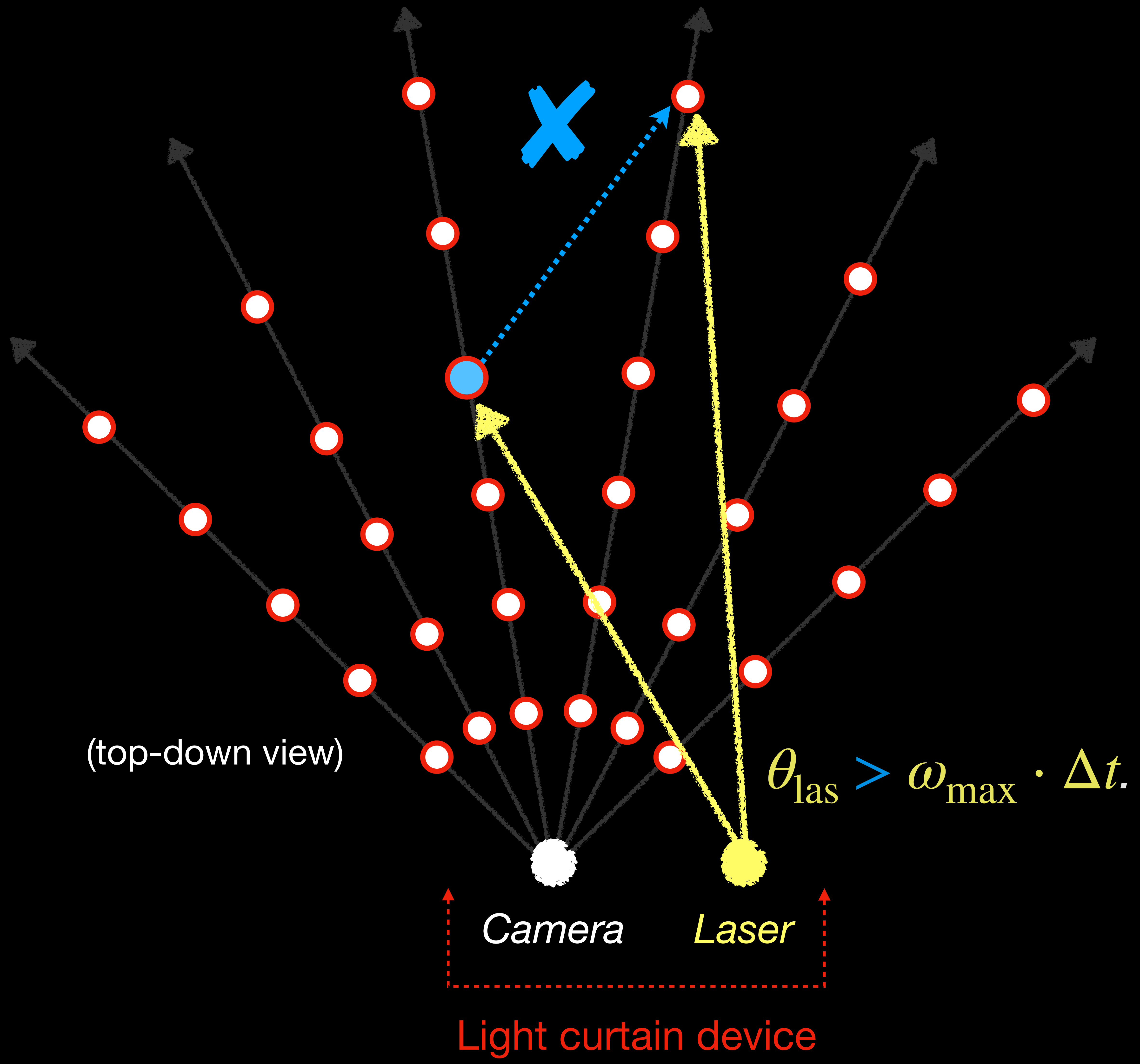
Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2 — *Graph edge*

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{las} \leq \omega_{max} \cdot \Delta t$ *as edges.*

Constructing a *Constraint Graph*



Step 1 — *Graph node*

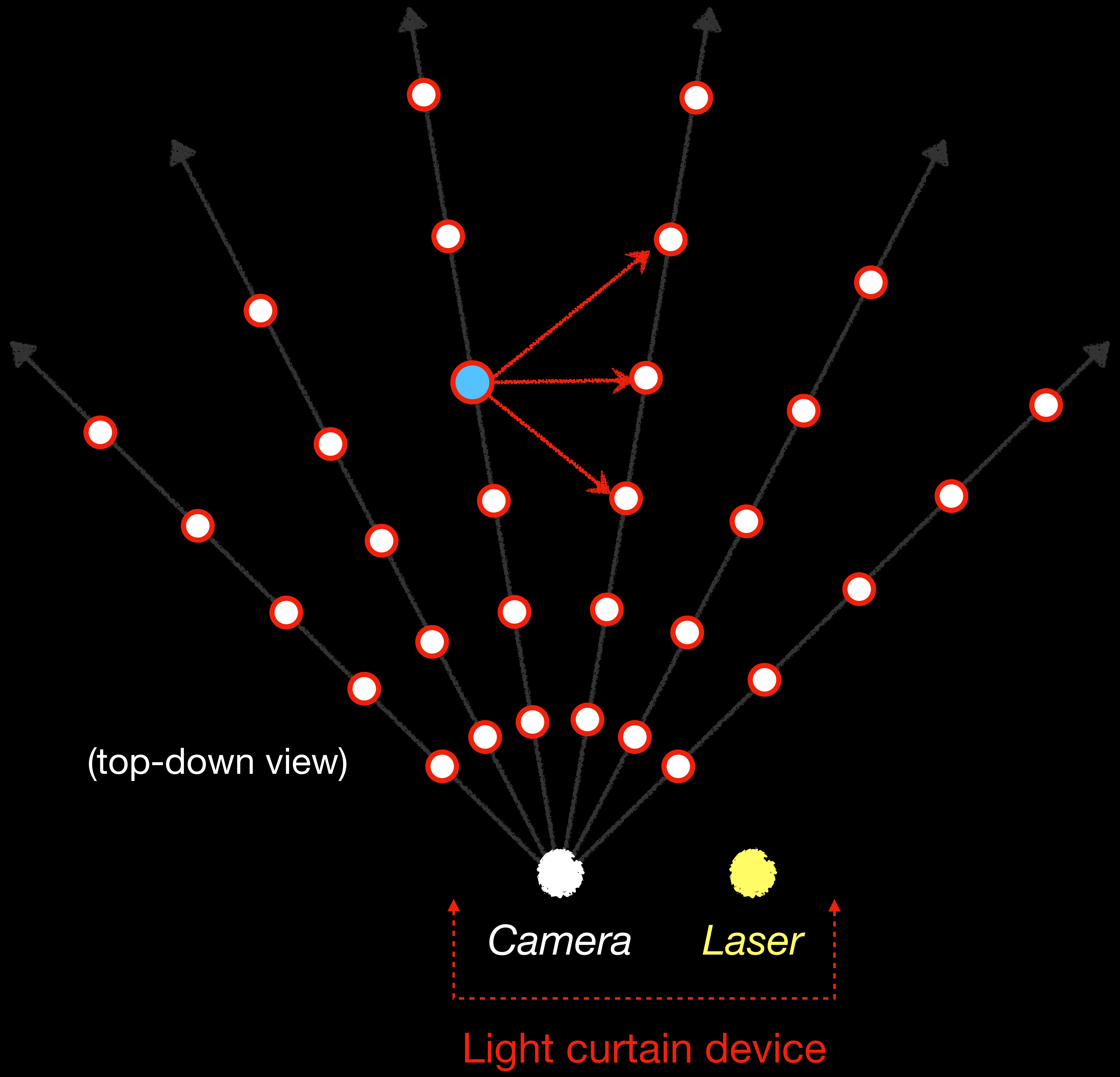
Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2 — *Graph edge*

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{las} \leq \omega_{max} \cdot \Delta t$ *as edges.*

Constructing a *Constraint Graph*



Step 1 — *Graph node*

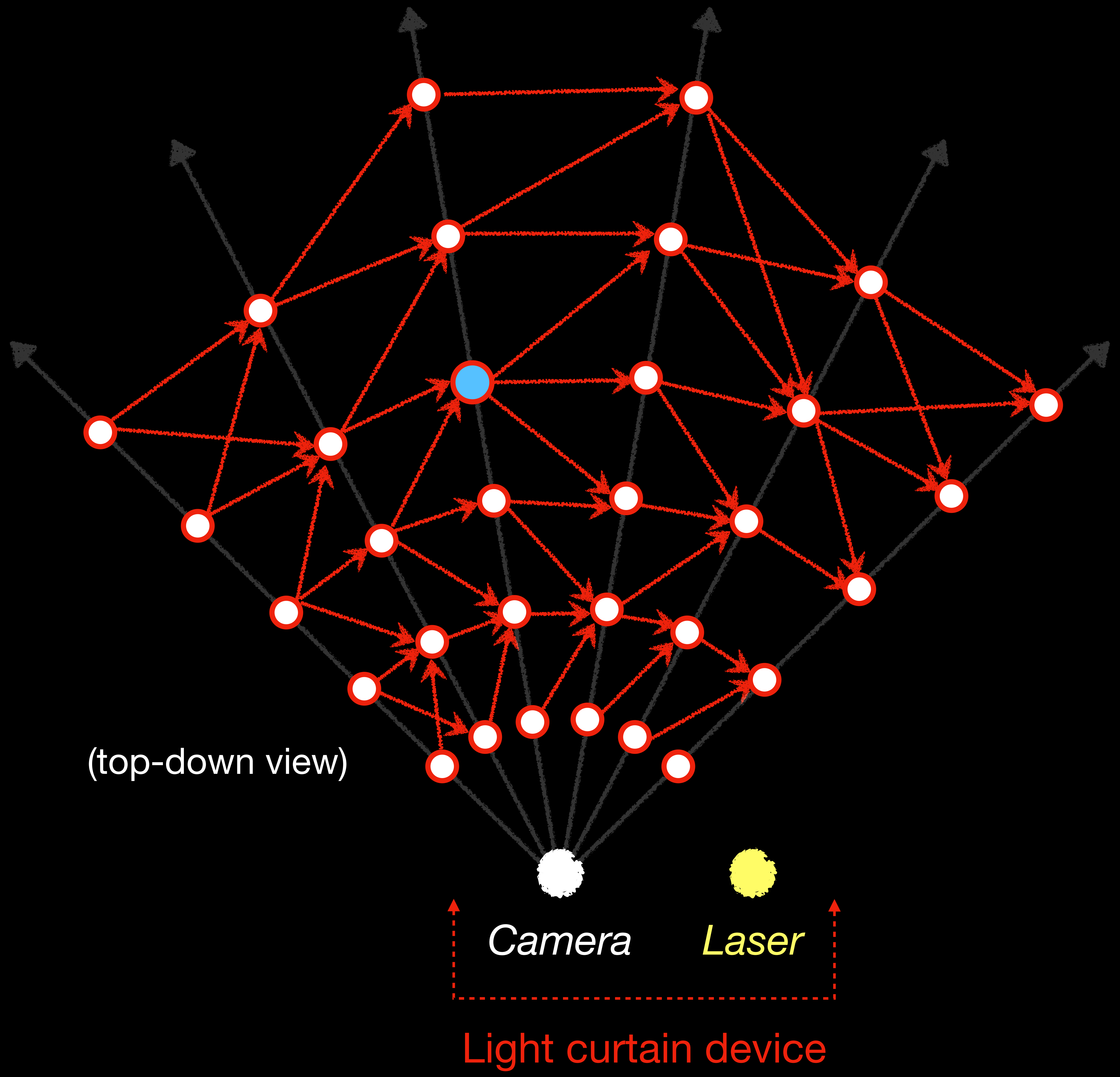
Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2 — *Graph edge*

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{\text{las}} \leq \omega_{\text{max}} \cdot \Delta t$ *as edges.*

Constructing a *Constraint Graph*



Step 1 ● *Graph node*

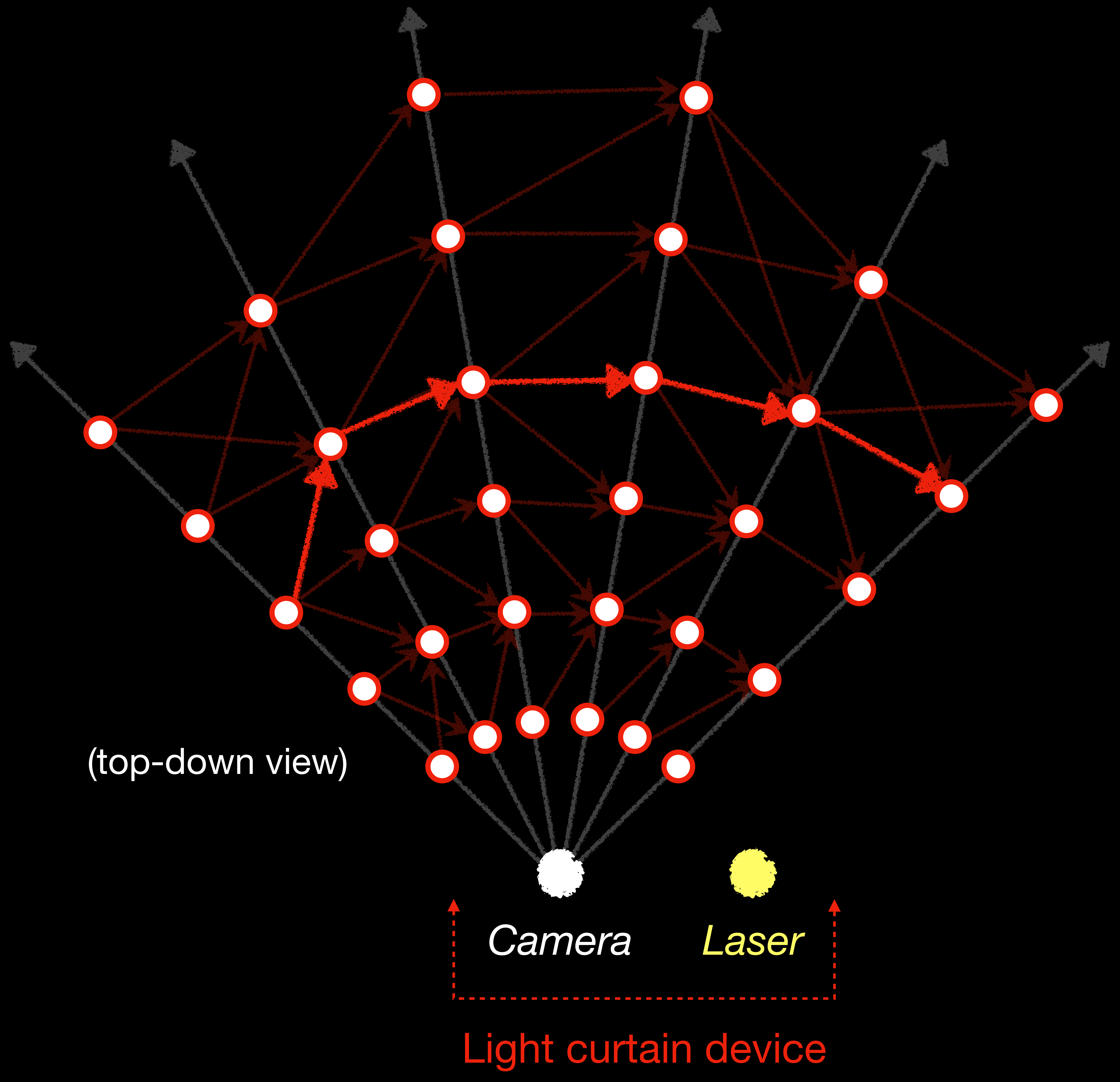
Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

Step 2 → *Graph edge*

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{las} \leq \omega_{max} \cdot \Delta t$ *as edges.*

Constructing a *Constraint Graph*



Step 1 *Graph node*

Select equally spaced points as candidate control points on each camera ray. These are the nodes of the graph.

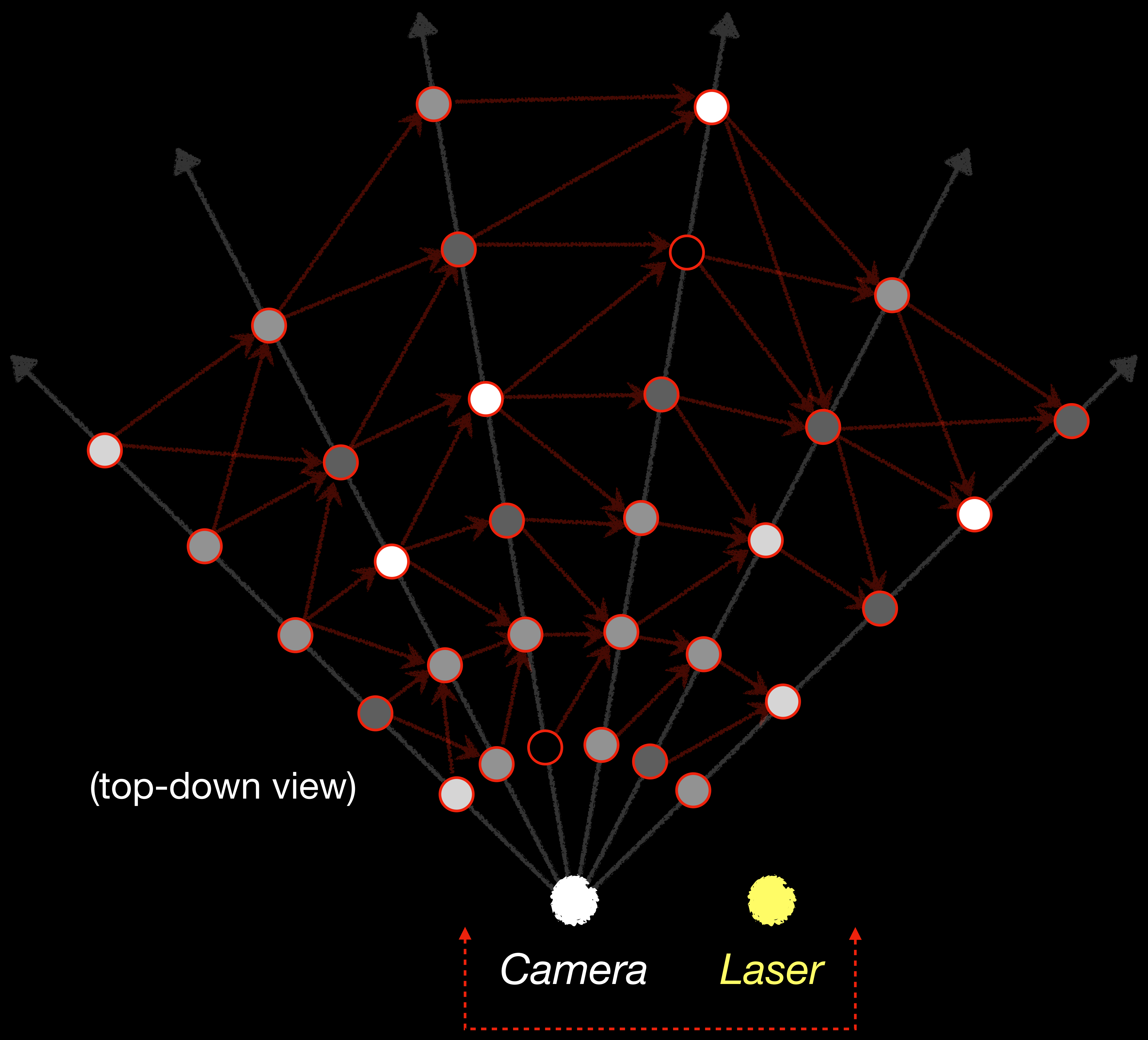
Step 2 *Graph edge*

For each node, consider all nodes on the next ray.

Add those nodes that satisfy the velocity constraint $\theta_{\text{las}} \leq \omega_{\text{max}} \cdot \Delta t$ *as edges.*

Then, any path in the graph is a valid curtain!

Uncertainty maximizing Light Curtain

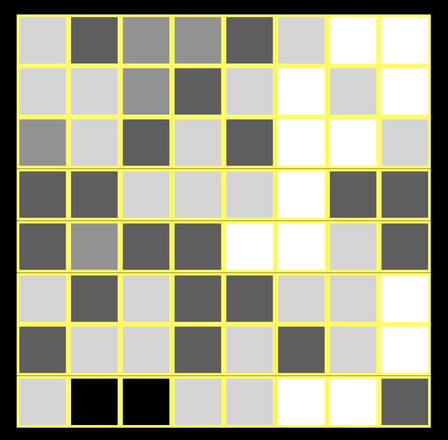


Step 1 Graph node Graph edge

Construct the constraint graph

Step 2

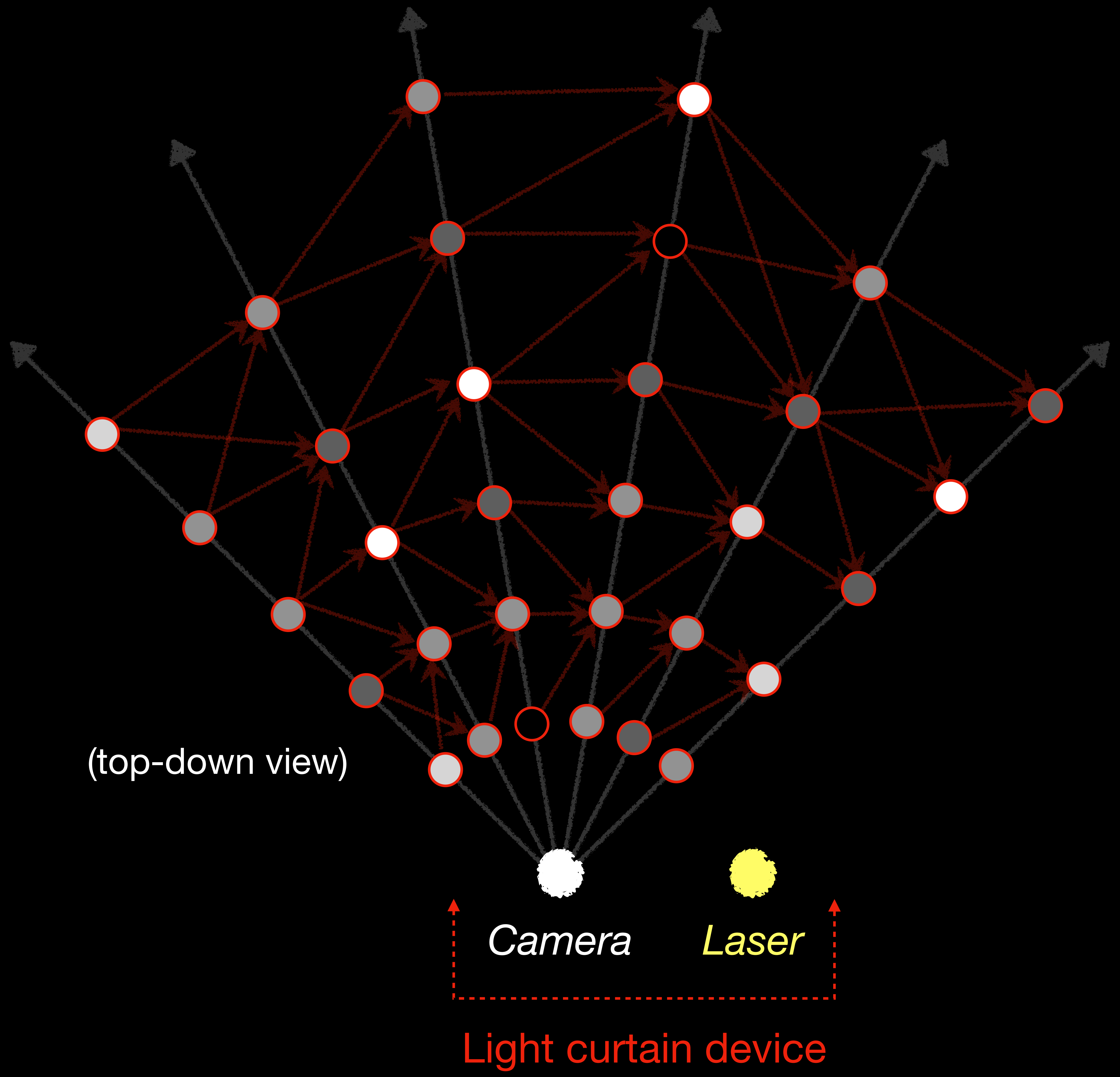
Assign uncertainty to each node by interpolating from the uncertainty map



Step 3

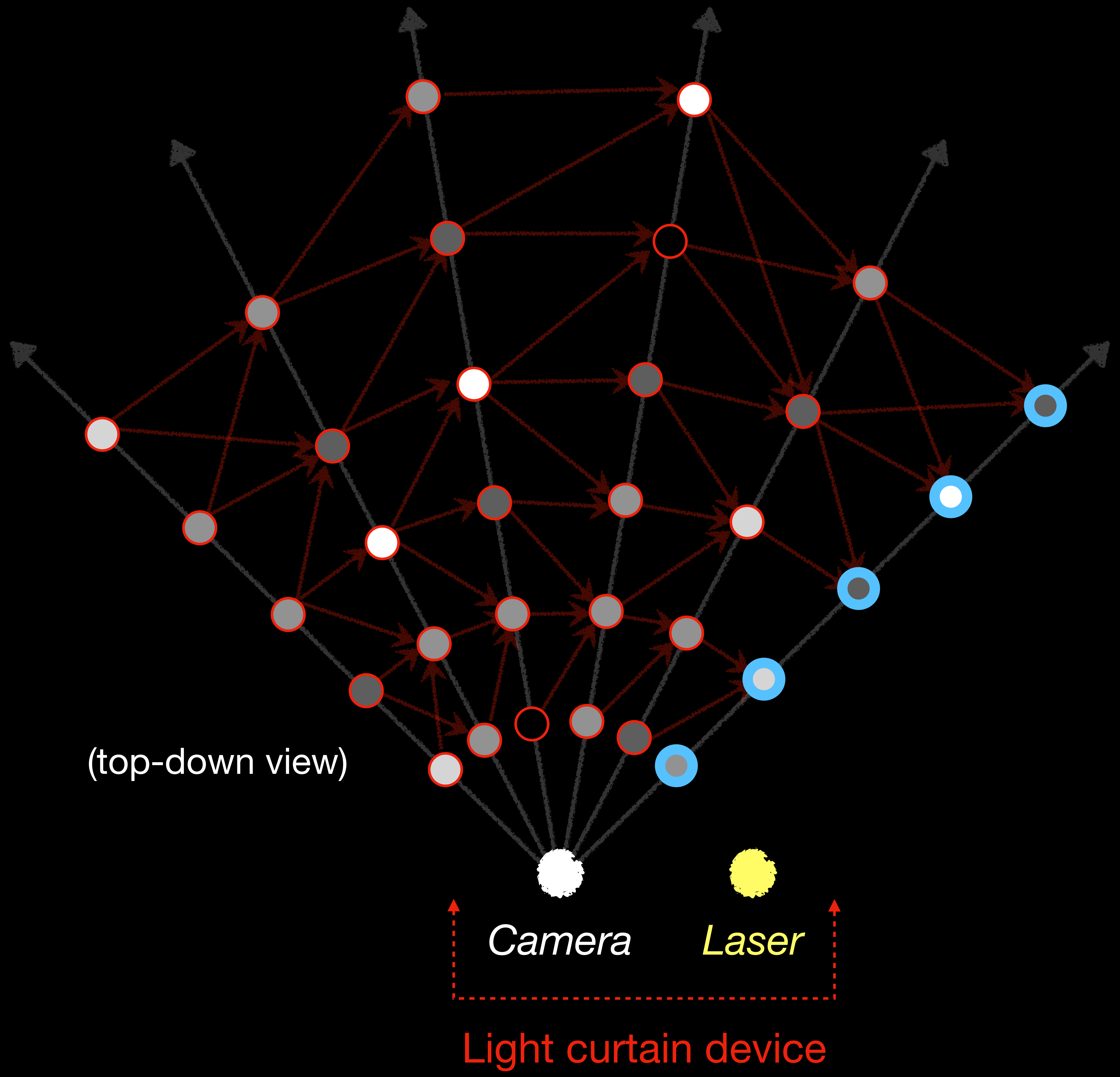
Perform dynamic programming

Dynamic Programming



For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

Dynamic Programming

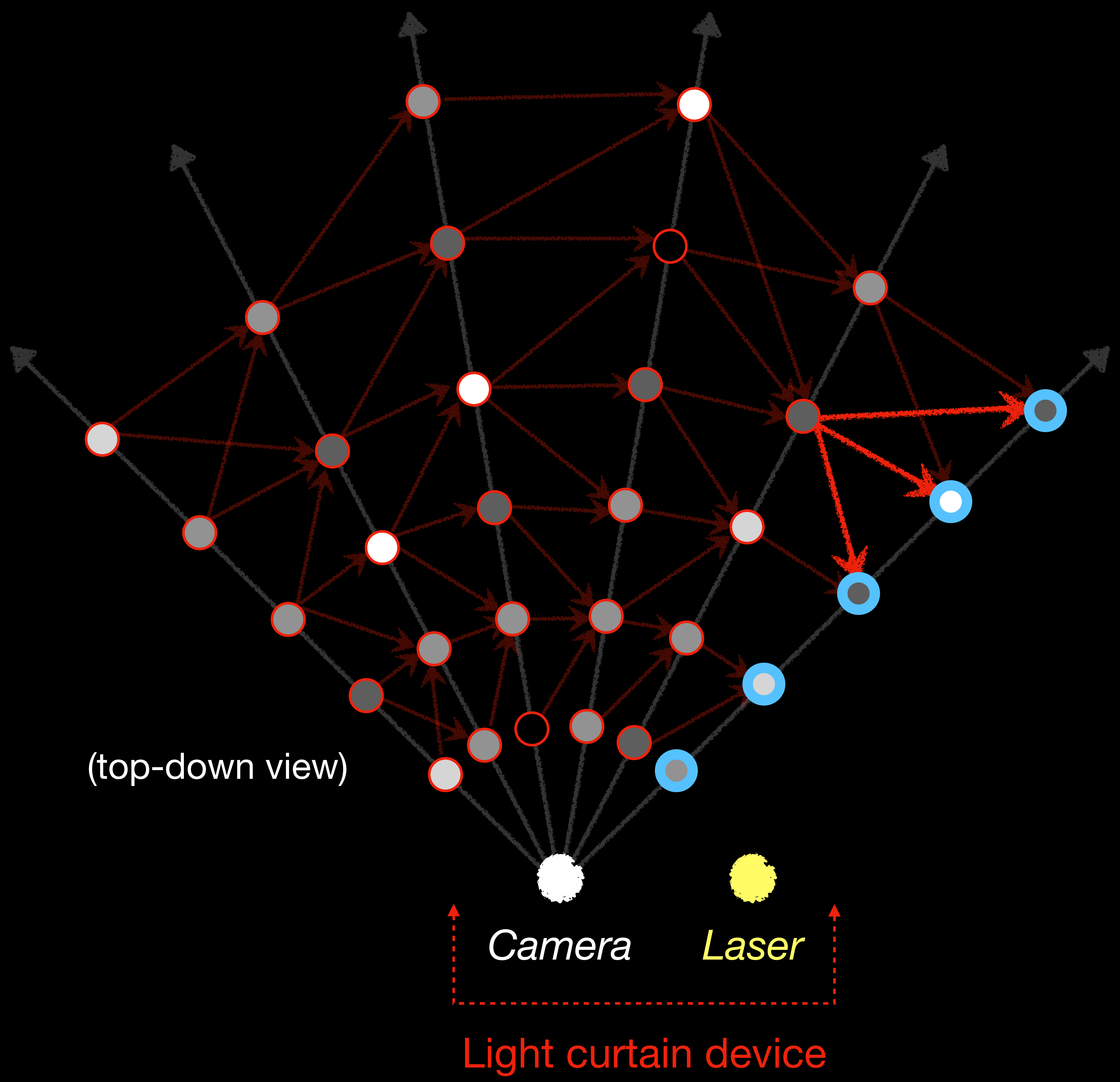


Graph node → Graph edge

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*

Dynamic Programming

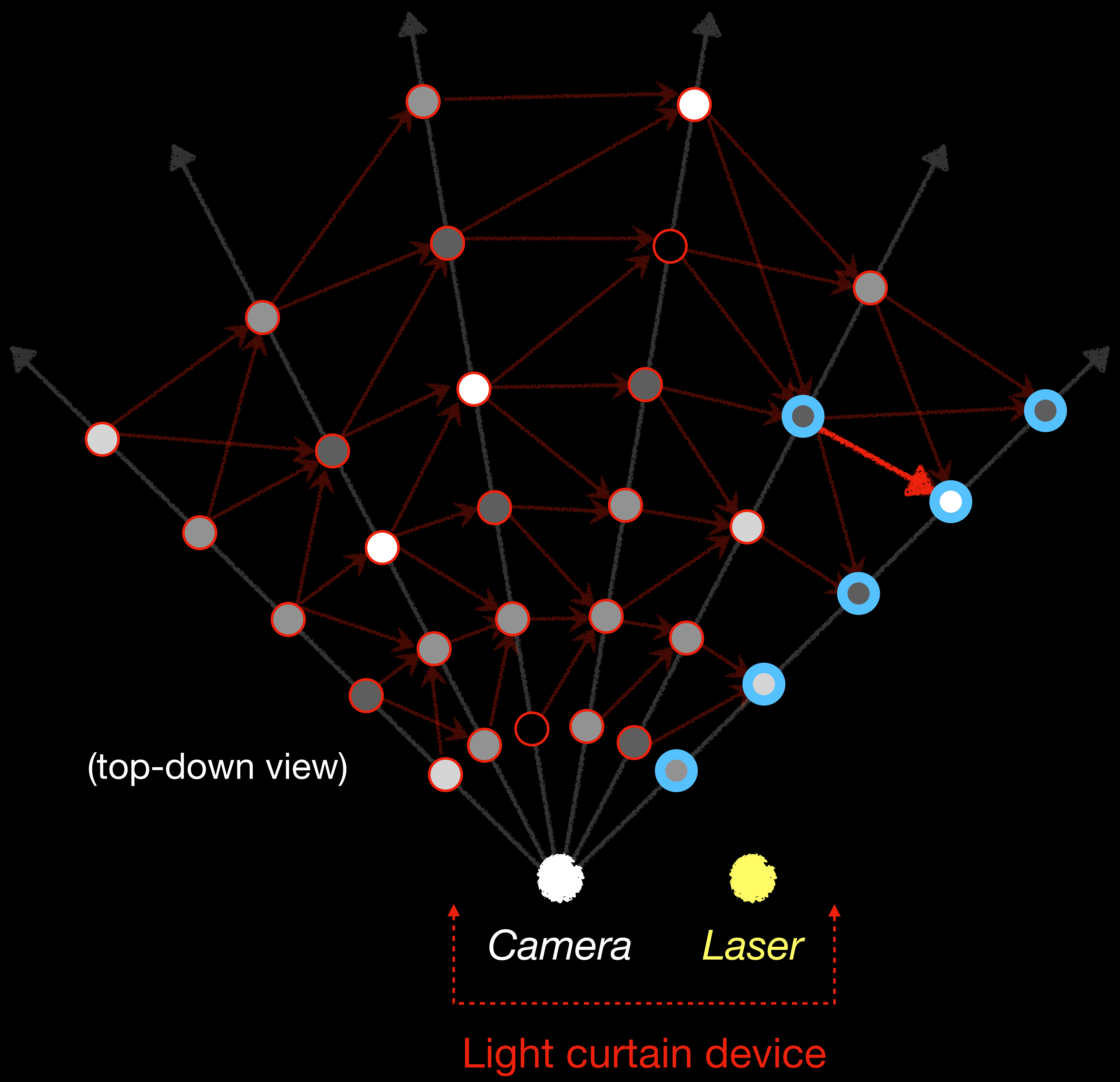


Graph node → Graph edge

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
    - *Look at all its edges* →

Dynamic Programming

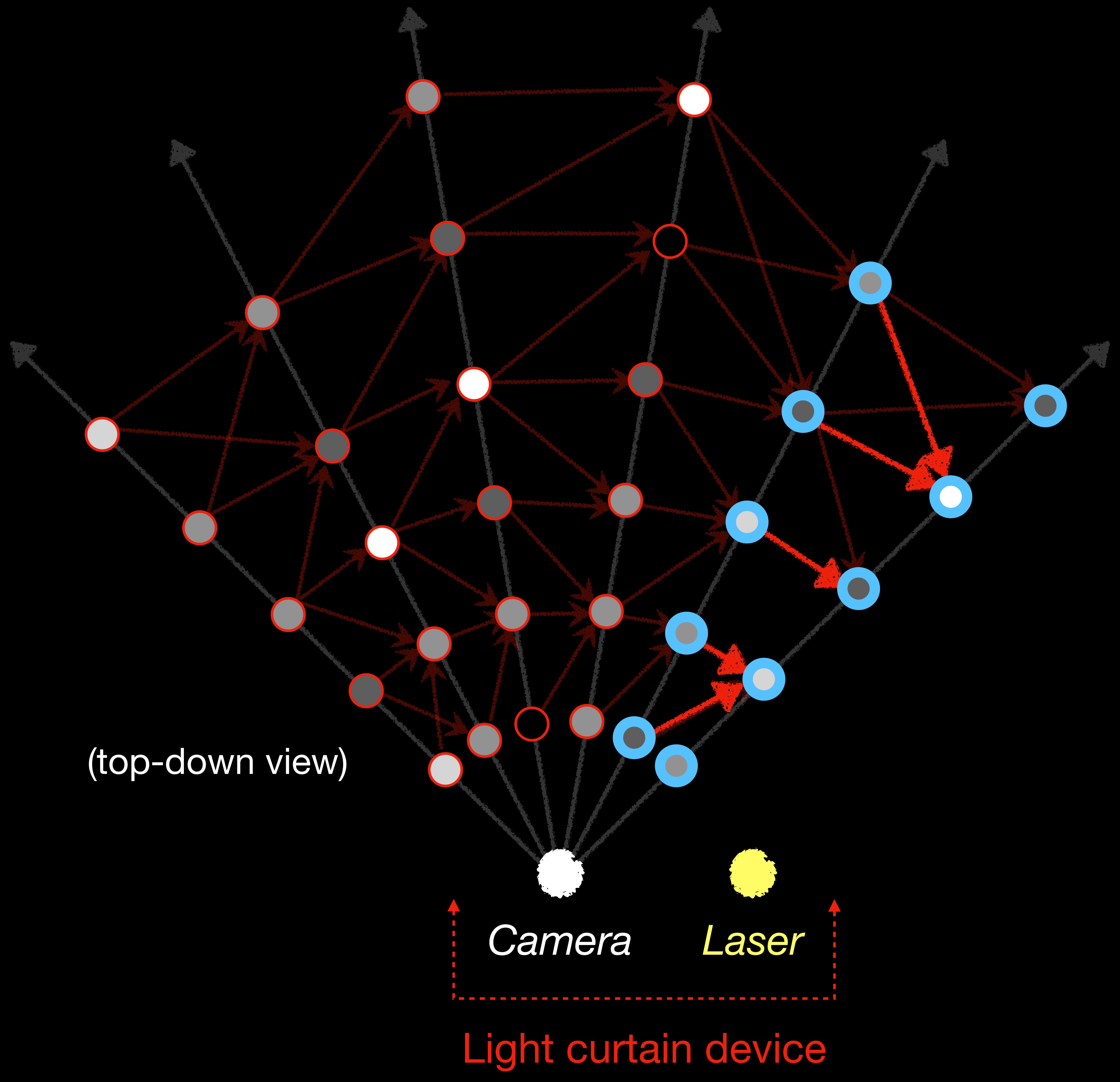


Graph node → Graph edge

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its →*
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*

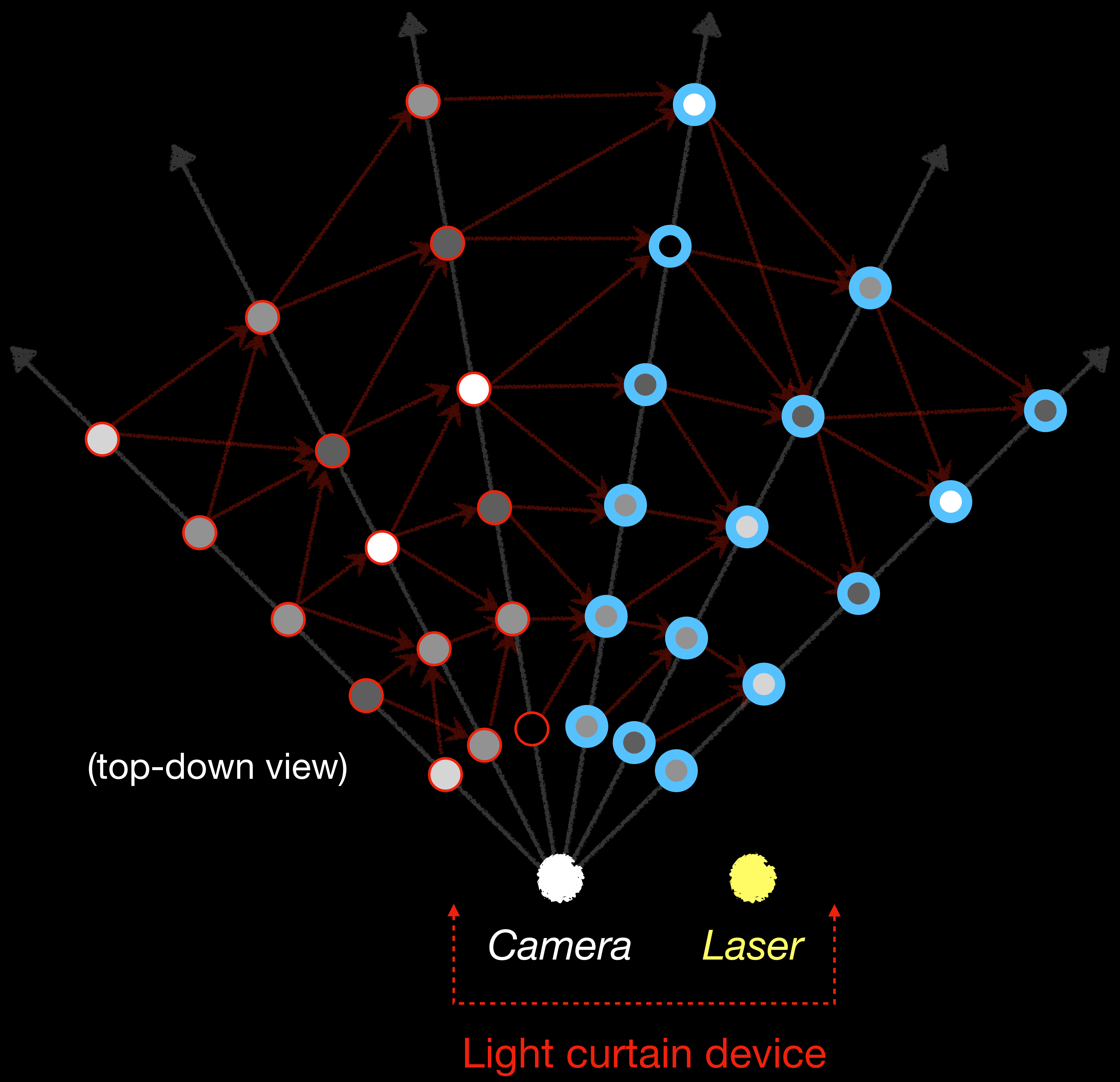
Dynamic Programming



For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its* →
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*
  - *Repeat for all nodes in ray*

Dynamic Programming

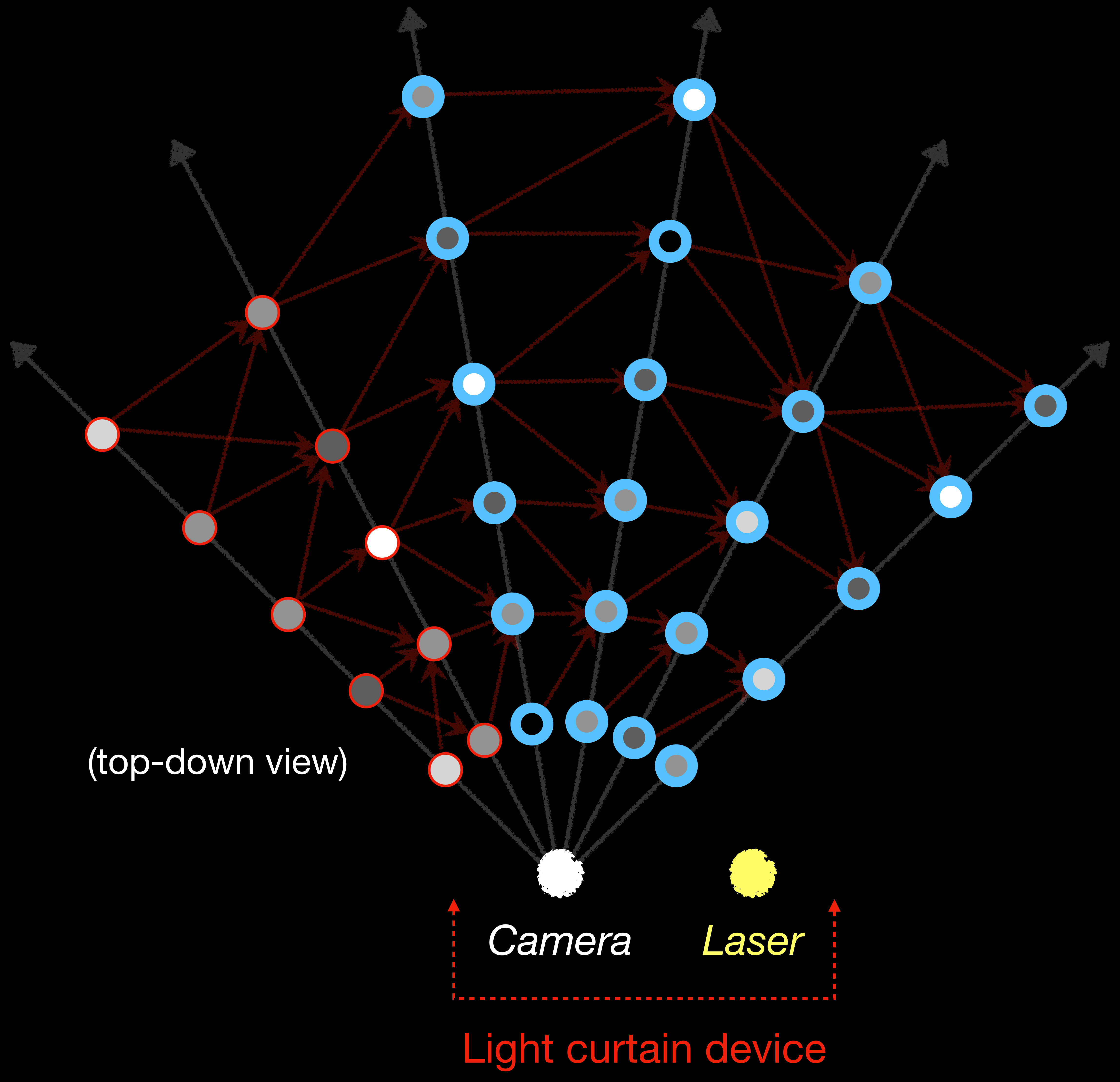


Graph node *Graph edge*

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its*
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*
  - *Repeat for all nodes in ray*
- *Repeat over each previous ray*

Dynamic Programming

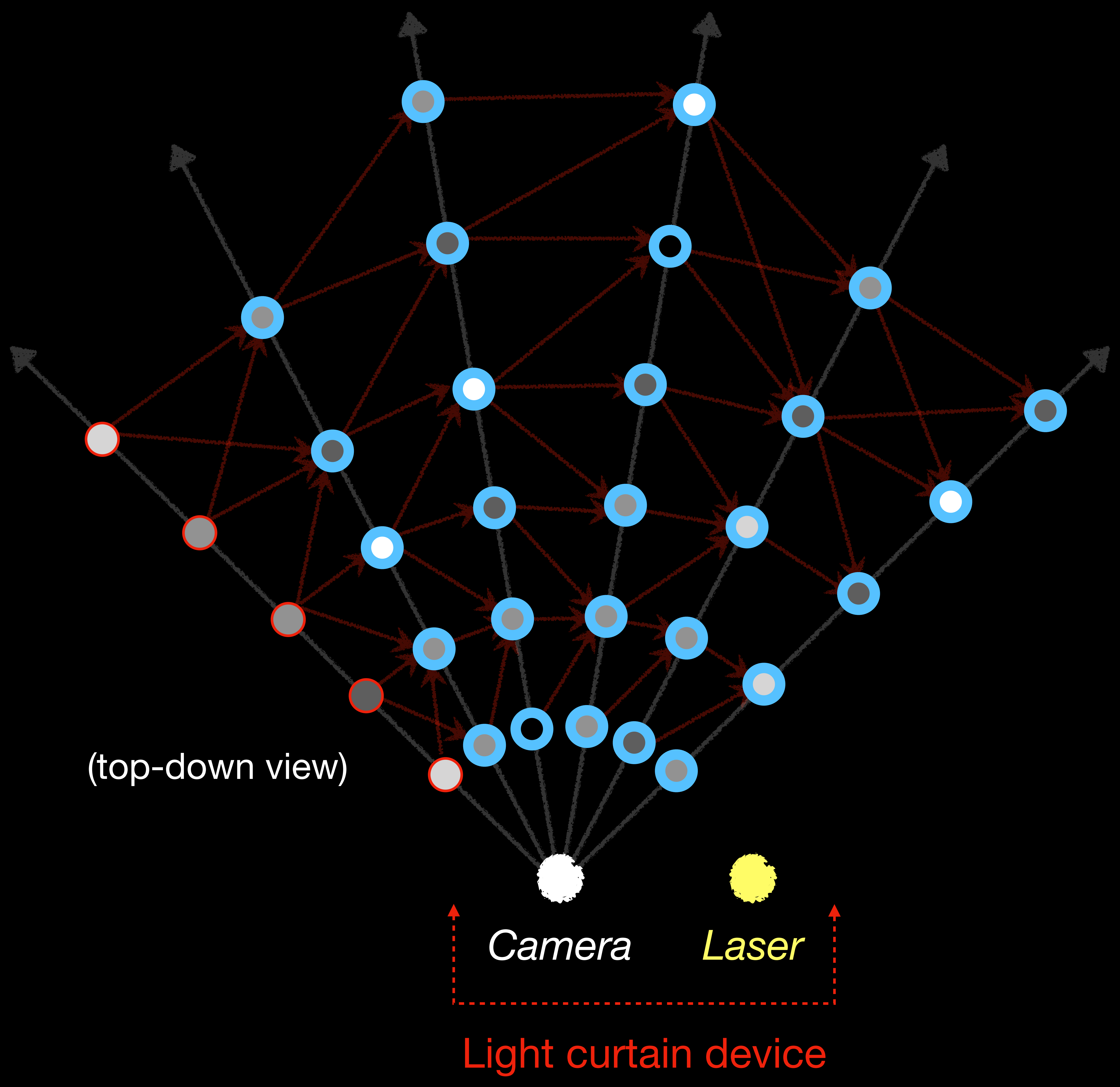


Graph node → *Graph edge*

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its* →
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*
  - *Repeat for all nodes in ray*
- *Repeat over each previous ray*

Dynamic Programming

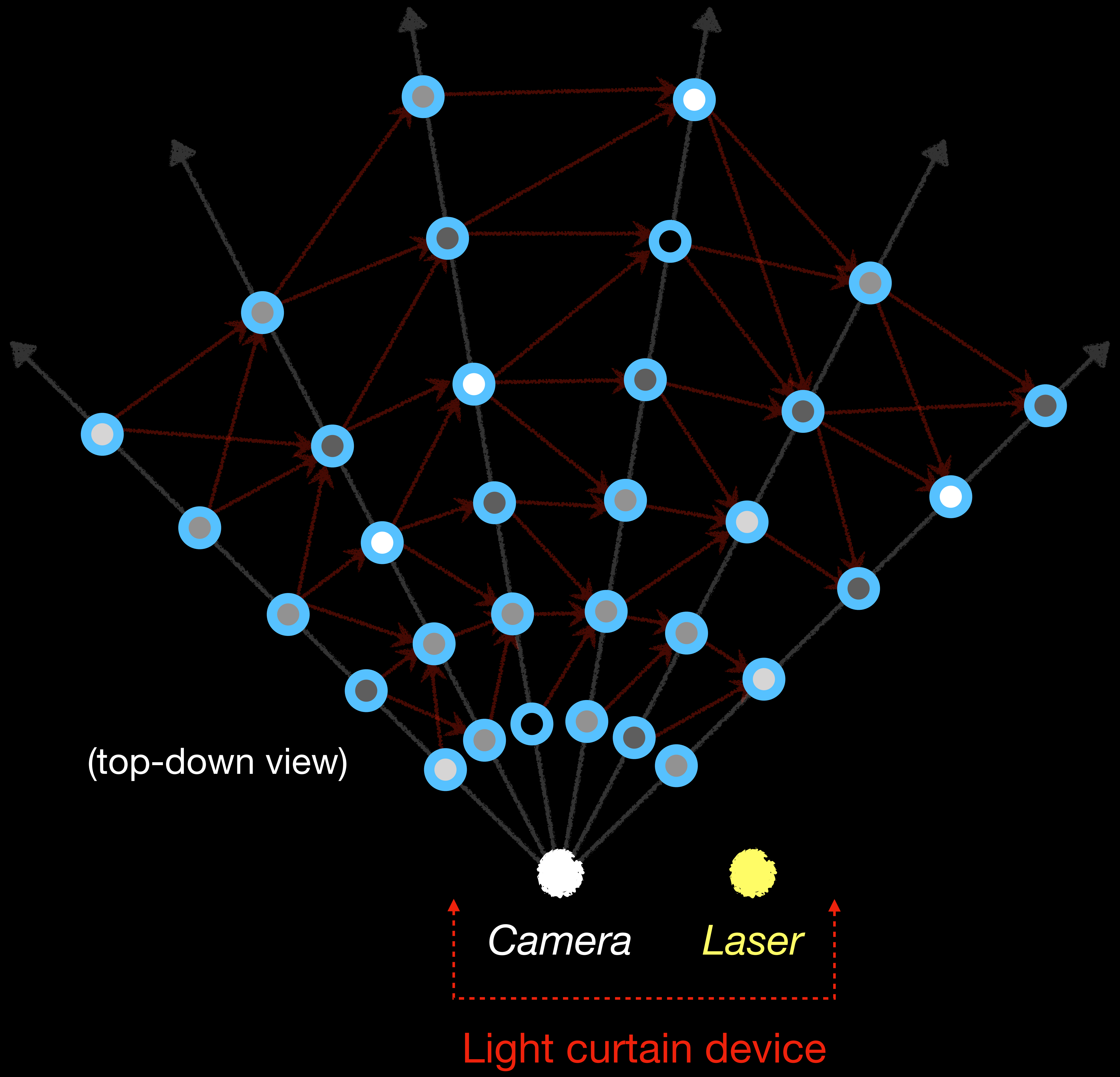


Graph node → *Graph edge*

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its* →
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*
  - *Repeat for all nodes in ray*
- *Repeat over each previous ray*

Dynamic Programming

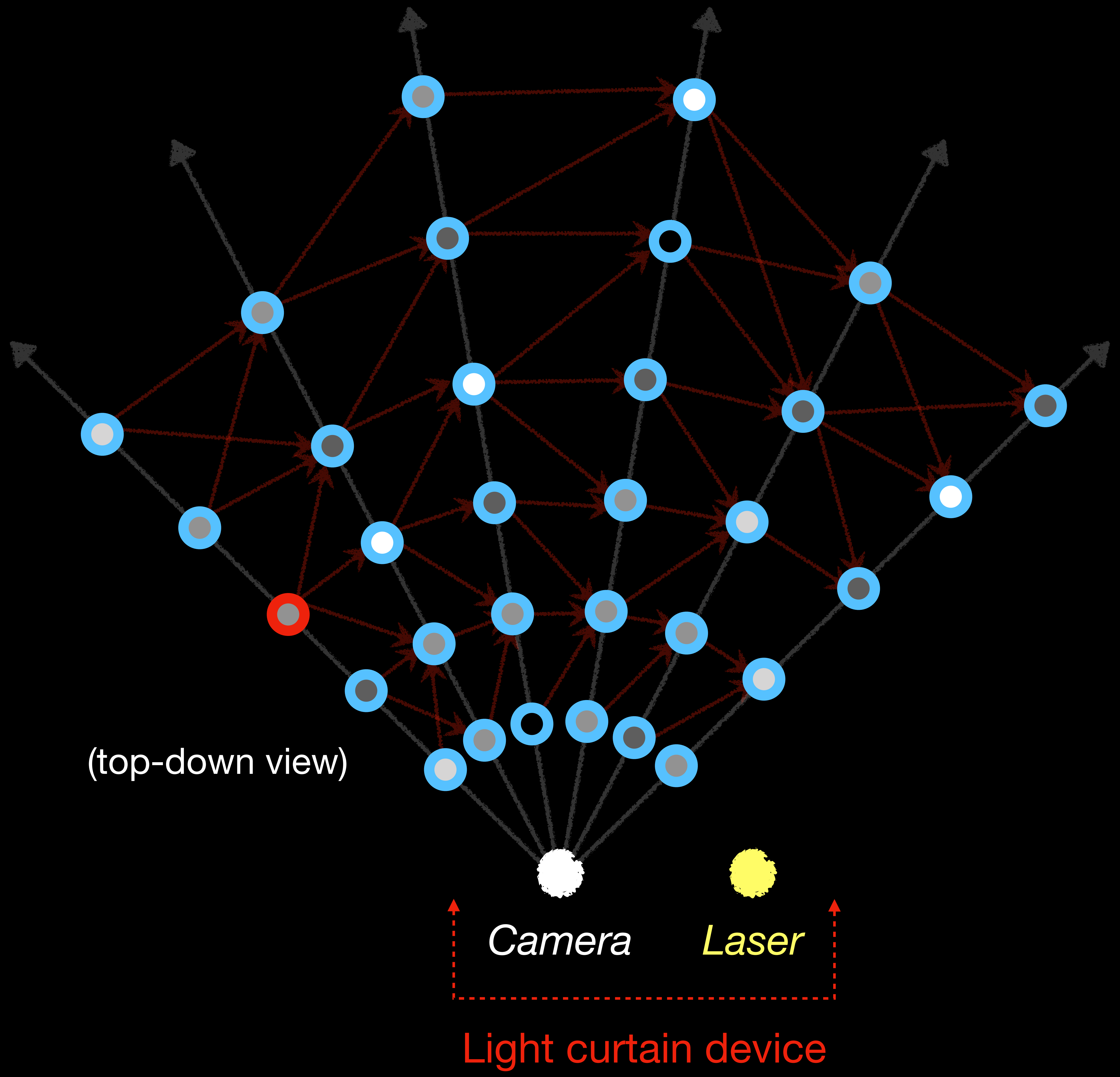


Graph node → Graph edge

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Start from nodes on rightmost ray. Partial curtain starts and ends there.*
- *For each node in the previous ray*
  - *Look at all its →*
  - *Select the one with highest uncertainty. Add own uncertainty to the sum.*
  - *Repeat for all nodes in ray*
- *Repeat over each previous ray*

Dynamic Programming

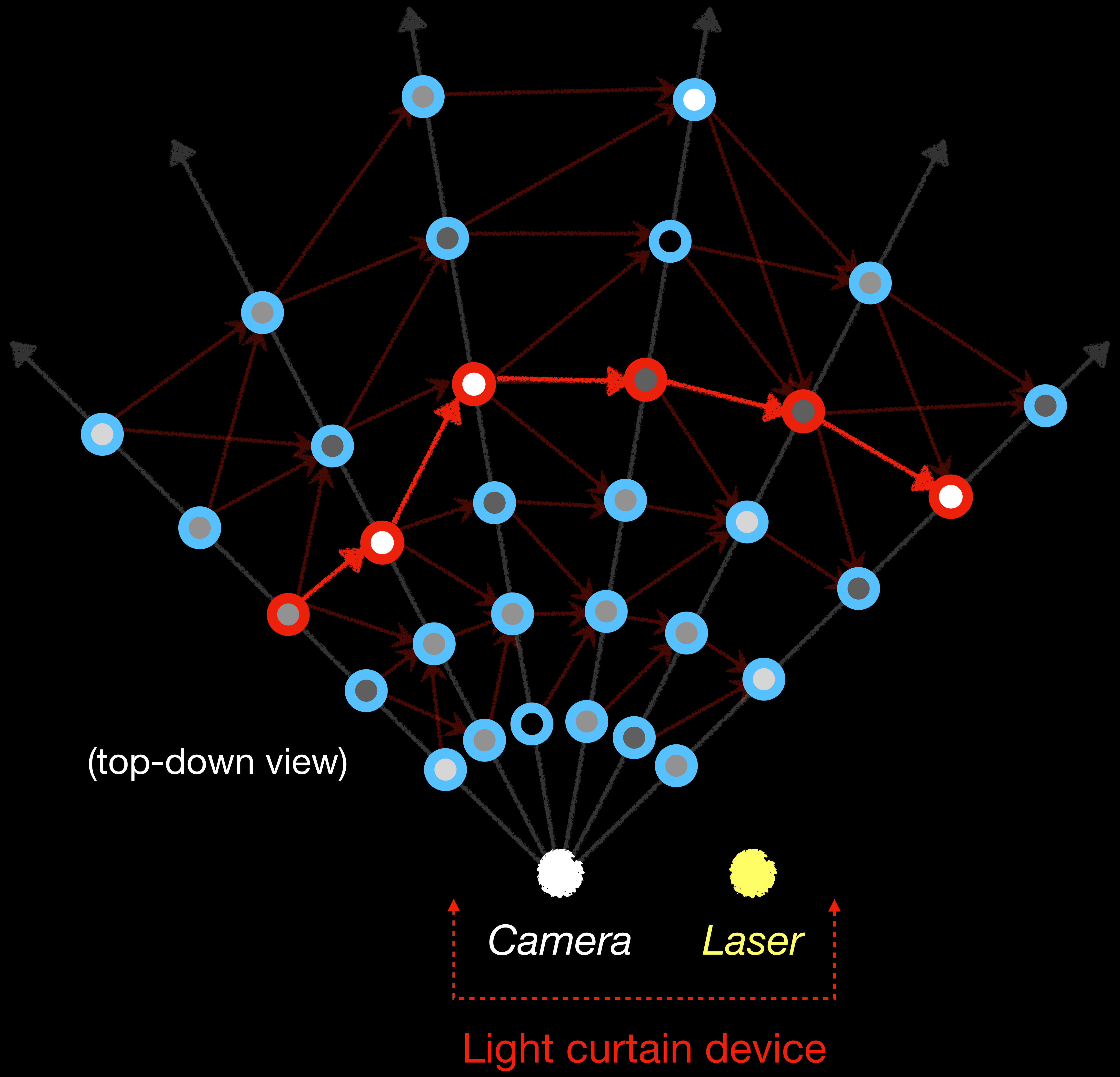


Graph node — Graph edge

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Select the [node] in the first ray that has the highest sum.*

Dynamic Programming



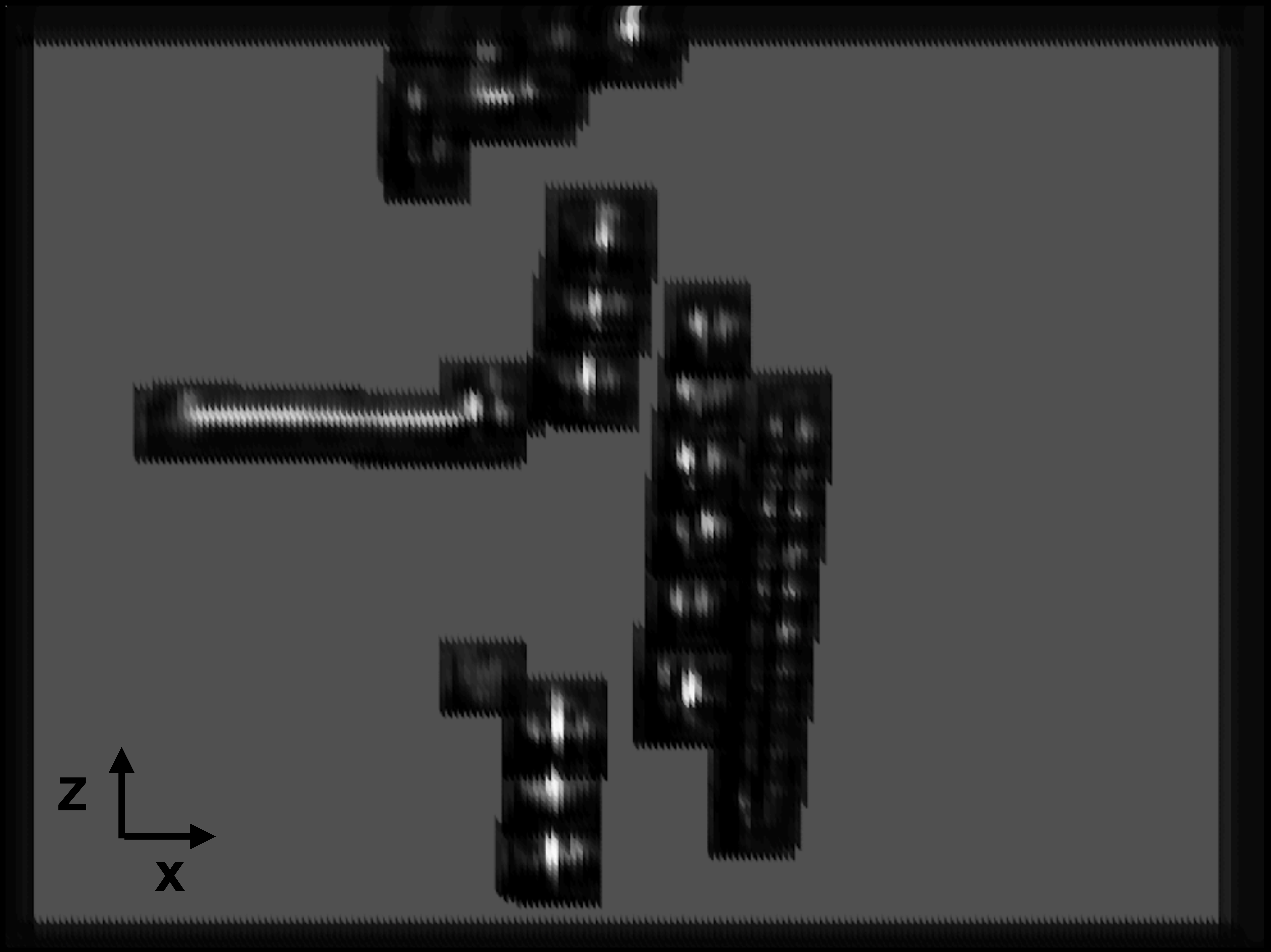
Graph node *Graph edge*

For each node, find the partial curtain starting from that node that maximizes sum of uncertainties

- *Select the ● in the first ray that has the highest sum.*
- *Backtrack connecting each node to its best neighbor.*

This is the light curtain that maximizes the sum of uncertainties!

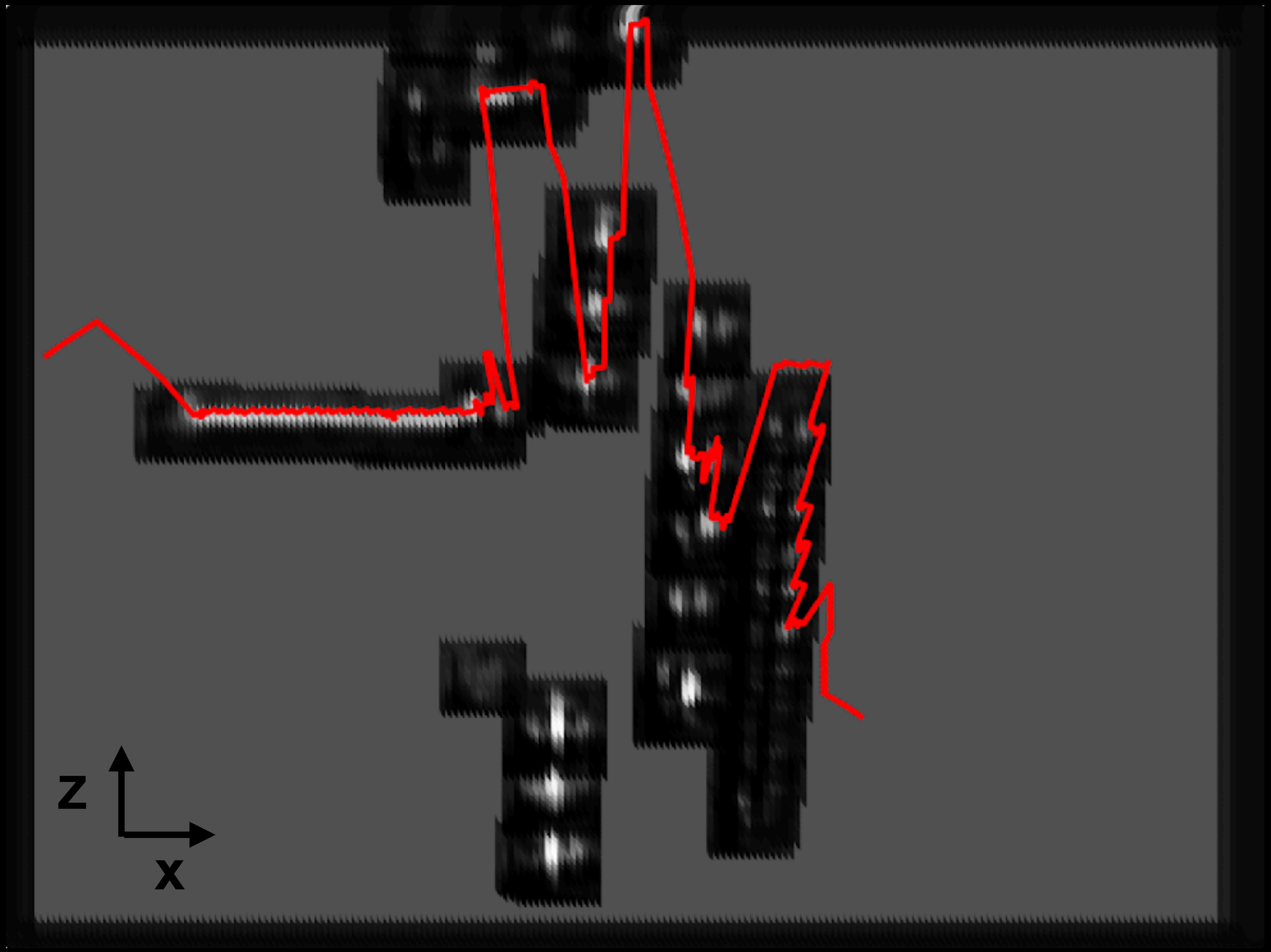
Result



Uncertainty map

(top-down view)

Result

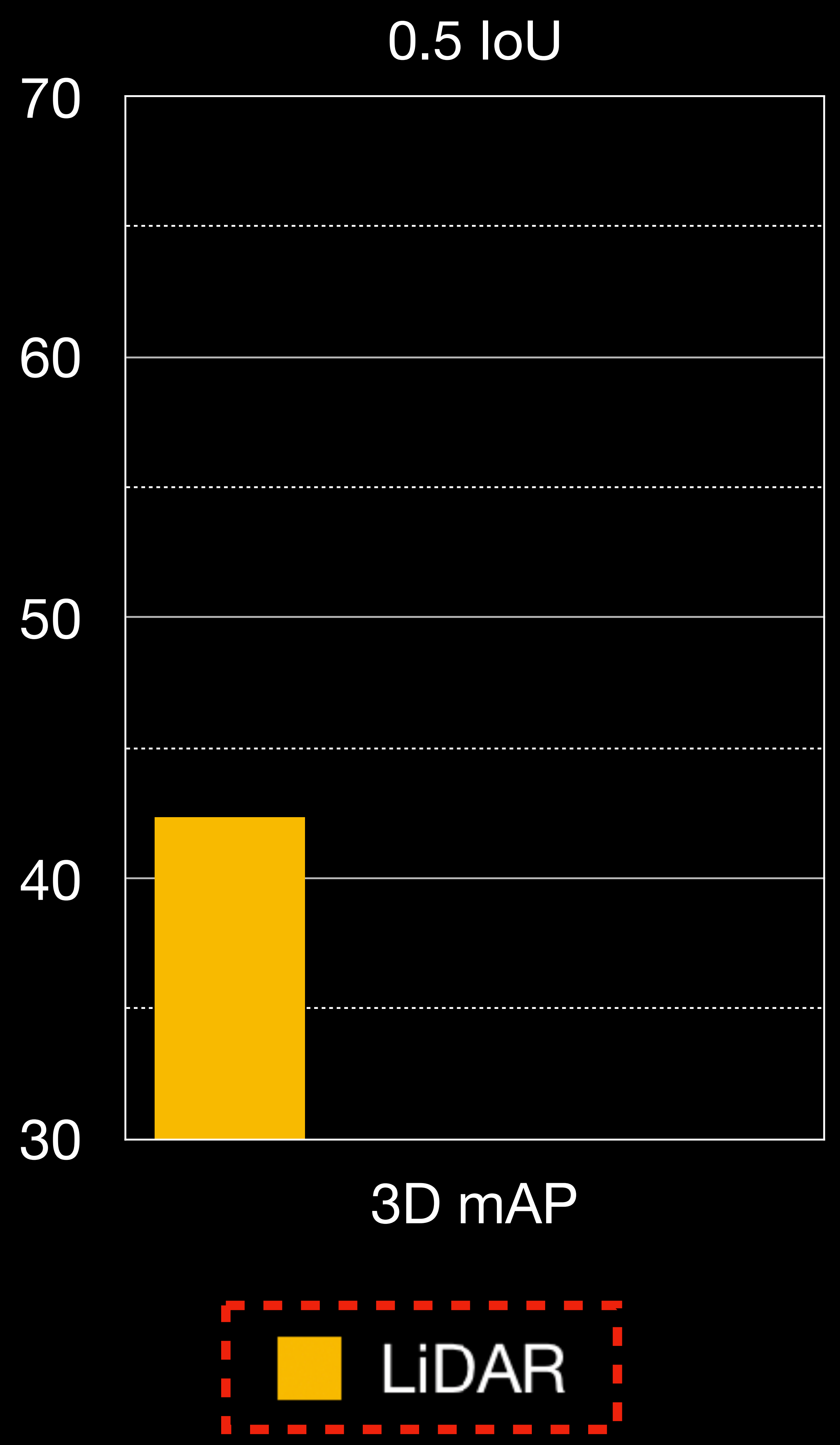


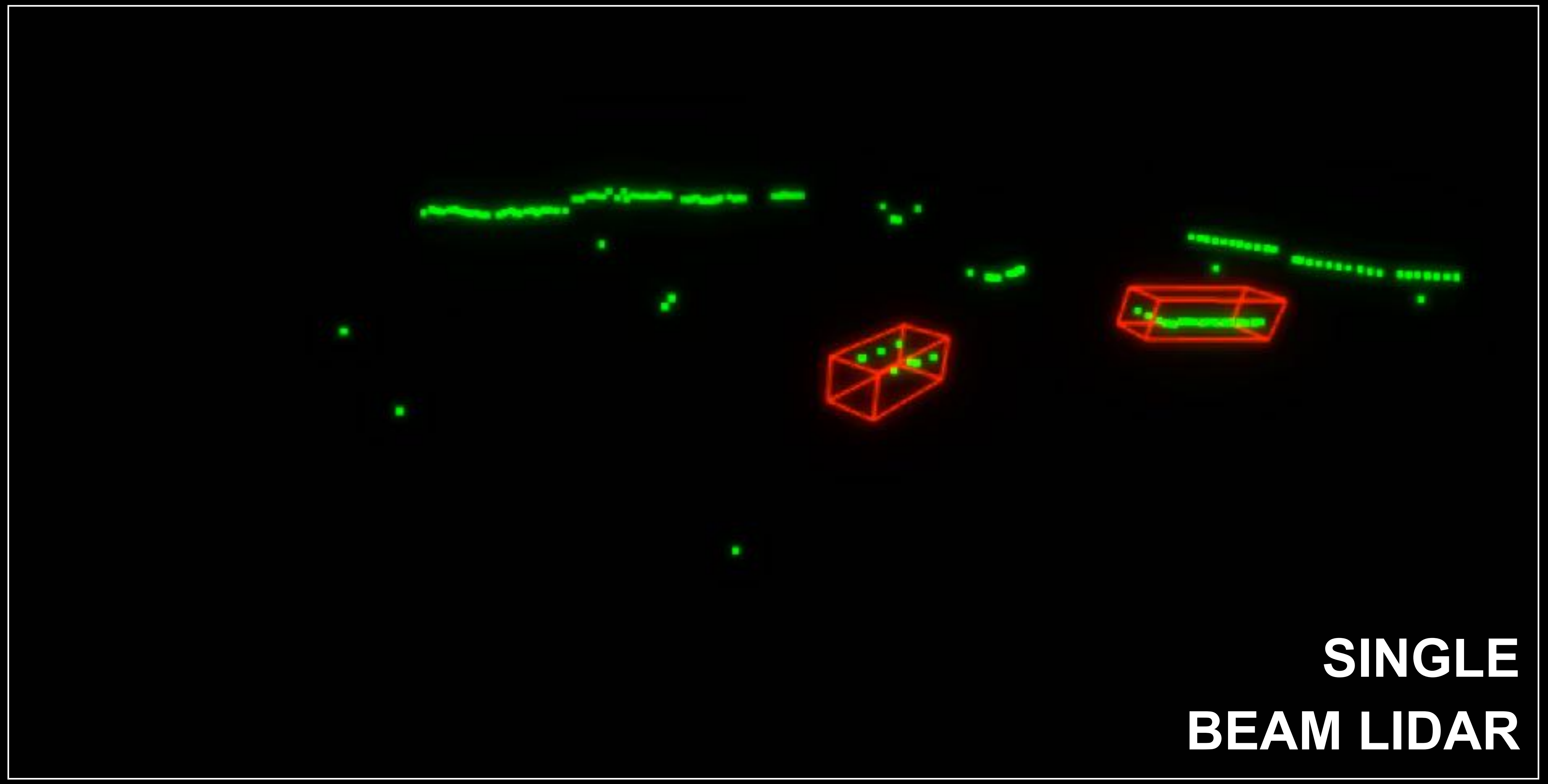
Uncertainty map

(top-down view)

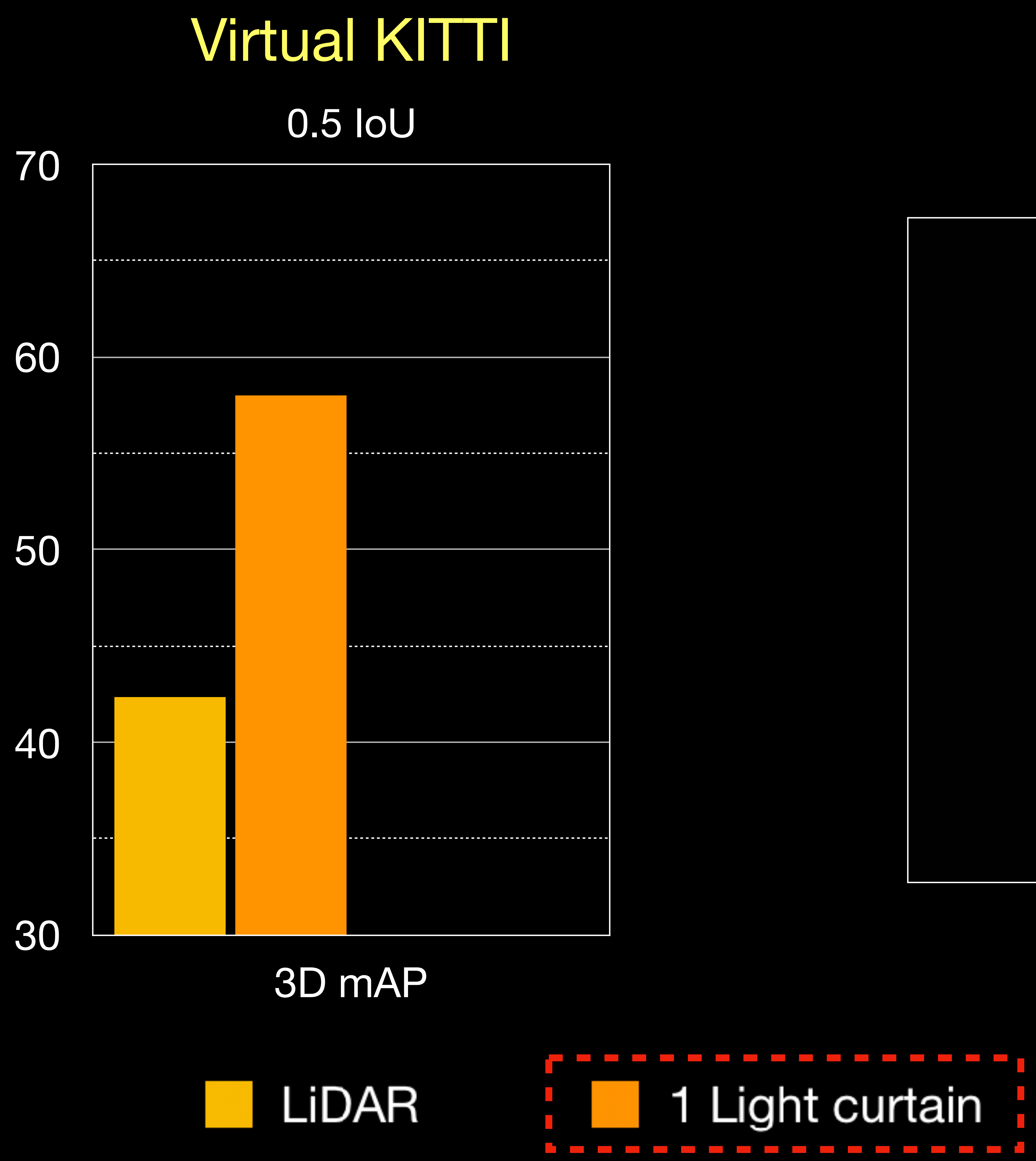
Successive light curtain placements improve detection performance

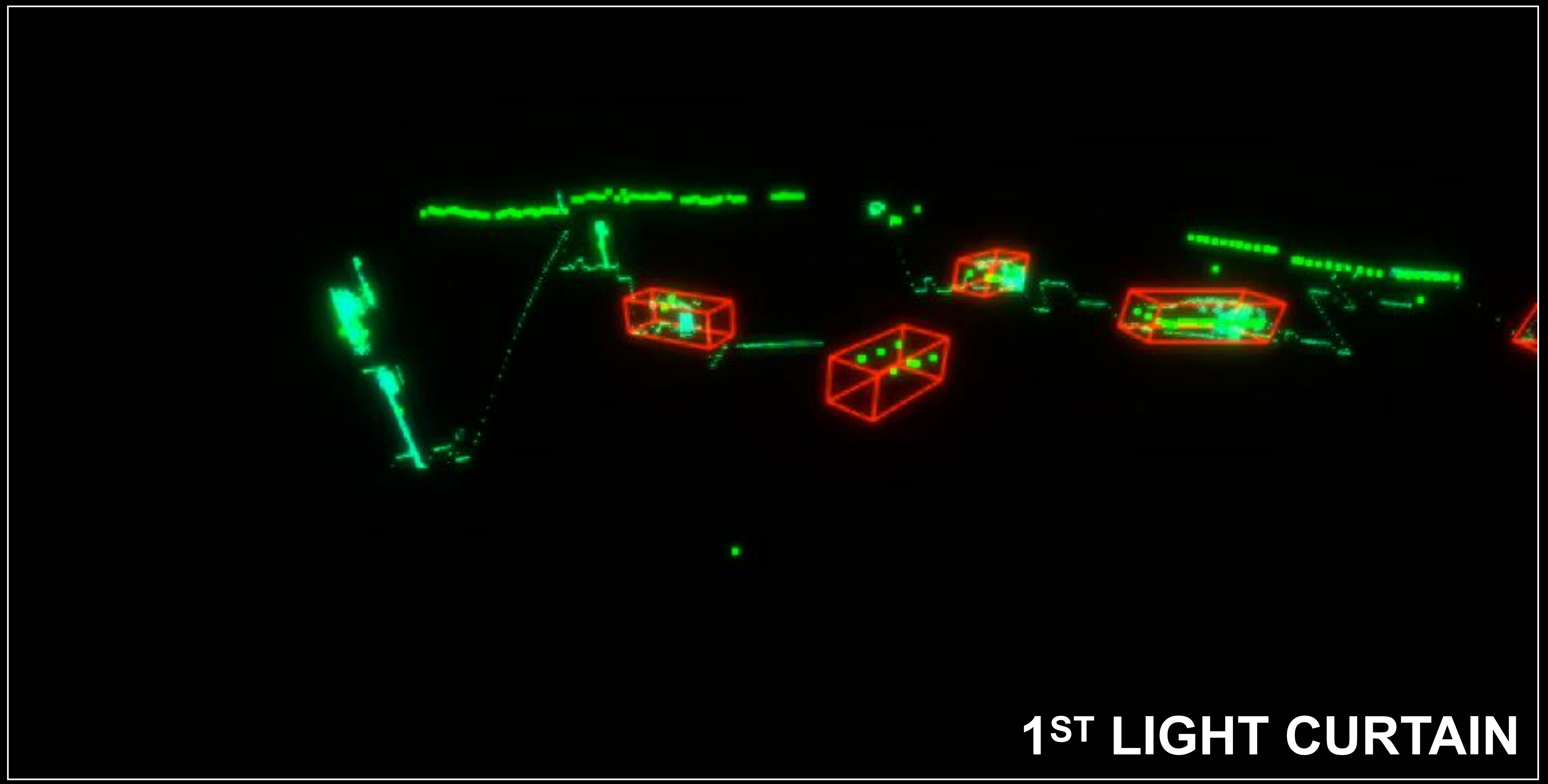
Virtual KITTI





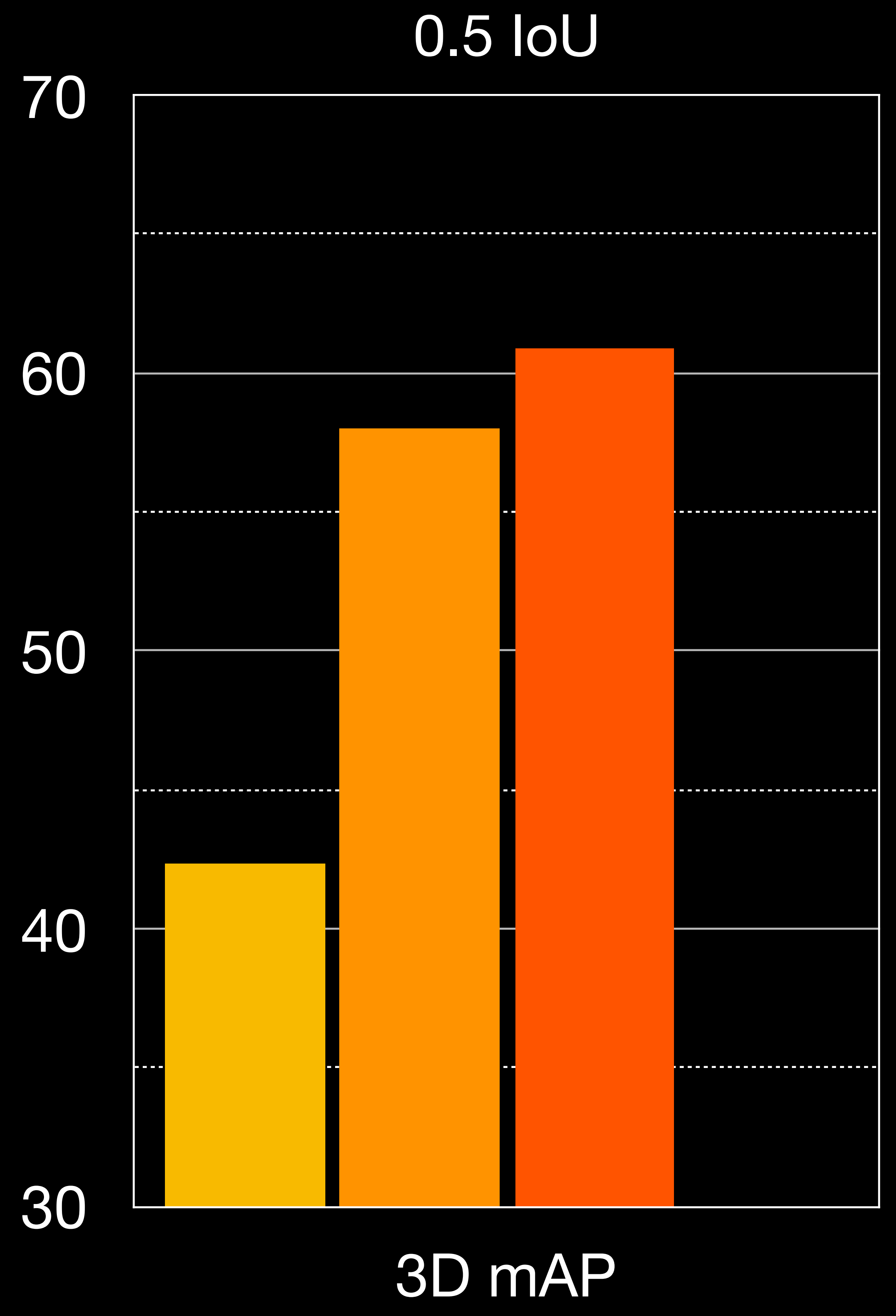
Successive light curtain placements improve detection performance

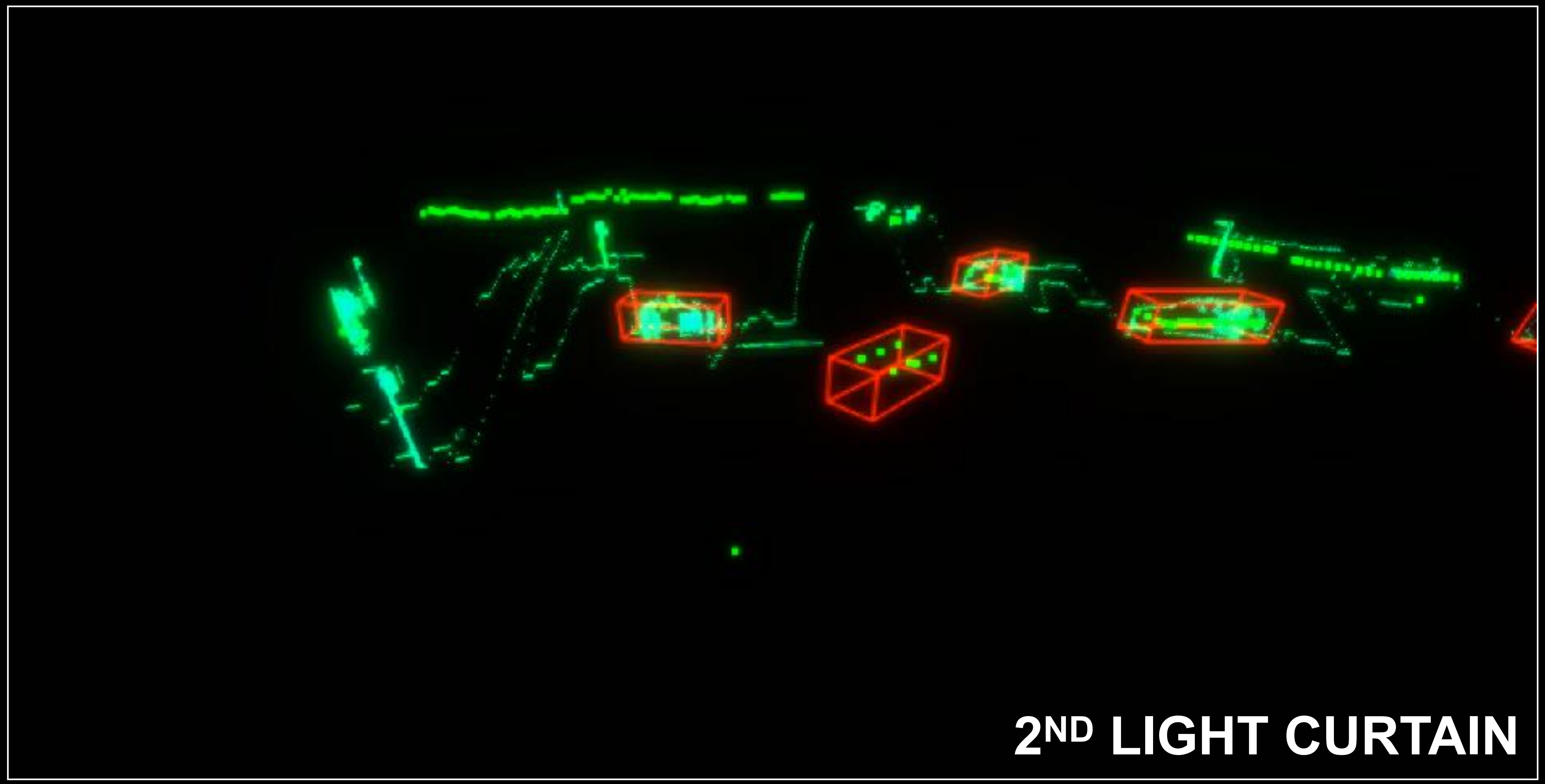




Successive light curtain placements improve detection performance

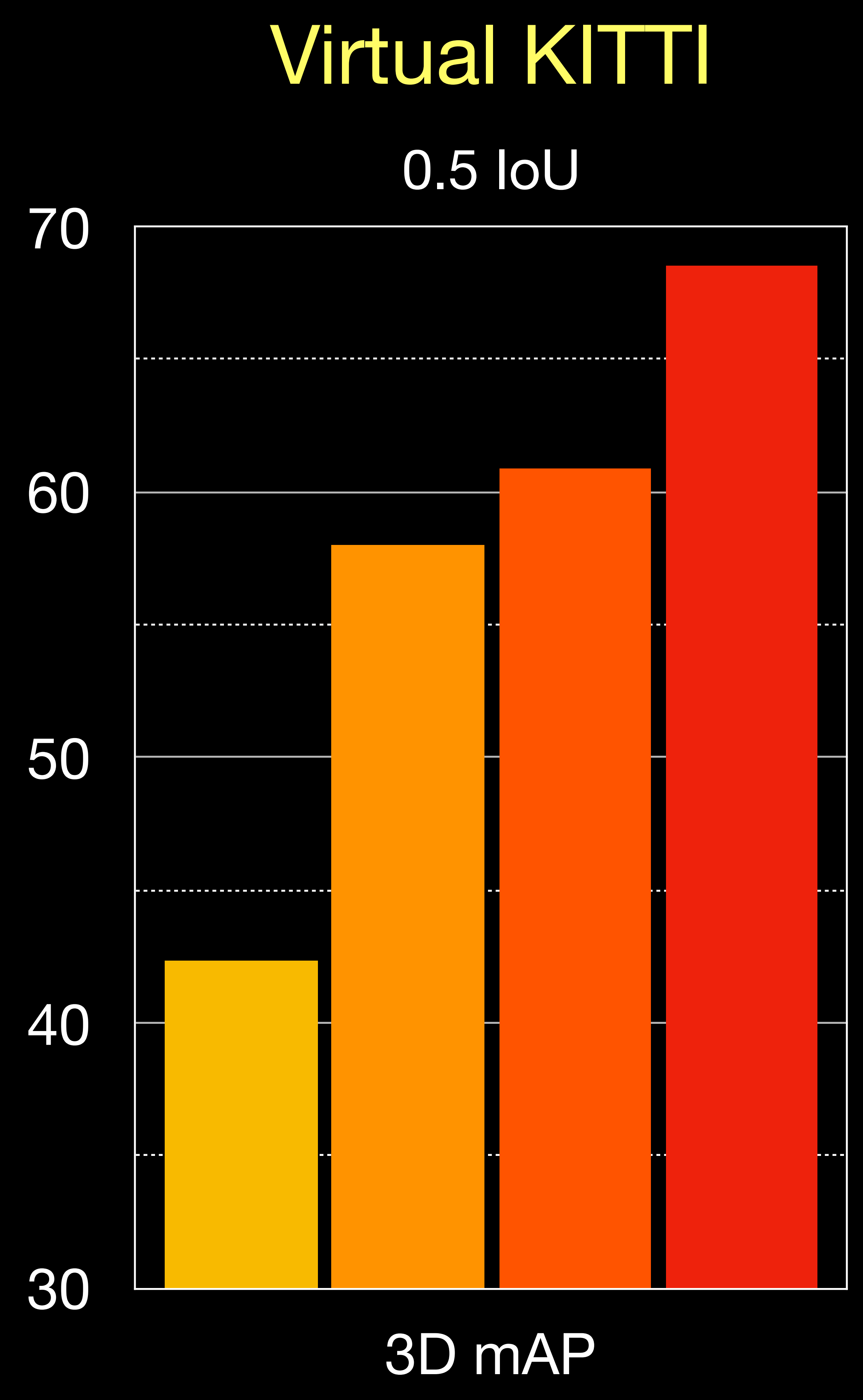
Virtual KITTI

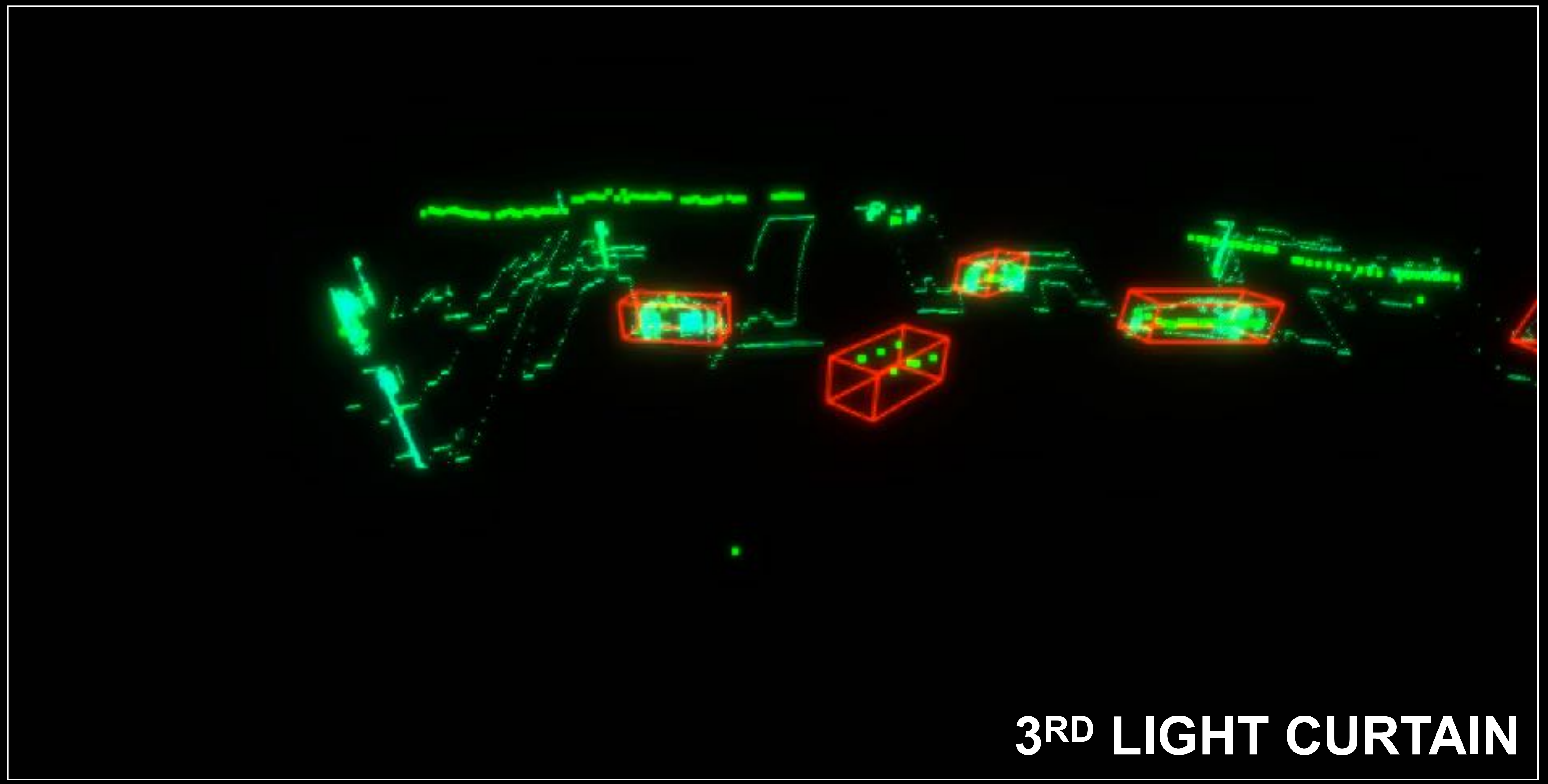




LiDAR   1 Light curtain   2 Light curtains

Successive light curtain placements improve detection performance



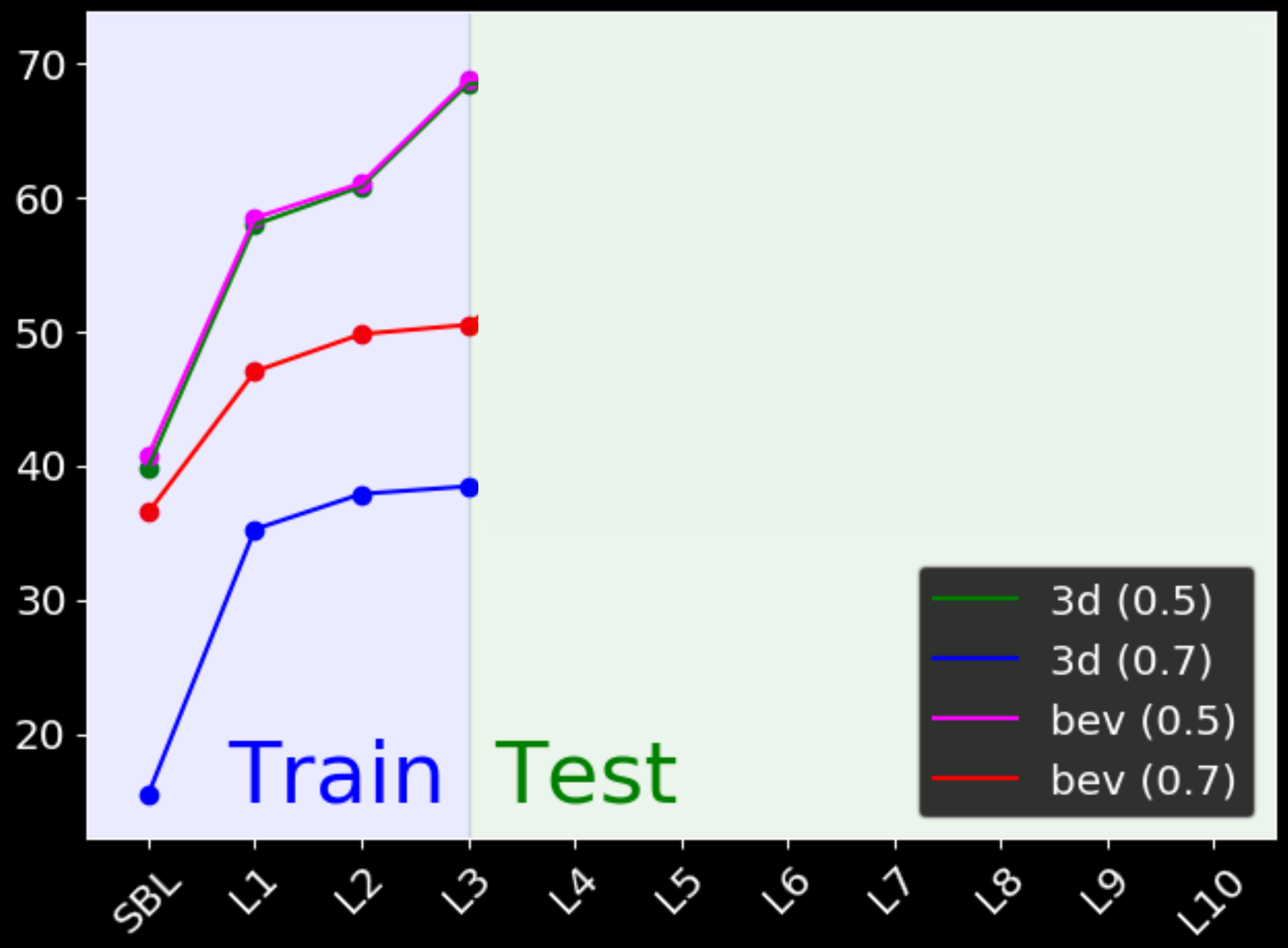


LiDAR 1 Light curtain 2 Light curtains 3 Light curtains

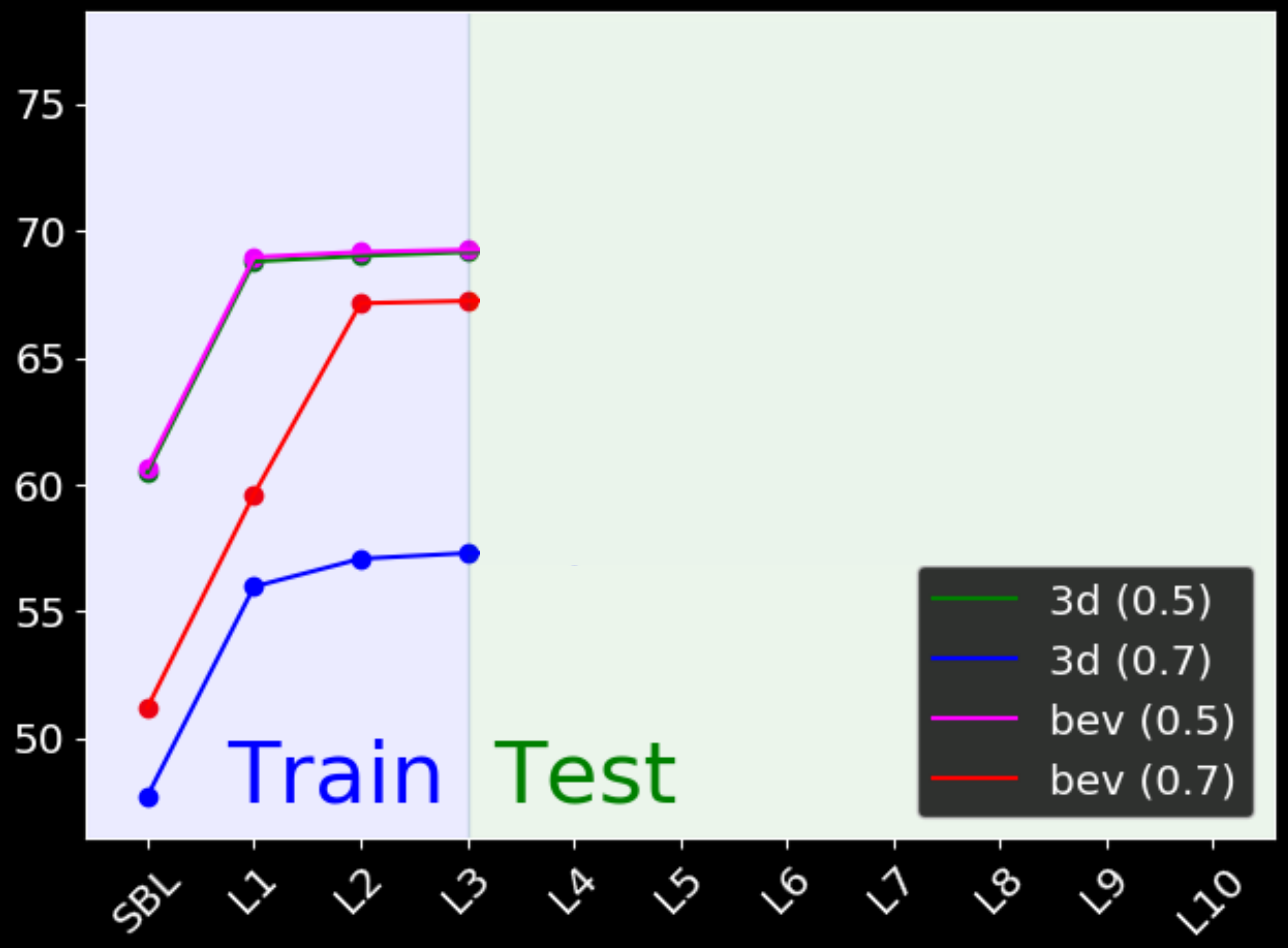
Successive light curtain placements improve detection performance

Performance generalizes to additional curtains

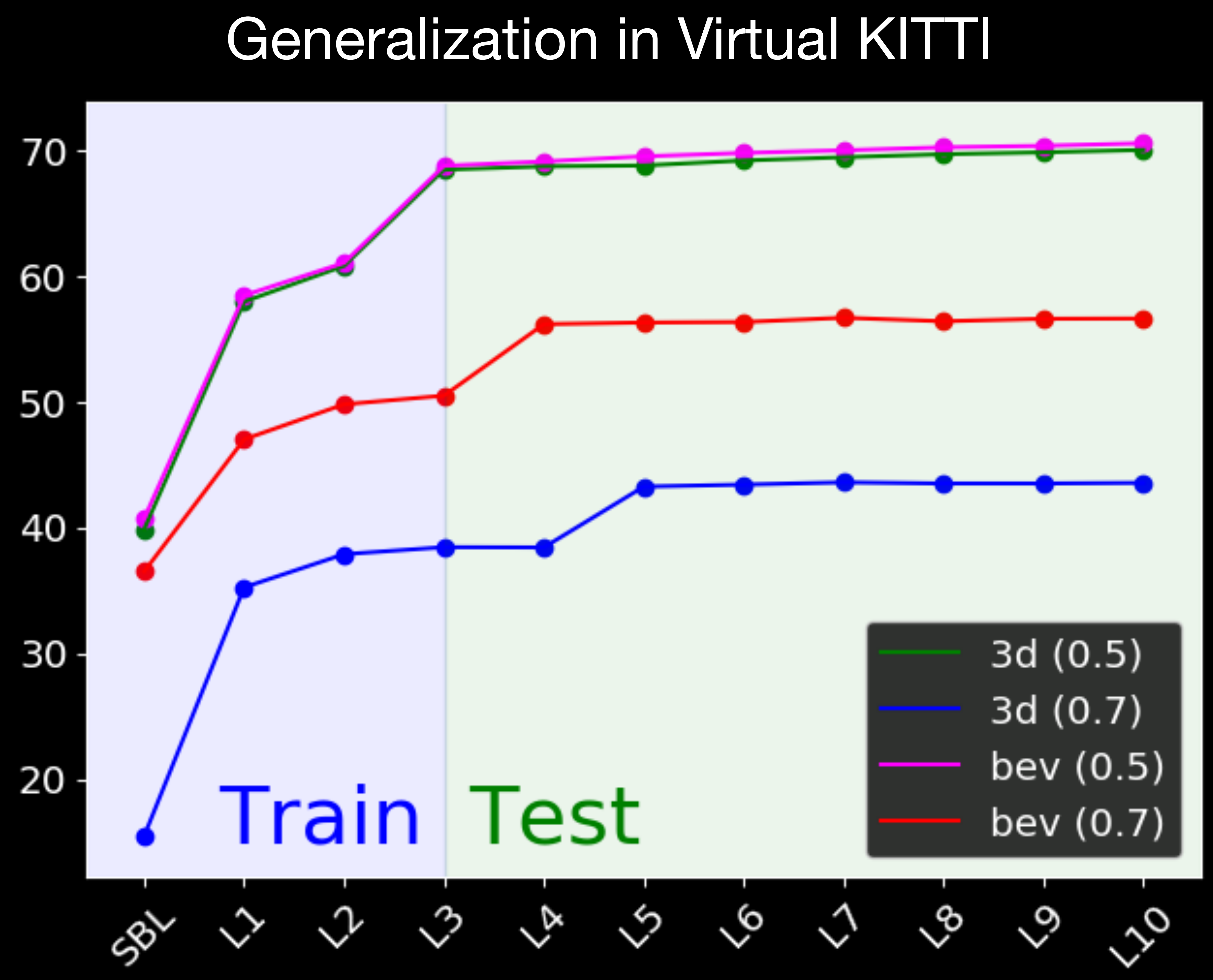
Generalization in Virtual KITTI

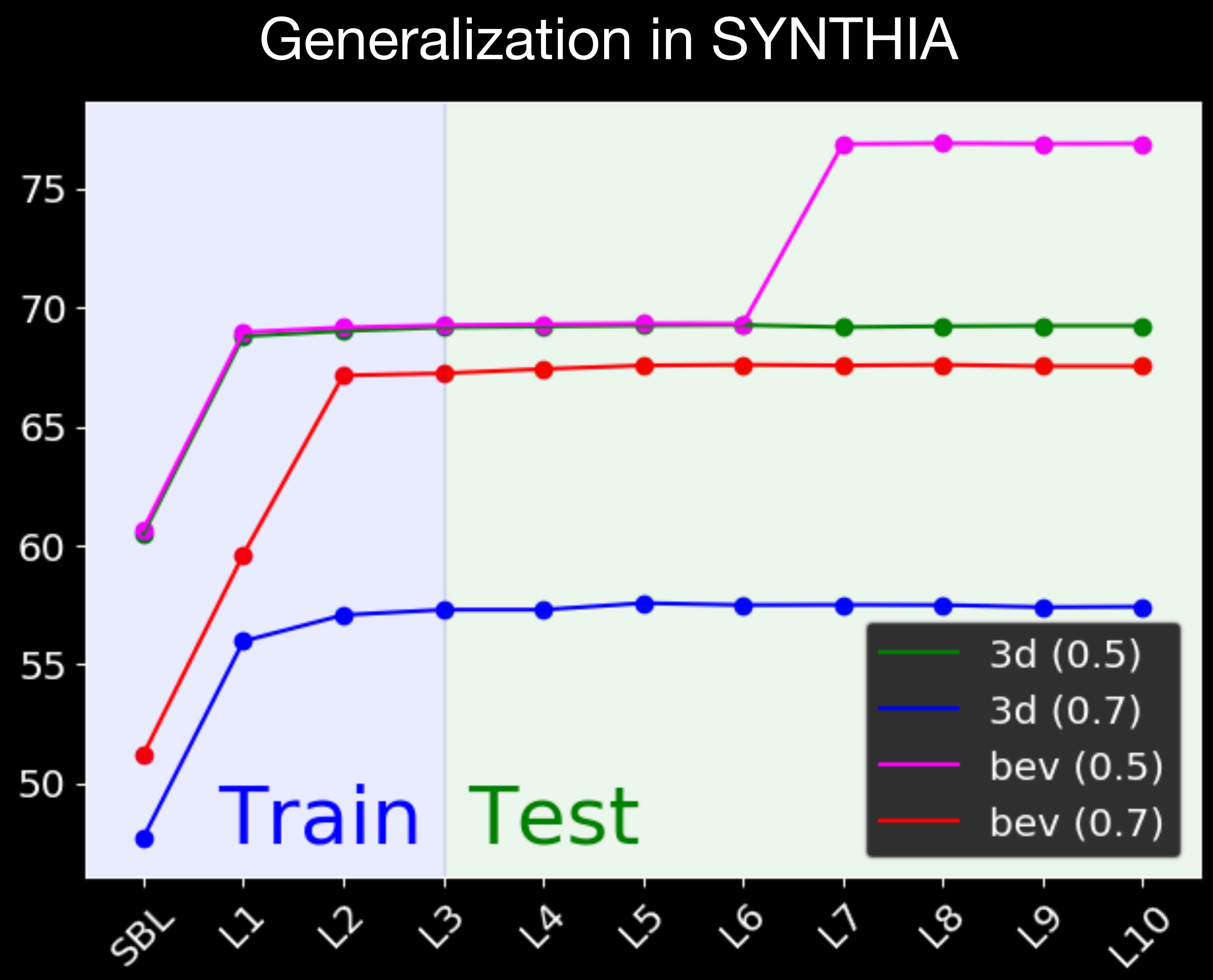


Generalization in SYNTHIA

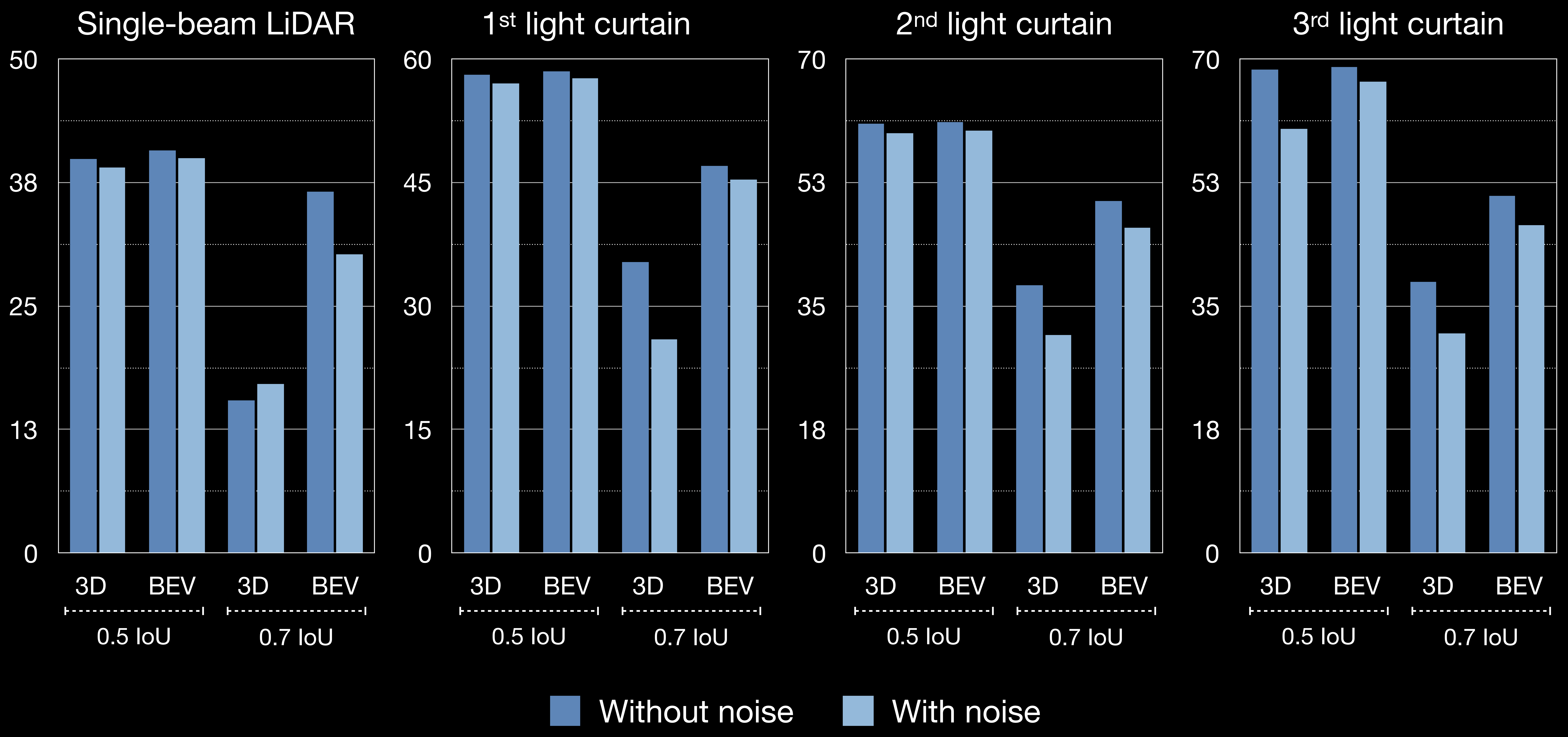


Performance generalizes to additional curtains

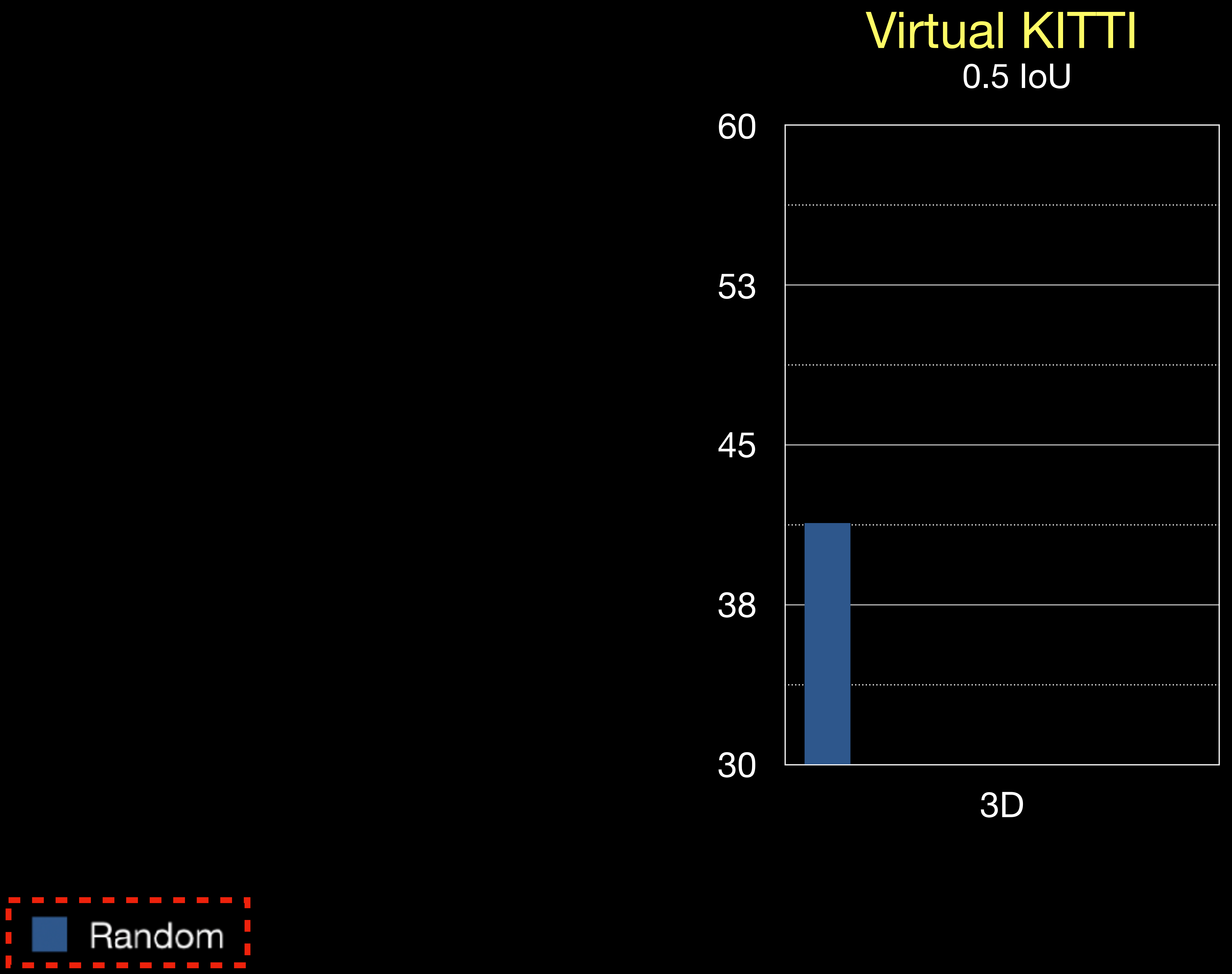




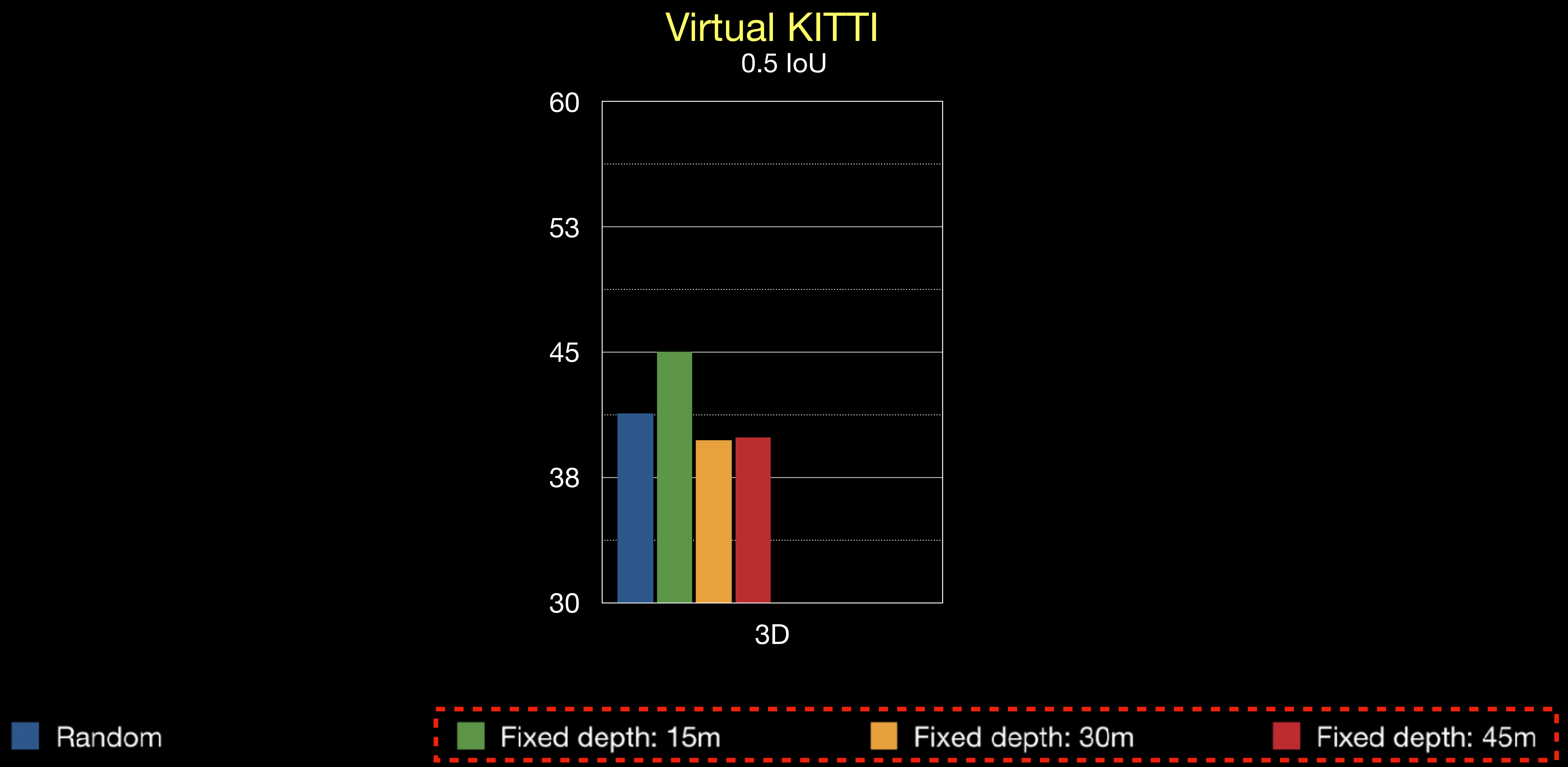
Performance generalizes to additional noise



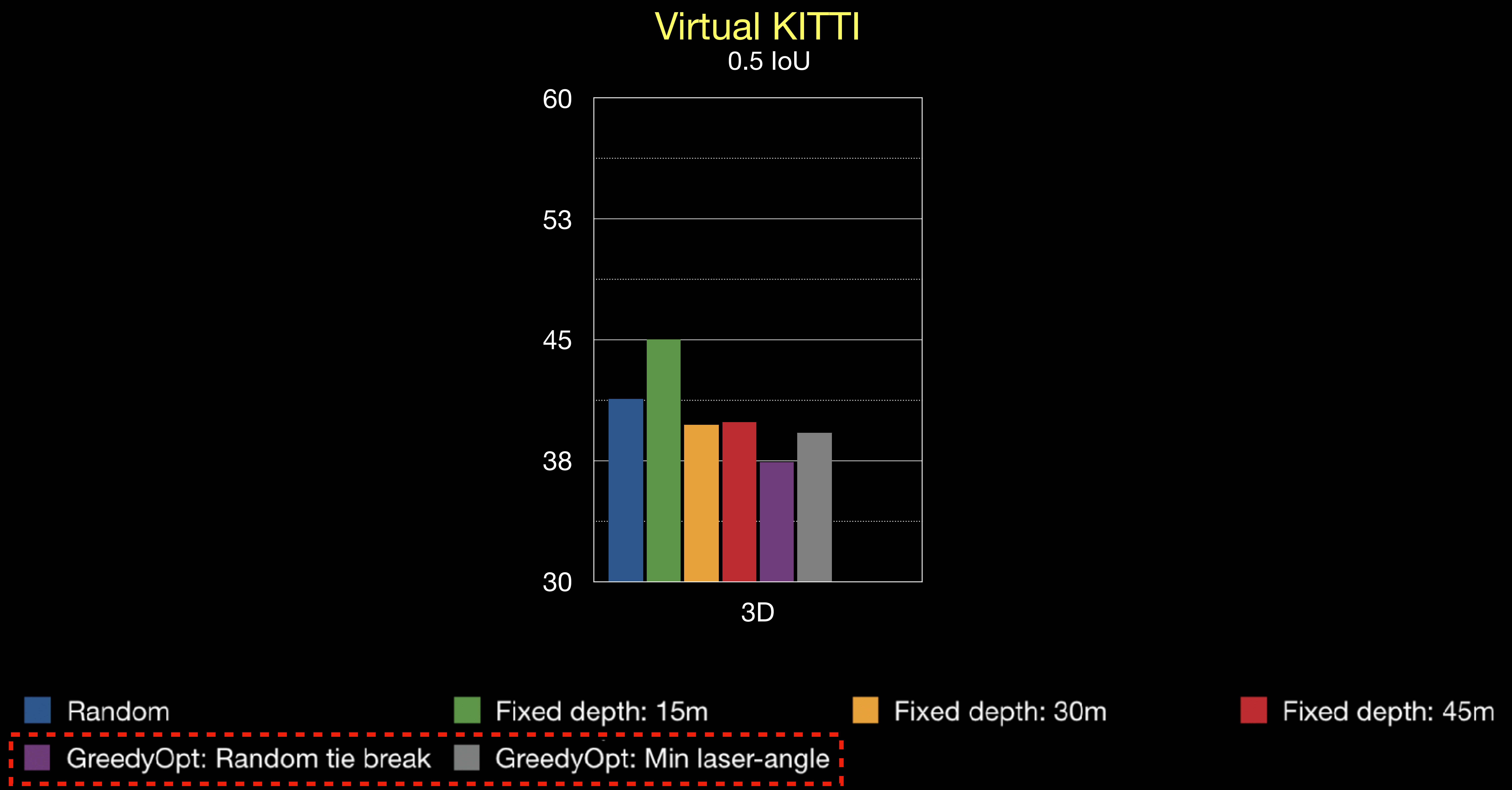
Comparison to baselines



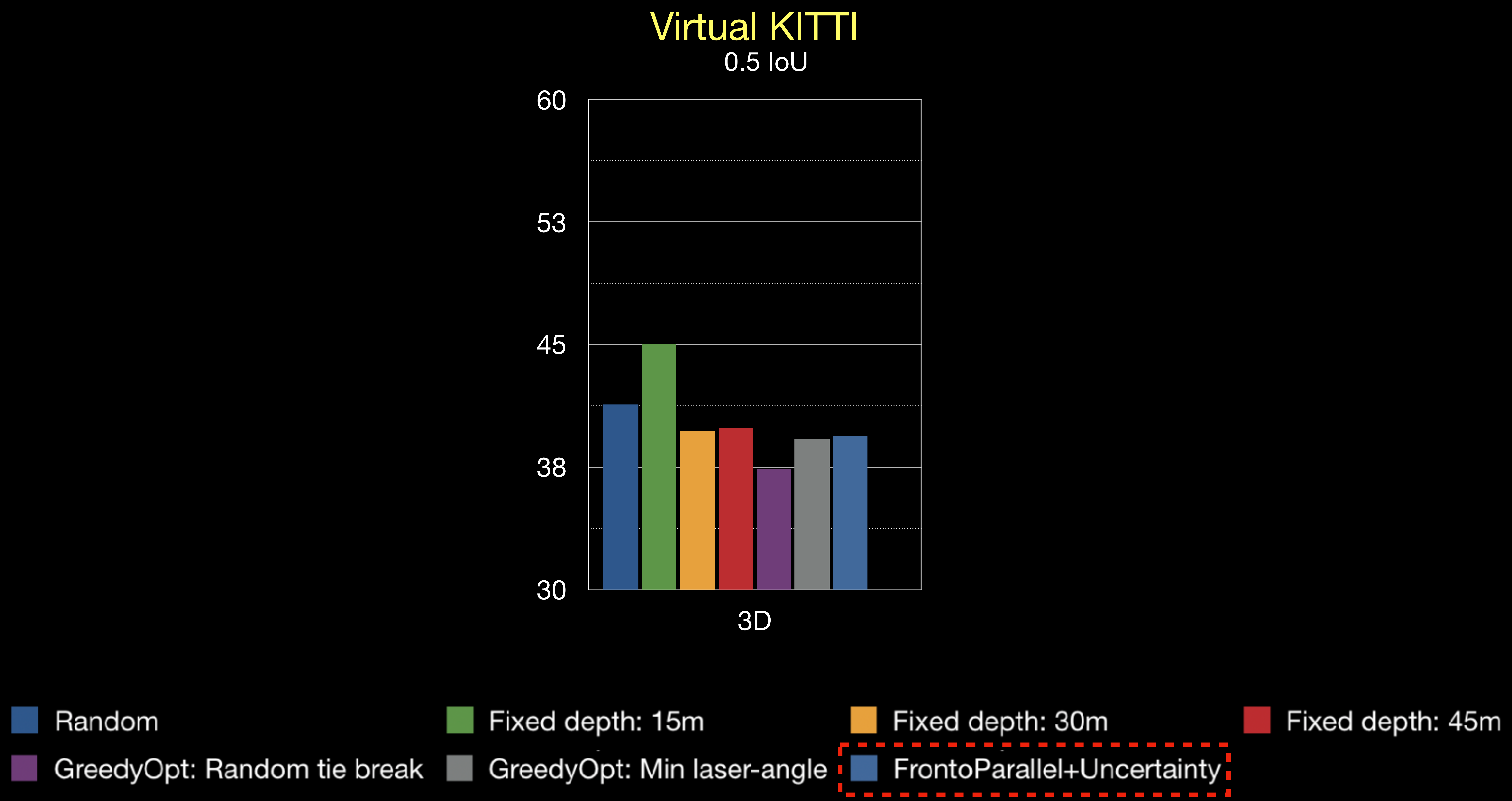
Comparison to baselines



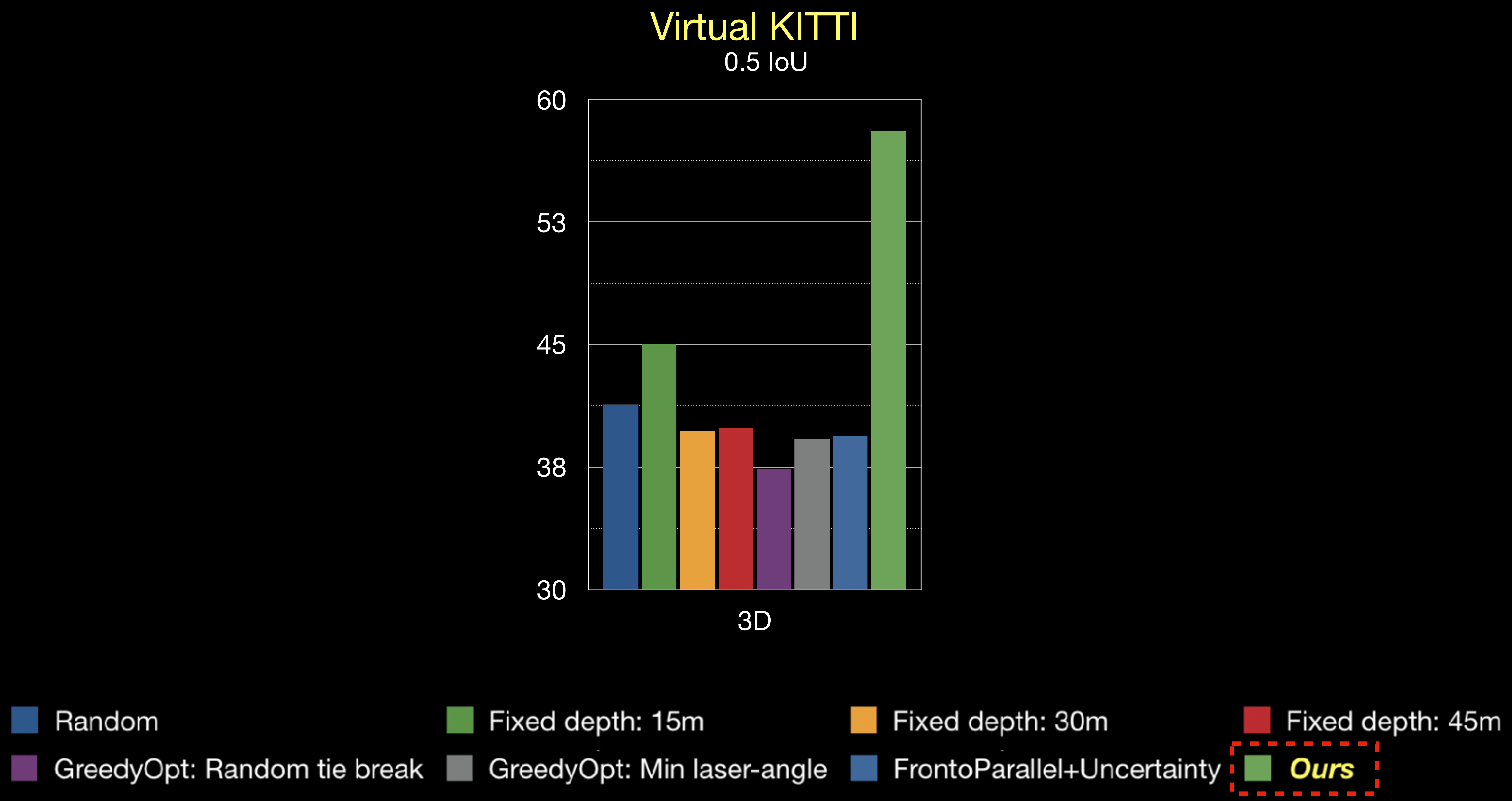
Comparison to baselines



Comparison to baselines



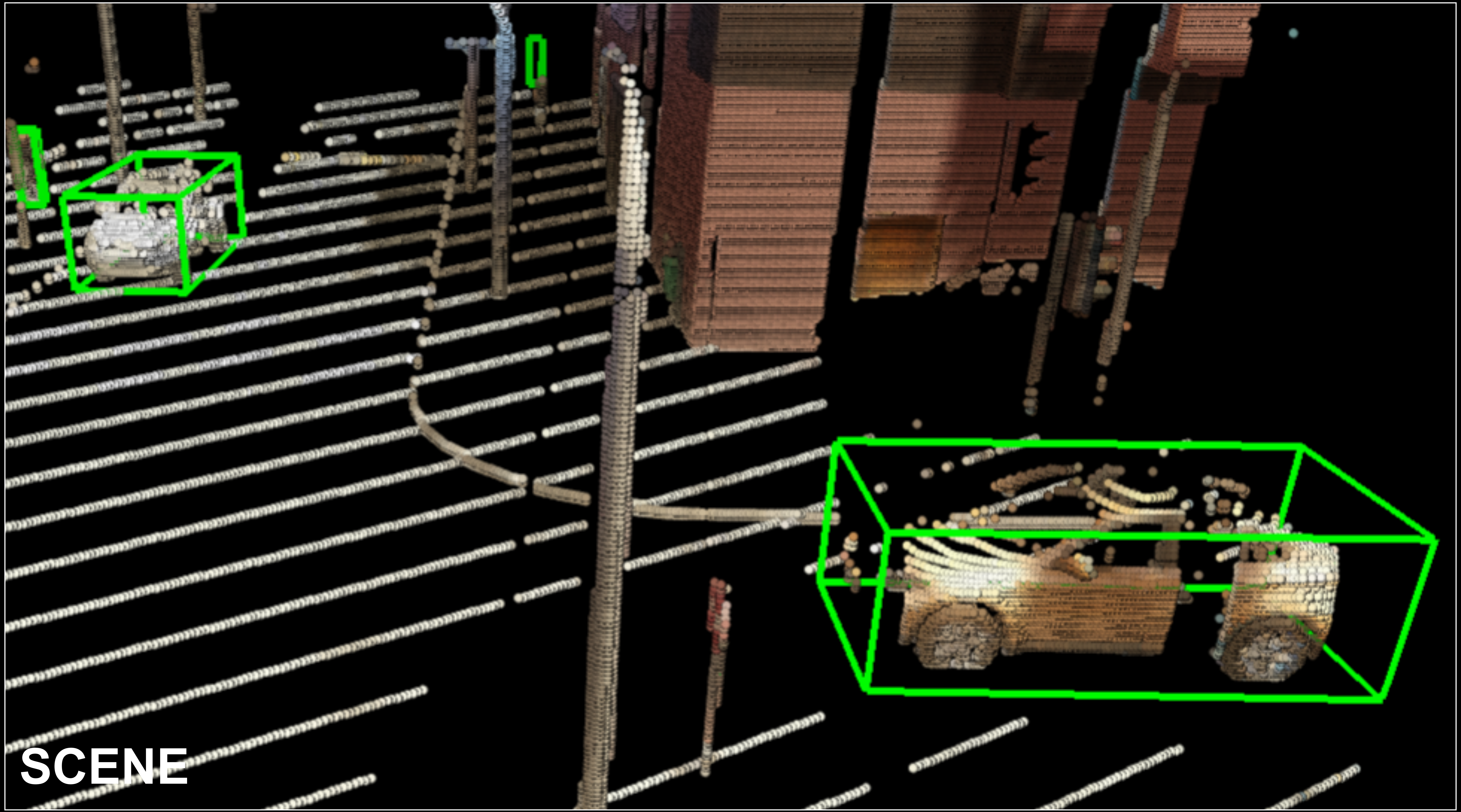
Comparison to baselines



Comparison to baselines

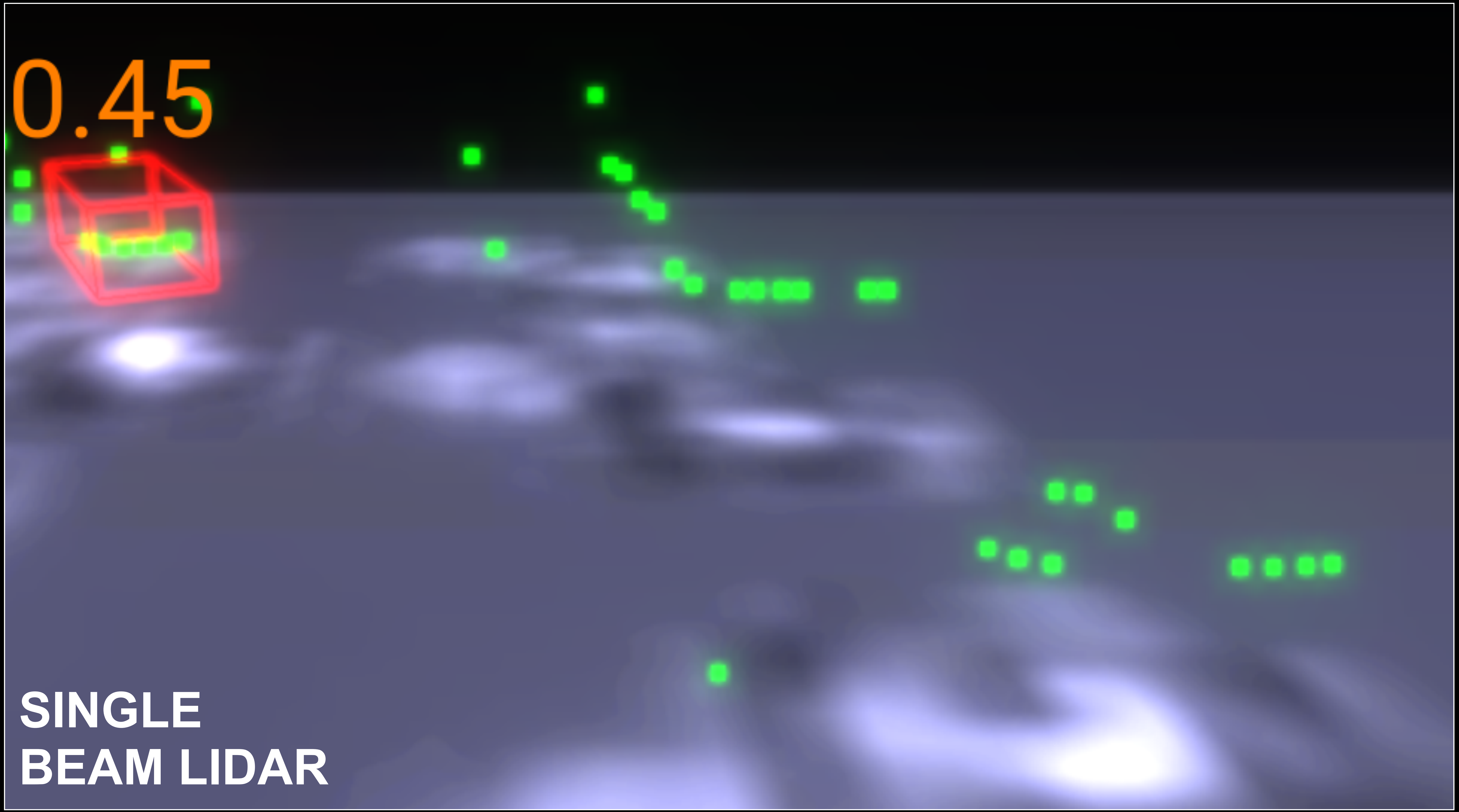
Detecting false negatives missed by LiDAR

(zoomed in)



Detecting false negatives missed by LiDAR

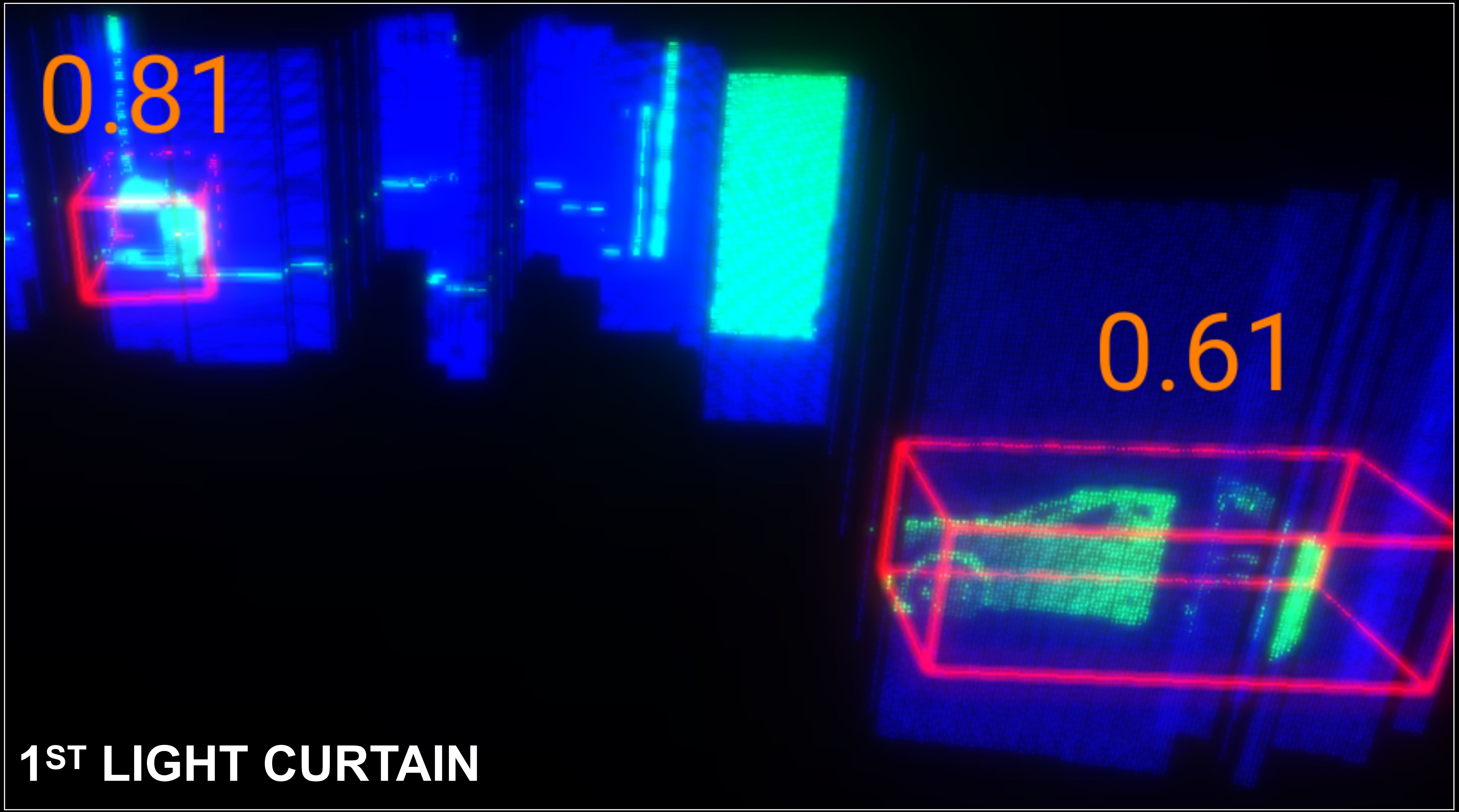
(zoomed in)



*White: more uncertain   Black: less uncertain

Detecting false negatives missed by LiDAR

(zoomed in)

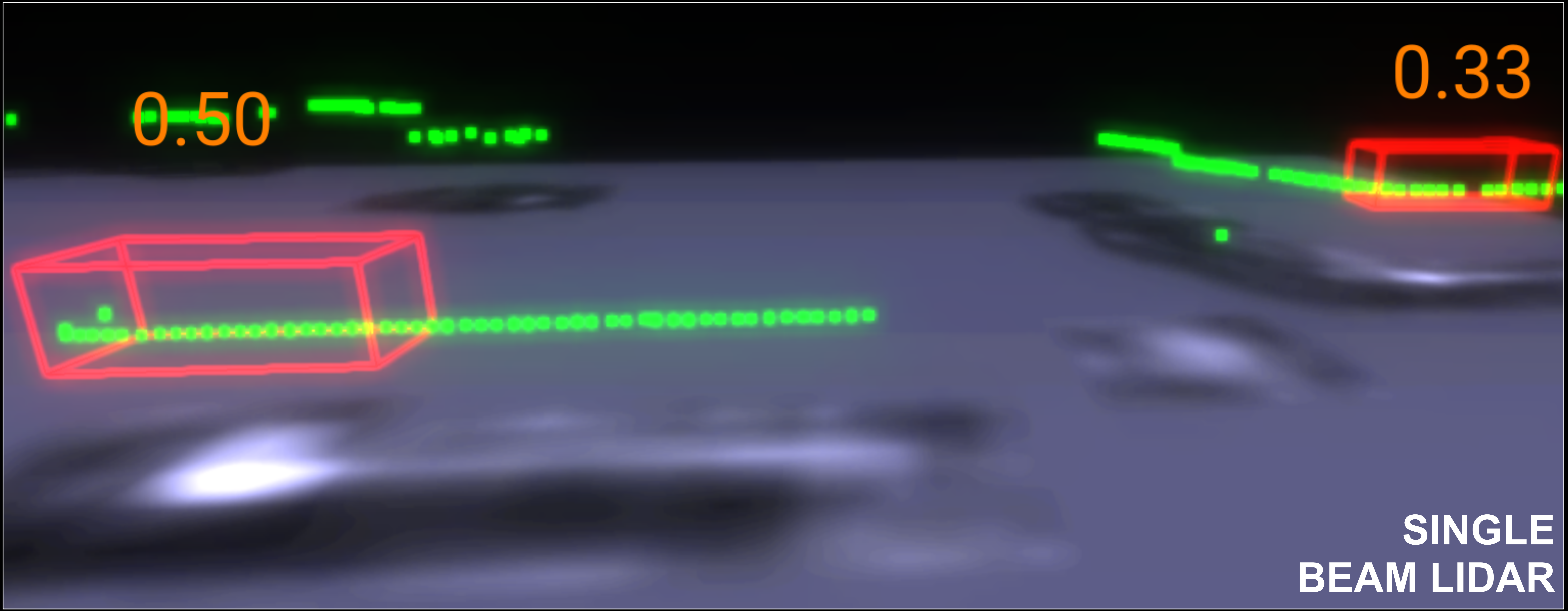


Removing false positives detected by LiDAR

(zoomed in)

Removing false positives detected by LiDAR

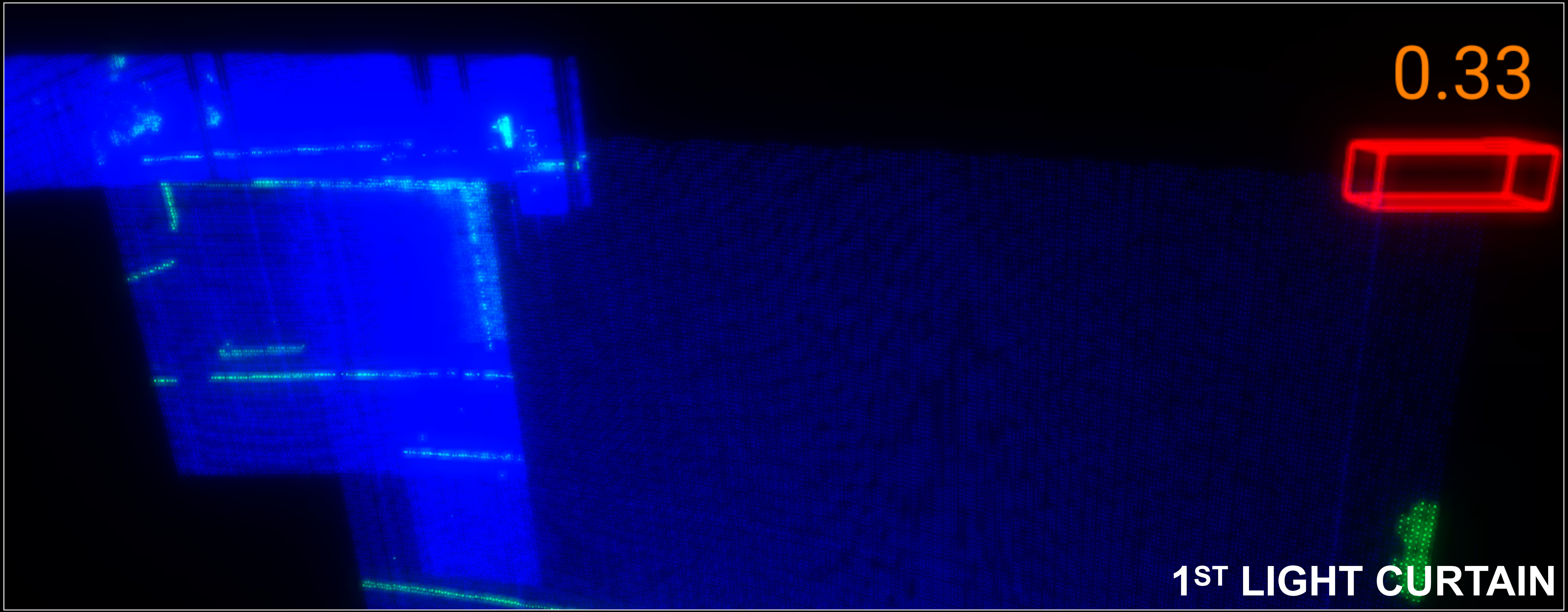
(zoomed in)



*White: more uncertain   Black: less uncertain

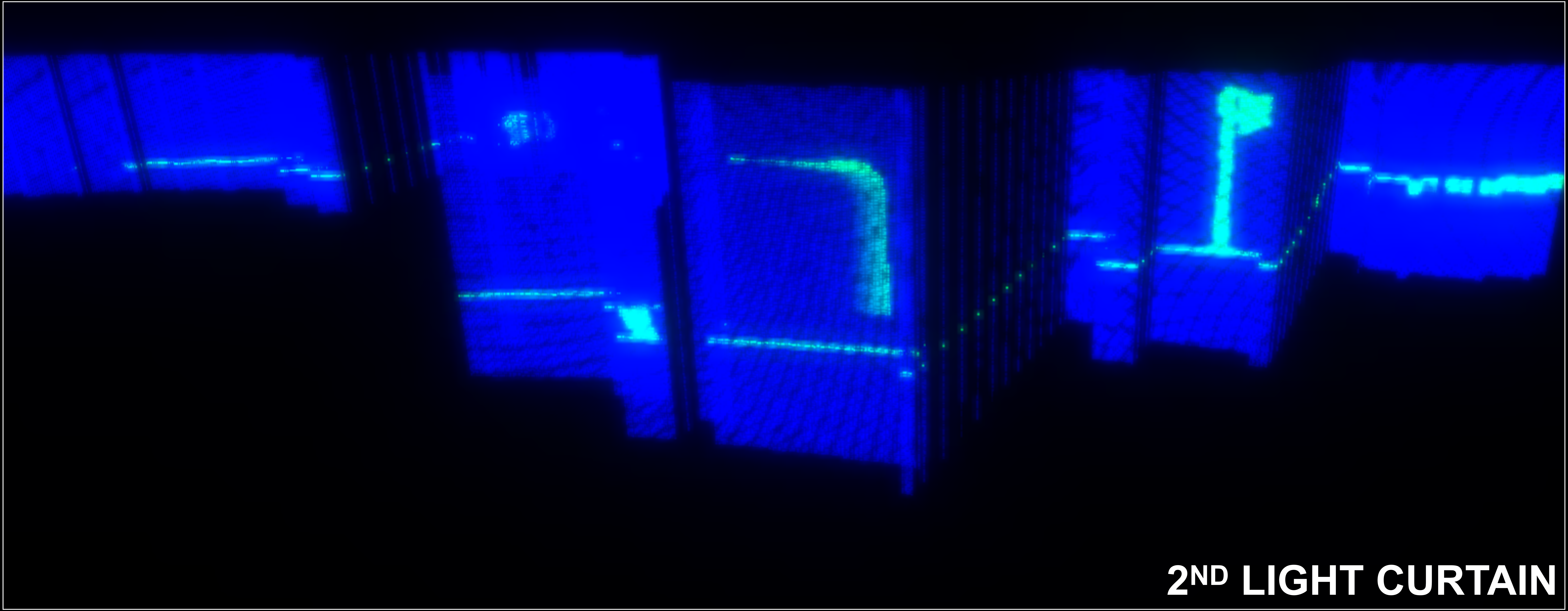
Removing false positives detected by LiDAR

(zoomed in)



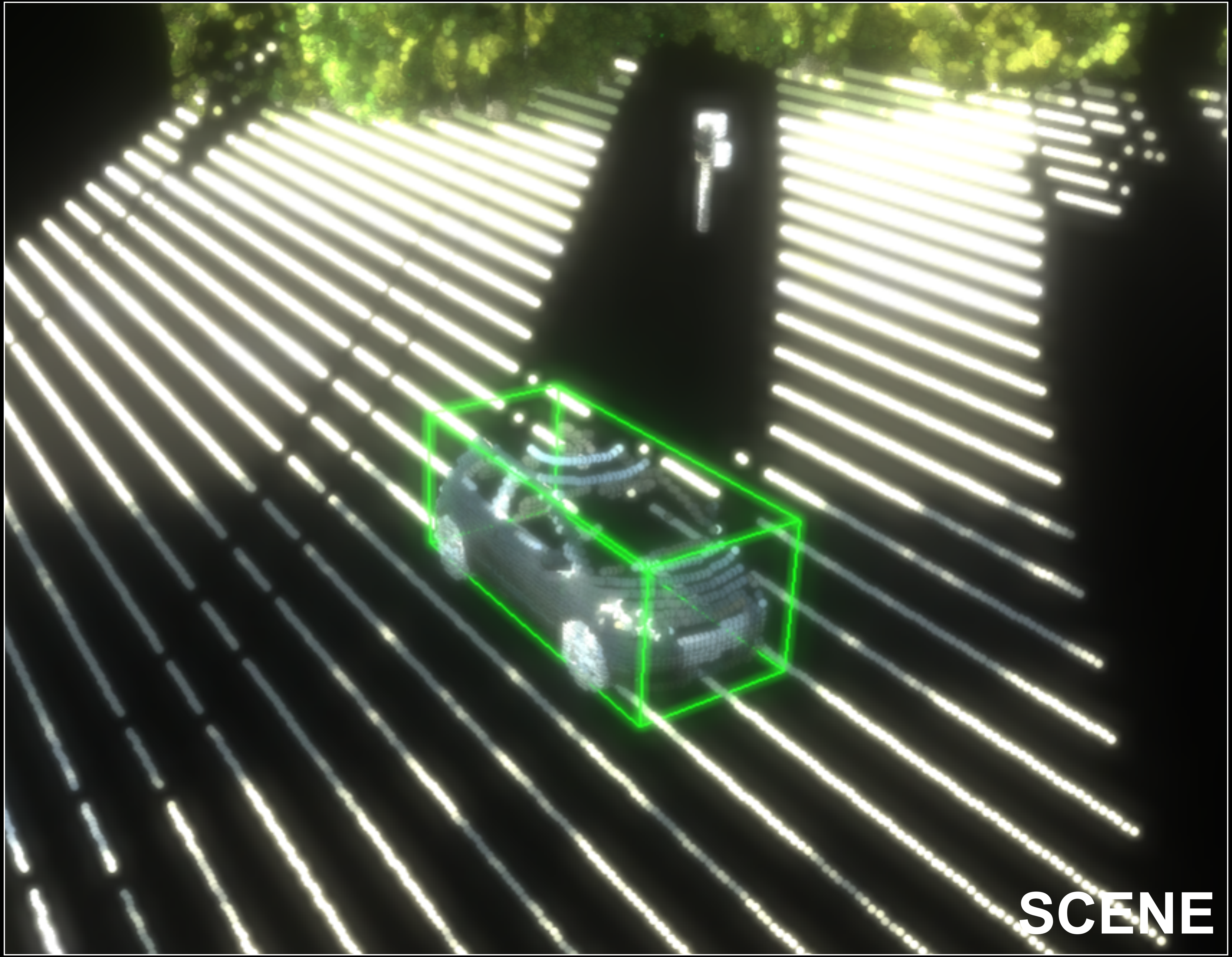
Removing false positives detected by LiDAR

(zoomed in)



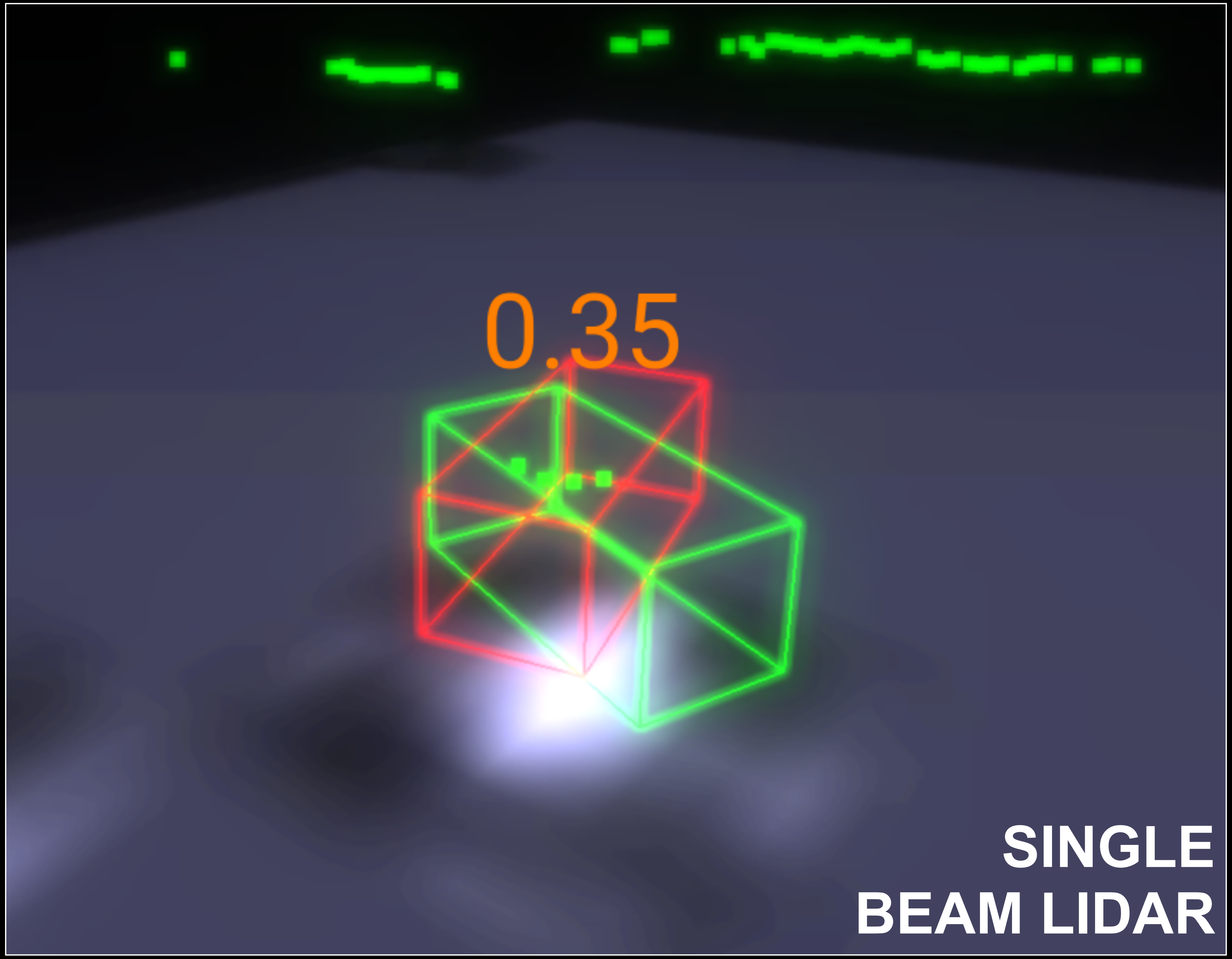
Correcting misaligned detection

(zoomed in)



Correcting misaligned detection

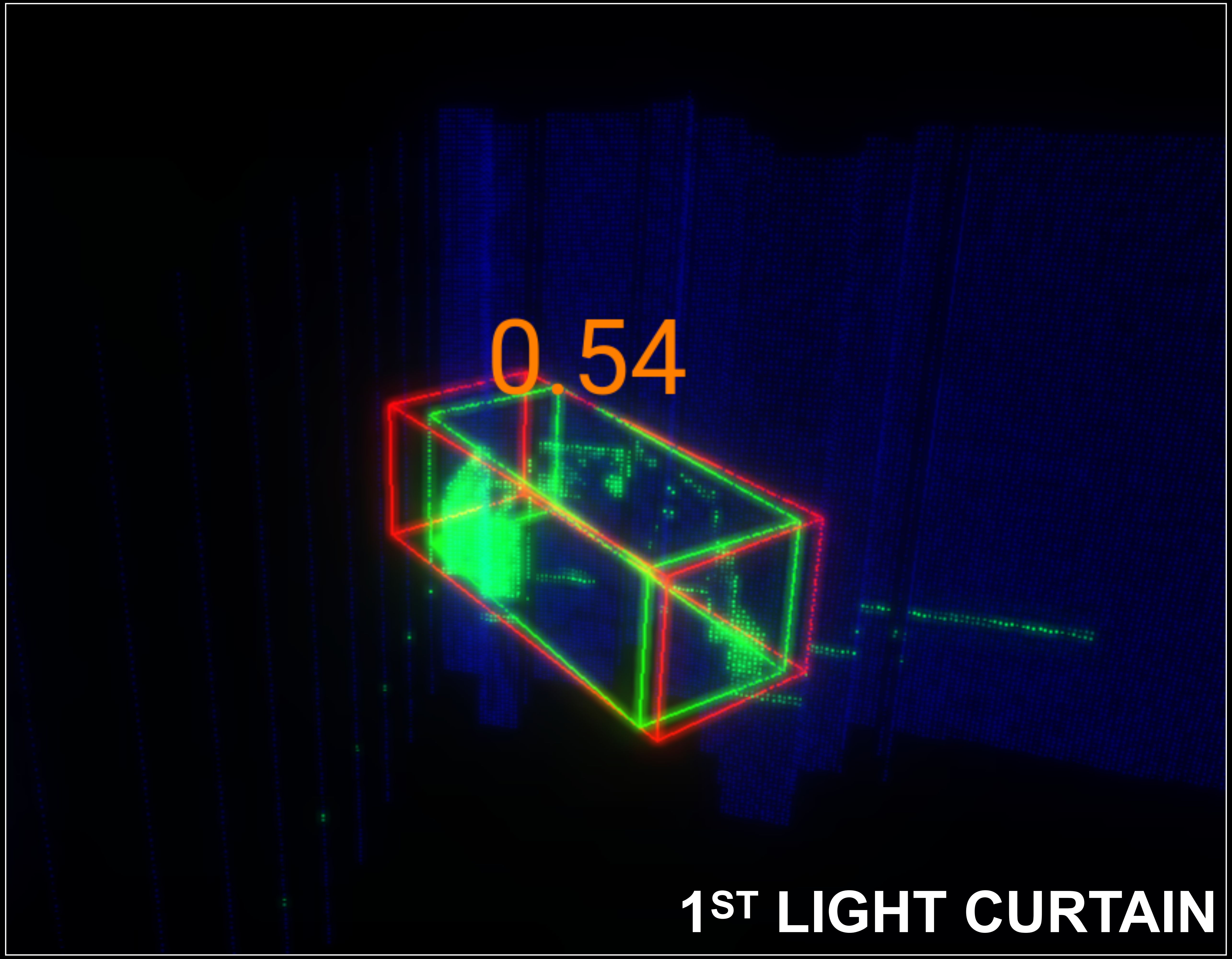
(zoomed in)



*White: more uncertain   Black: less uncertain

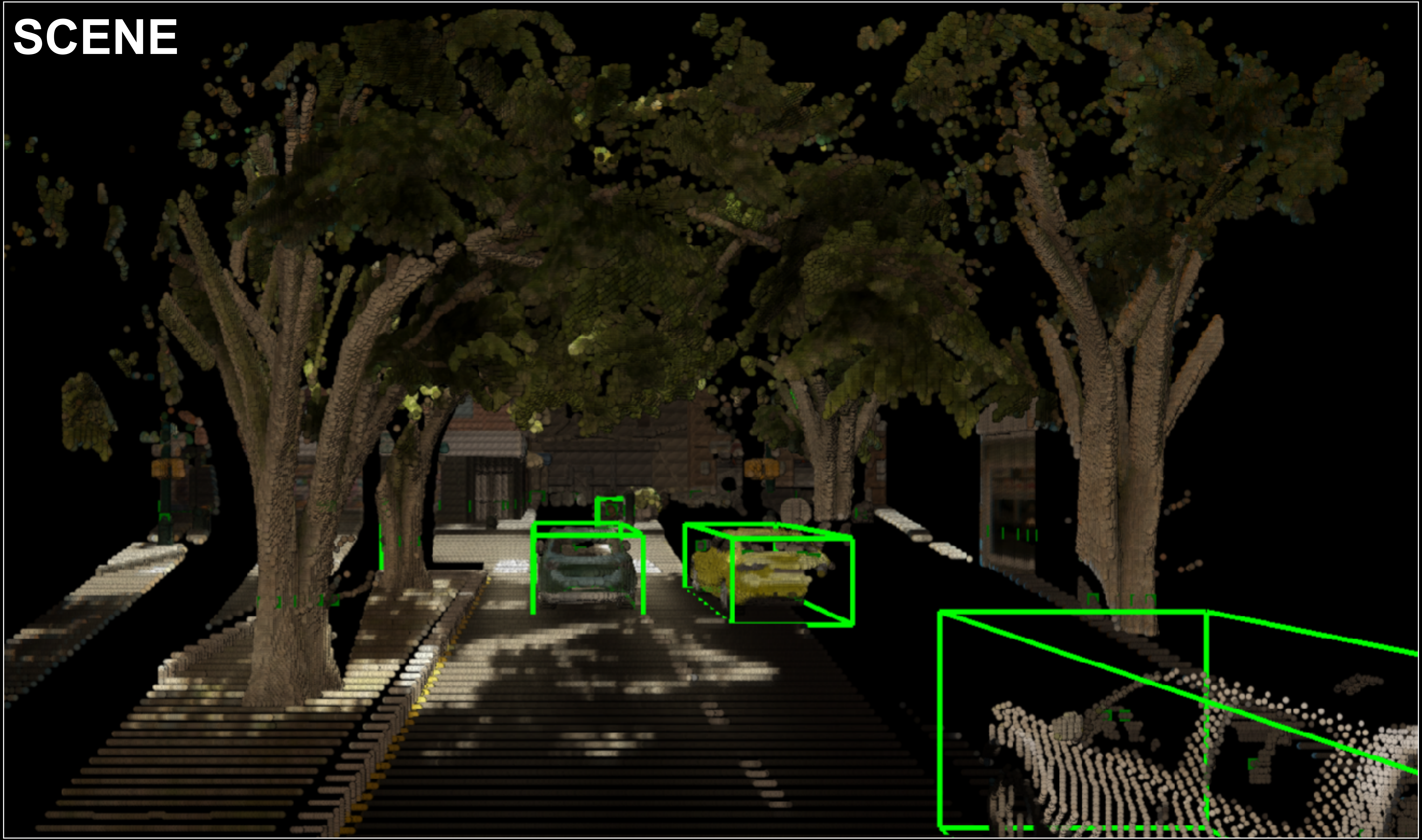
Correcting misaligned detection

(zoomed in)



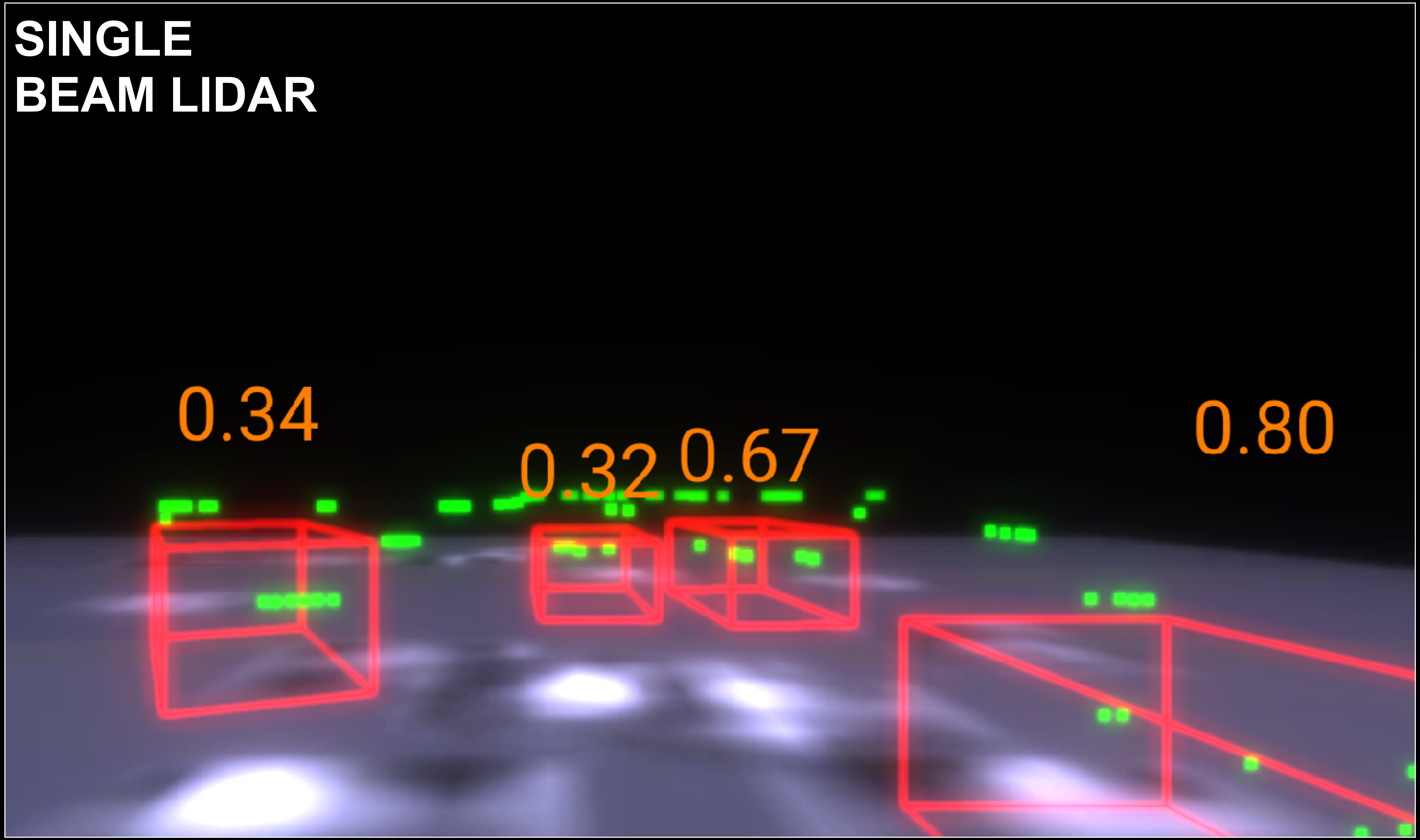
Failure case: many curtains might be required

(zoomed in)



Failure case: many curtains might be required

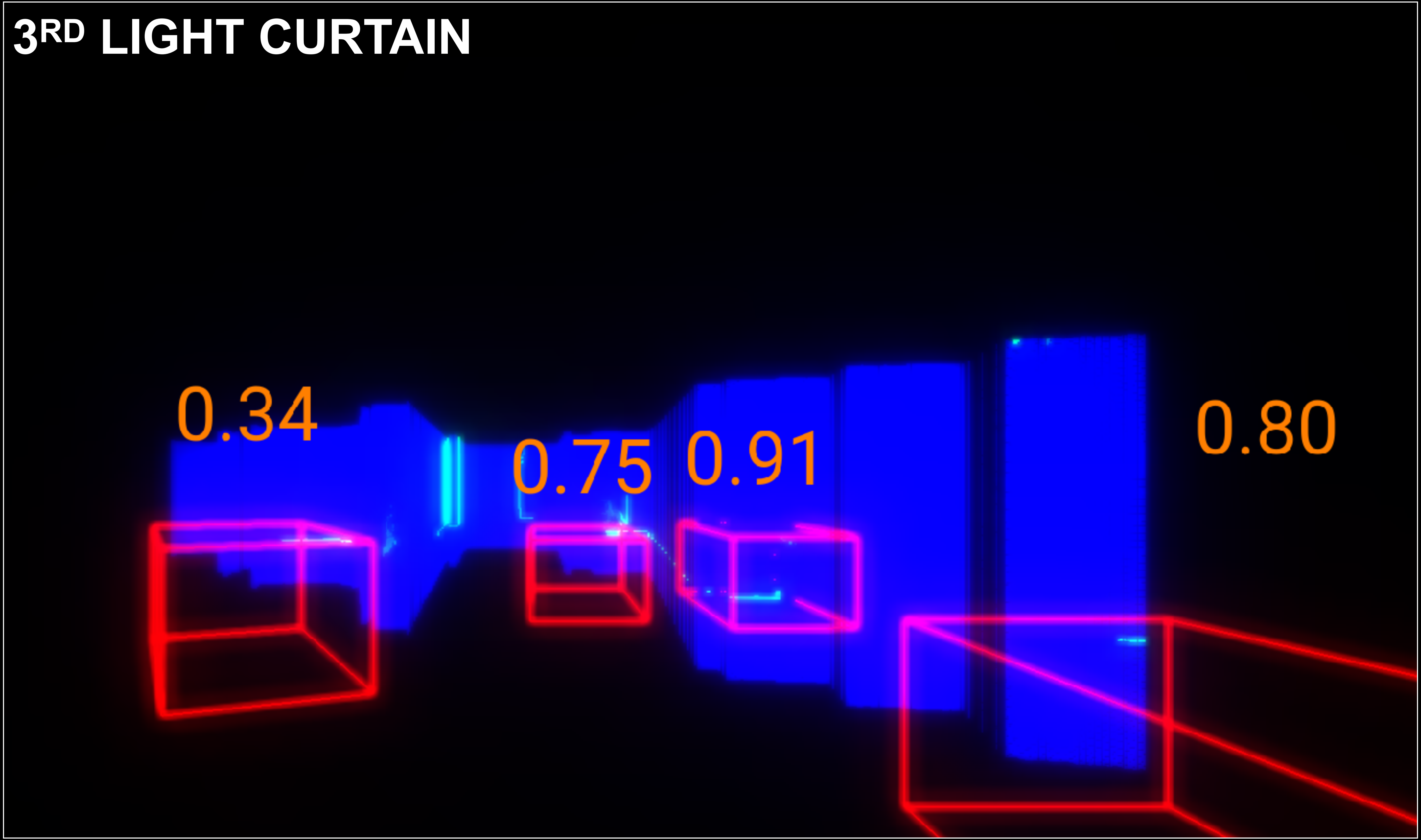
(zoomed in)



*White: more uncertain   Black: less uncertain

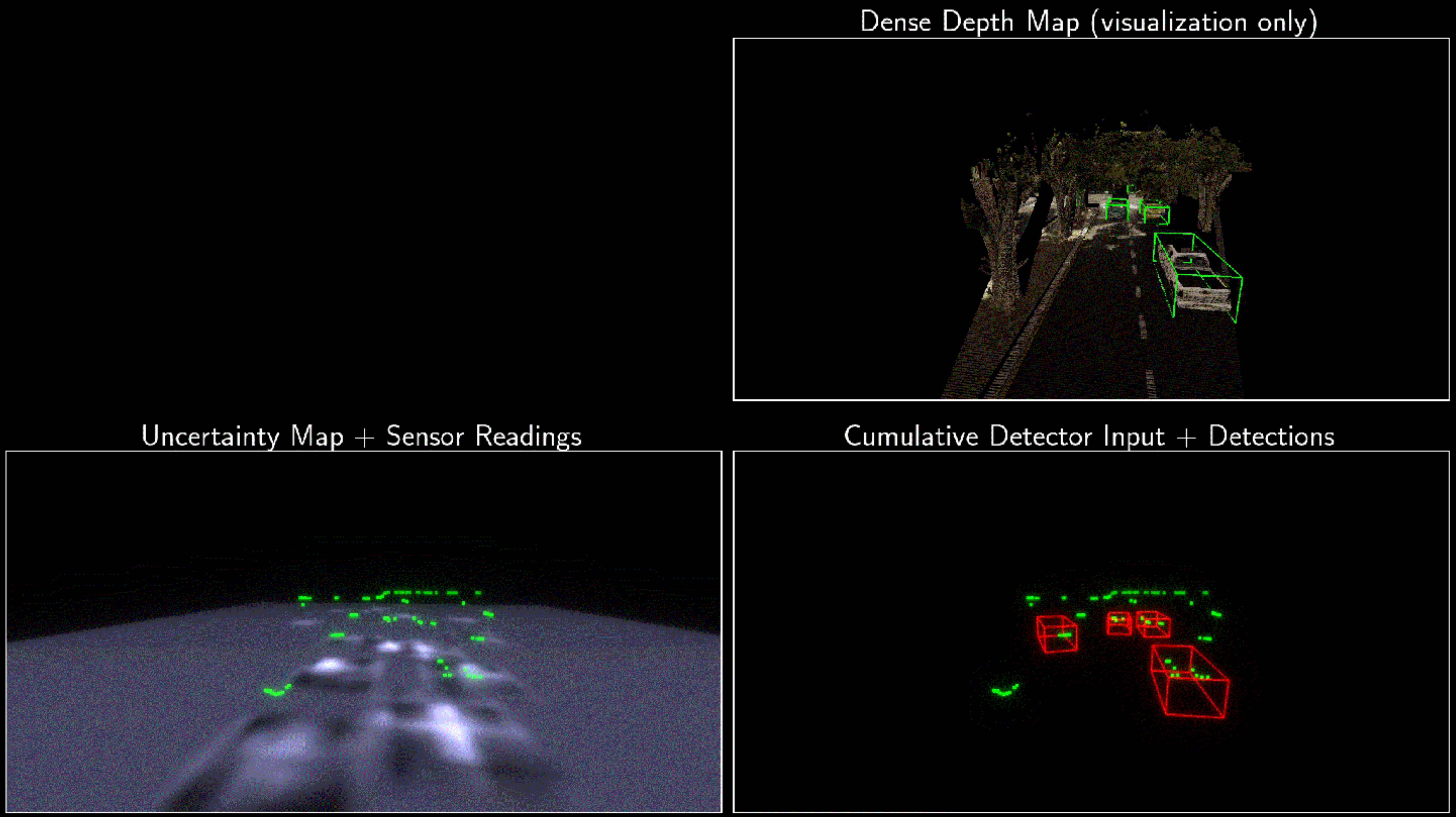
Failure case: many curtains might be required

(zoomed in)



*White: more uncertain   Black: less uncertain

Failure case: many curtains might be required

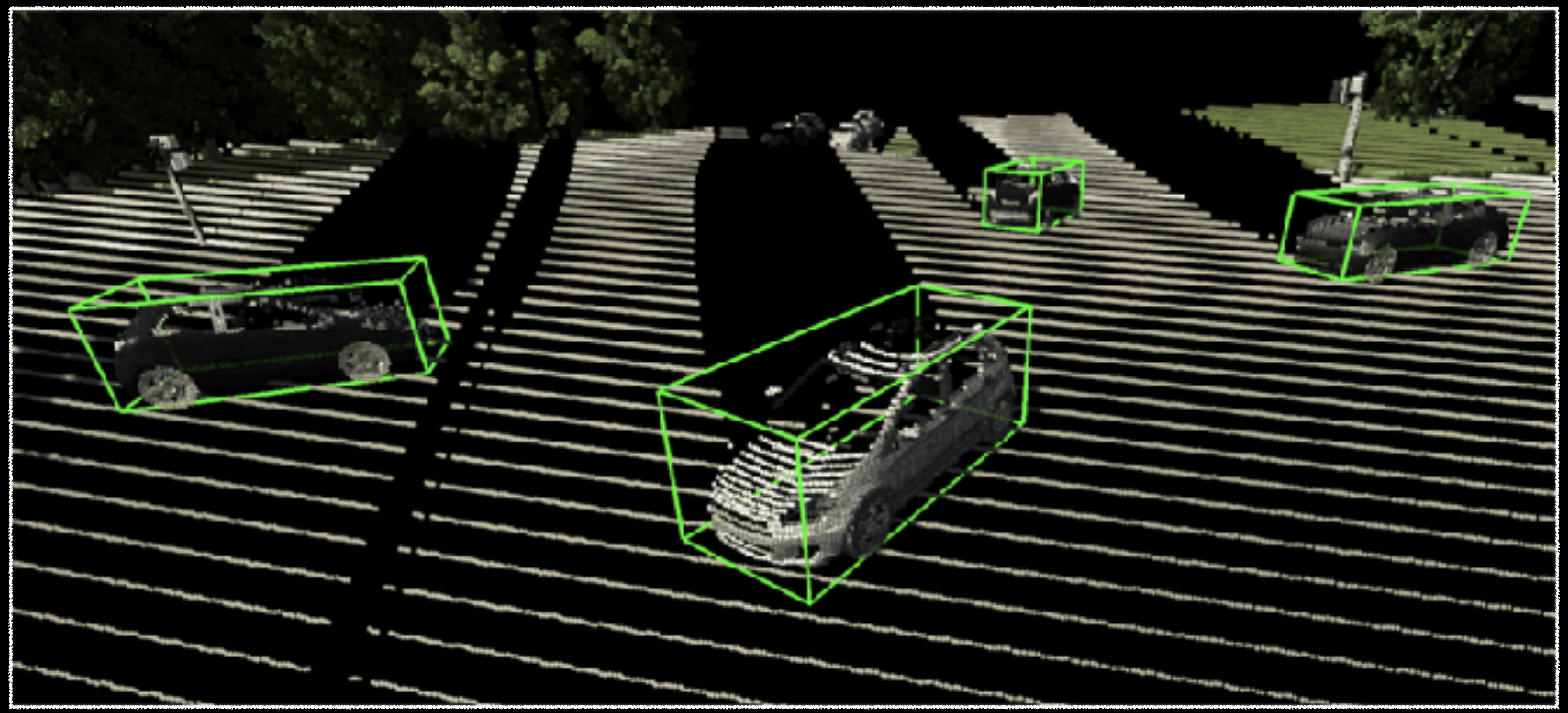


Single Beam LiDAR

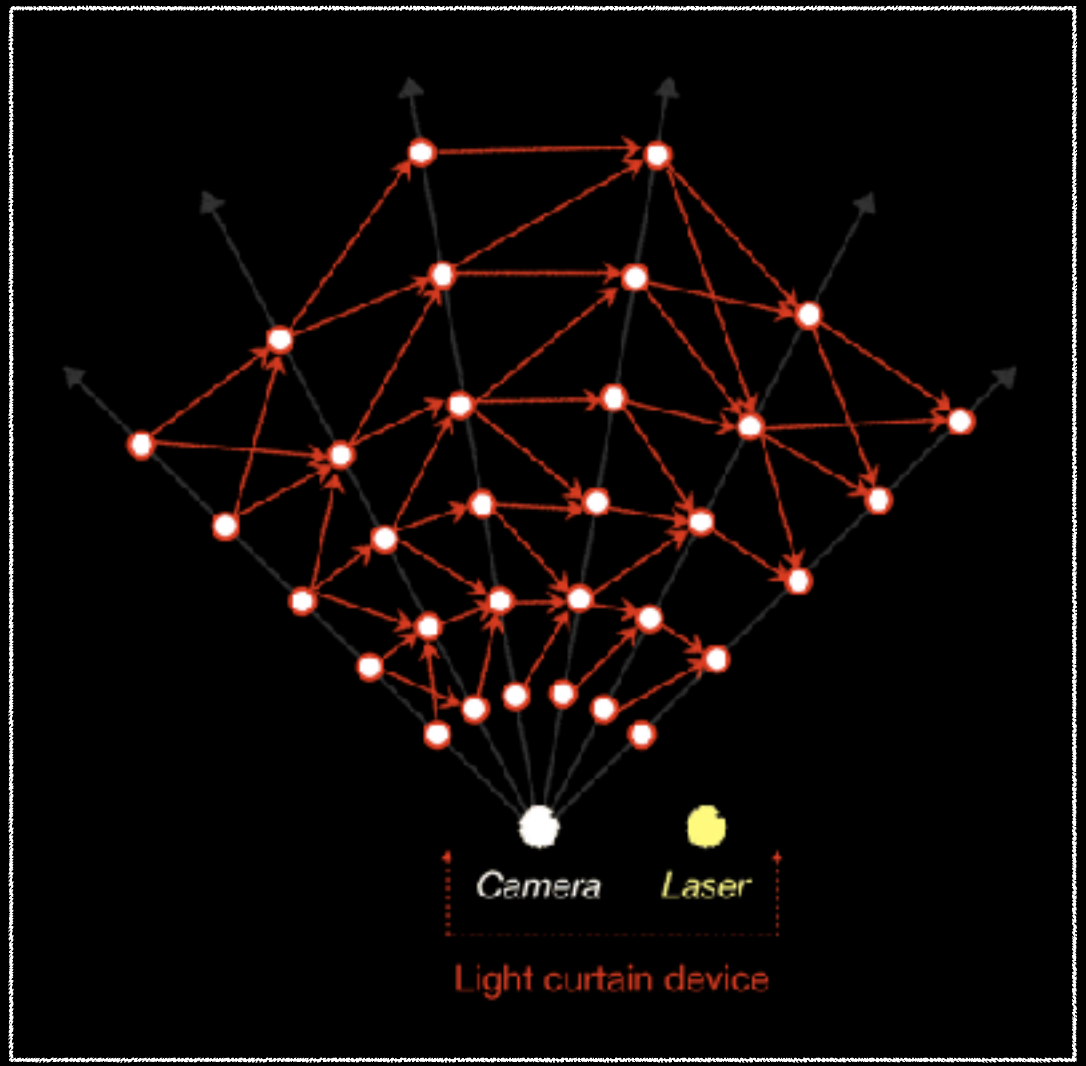
*White: more uncertain   Black: less uncertain

Conclusions

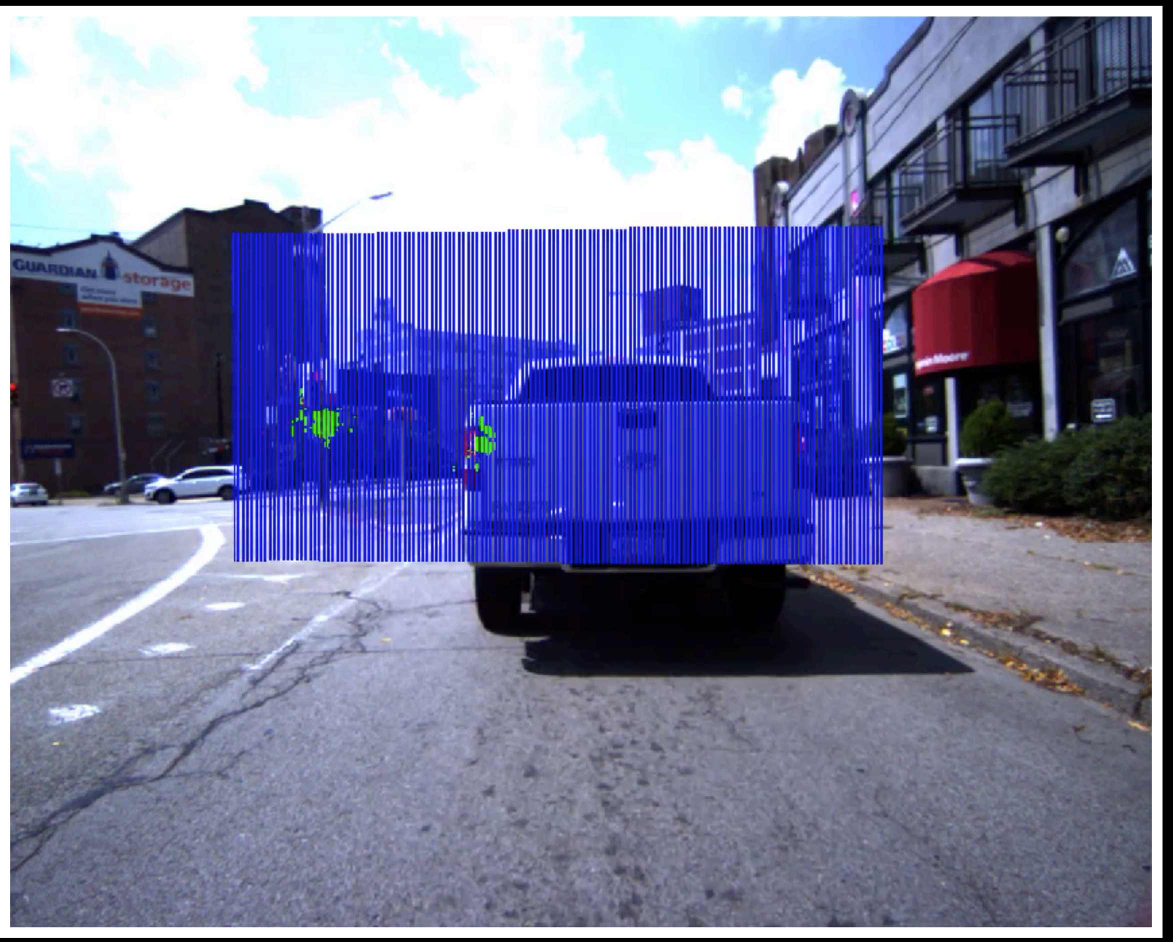
Active Detection



Dynamic Programming



Light Curtain

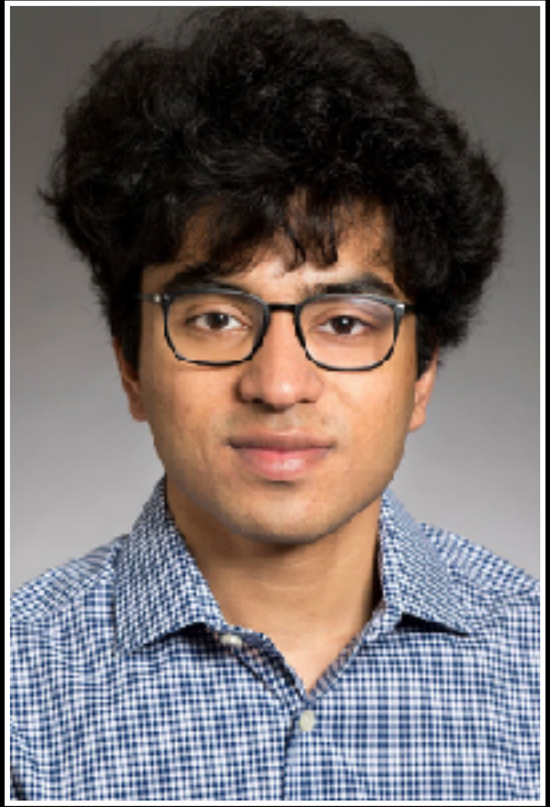


- We propose for a method for active detection using light curtains for autonomous driving.
- We derive an information-gain based objective for light curtain placement.
- We propose a novel optimization algorithm by encoding the light curtain constrains into a constraint graph, and using dynamic programming to maximize the objective.
- We show that our method can successively improve detection accuracy of LiDAR, and is a step towards replacing expensive multi-beam LiDAR systems with inexpensive controllable sensors.

Active Perception using Light Curtains for Autonomous Driving

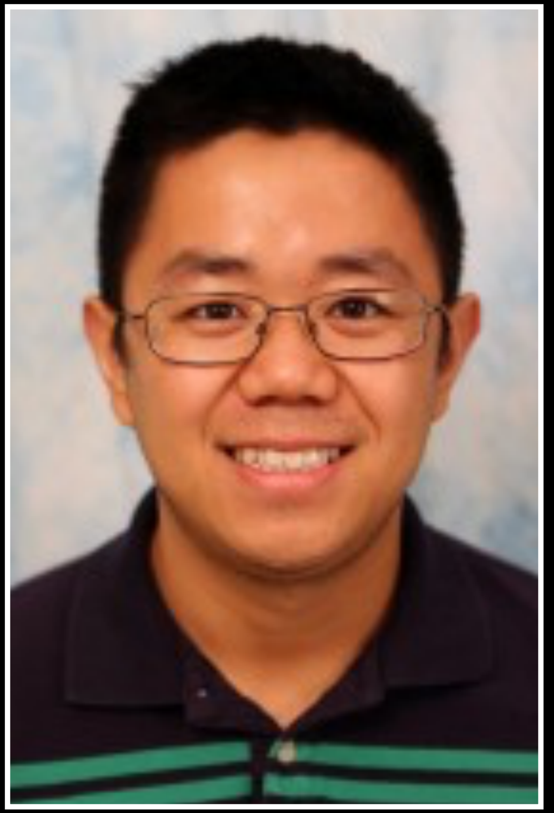
Webpage

http://siddancha.github.io/projects/active-perception-light-curtains



Siddharth Ancha

Yaadhav Raaj



Peiyun Hu

Srinivasa Narasimhan

David Held

Carnegie Mellon University

THE ROBOTICS INSTITUTE

MACHINE LEARNING Carnegie Mellon University