Active Perception using Light Curtains for Autonomous Driving

Siddharth Ancha, Yaadhav Raaj, Peiyun Hu,
Srinivasa G. Narasimhan, and David Held

Carnegie Mellon University, Pittsburgh PA 15213, USA
{sancha,ryaadhav,peiyunh,srinivas,dheld}@andrew.cmu.edu

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

Abstract. Most real-world 3D sensors such as LiDARs perform fixed scans of the entire environment, while being decoupled from the recognition system that processes the sensor data. In this work, we propose a method for 3D object recognition using light curtains, a resource-efficient *controllable* sensor that measures depth at user-specified locations in the environment. Crucially, we propose using prediction uncertainty of a deep learning based 3D point cloud detector to guide active perception. Given a neural network's uncertainty, we develop a novel optimization algorithm to optimally place light curtains to maximize coverage of uncertain regions. Efficient optimization is achieved by encoding the physical constraints of the device into a constraint graph, which is optimized with dynamic programming. We show how a 3D detector can be trained to detect objects in a scene by sequentially placing uncertainty-guided light curtains to successively improve detection accuracy. Links to code can be found on the project webpage.

Keywords: Active Vision, Robotics, Autonomous Driving, 3D Vision

1 Introduction

3D sensors, such as LiDAR, have become ubiquitous for perception in autonomous systems operating in the real world, such as self-driving vehicles and field robots. Combined with recent advances in deep-learning based visual recognition systems, they have lead to significant breakthroughs in perception for autonomous driving, enabling the recent surge of commercial interest in self-driving technology.

However, most 3D sensors in use today perform *passive perception*, i.e. they continuously sense the entire environment while being completely decoupled from the recognition system that will eventually process the sensor data. In such a case, sensing the entire scene can be potentially inefficient. For example, consider an object detector running on a self-driving car that is trying to recognize objects in its environment. Suppose that it is confident that a tree-like structure on the side of the street is not a vehicle, but it is unsure whether an object turning around the curb is a vehicle or a pedestrian. In such a scenario, it might be

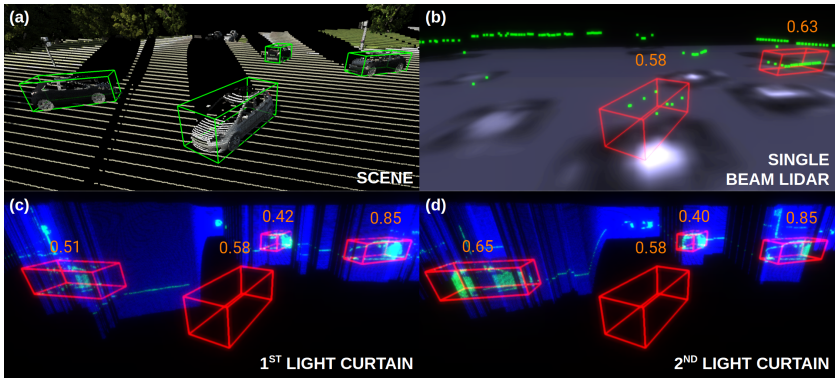


Fig. 1: *Object detection using light curtains.* (a) Scene with 4 cars; ground-truth boxes shown in green. (b) Sparse green points are from a single-beam LiDAR; it can detect only two cars (red boxes). Numbers above detections boxes are confidence scores. Uncertainty map in greyscale is displayed underneath: whiter means higher uncertainty. (c) First light curtain (blue) is placed to optimally cover the most uncertain regions. Dense points (green) from light curtain results in detecting 2 more cars. (d) Second light curtain senses even more points and fixes the misalignment error in the leftmost detection.

beneficial if the 3D sensor focuses on collecting more data from the latter object, rather than distributing its sensing capacity uniformly throughout the scene.

In this work, we propose a method for 3D object detection using *active perception*, i.e. using sensors that can be purposefully controlled to sense specific regions in the environment. Programmable light curtains [22,2] were recently proposed as controllable, light-weight, and resource efficient sensors that measure the presence of objects intersecting any vertical ruled surface whose shape can be specified by the user (see Fig. 2). There are two main advantages of using programmable light curtains over LiDARs. First, they can be cheaply constructed, since light curtains use ordinary CMOS sensors (a current lab-built prototype costs $1000, and the price is expected to go down significantly in production). In contrast, a 64-beam Velodyne LiDAR that is commonly used in 3D self-driving datasets like KITTI [10] costs upwards of $80,000. Second, light curtains generate data with much higher resolution in regions where they actively focus [2] while LiDARs sense the entire environment and have low spatial and angular resolution.

One weakness of light curtains is that they are able to sense only a subset of the environment – a vertical ruled surface (see Fig. 1(c,d), Fig 2). In contrast, a LiDAR senses the entire scene. To mitigate this weakness, we can take advantage of the fact that the light curtain is a *controllable* sensor – we can choose where to place the light curtains. Thus, we must intelligently place light curtains in the appropriate locations, so that they sense the most important parts of the scene. In this work, we develop an algorithm for determining how to best place the light curtains for maximal detection performance.

We propose to use a deep neural network's prediction uncertainty as a guide for determining how to actively sense an environment. Our insight is that if an active sensor images the regions which the network is most uncertain about, the data obtained from those regions can help resolve the network's uncertainty and improve recognition. Conveniently, most deep learning based recognition systems output confidence maps, which can be used for this purpose when converted to an appropriate notion of uncertainty.

Given neural network uncertainty estimates, we show how a light curtain can be placed to *optimally* cover the regions of maximum uncertainty. First, we use an information-gain based framework to propose placing light curtains that maximize the sum of uncertainties of the covered region (Sec. 4.3, Appendix A). However, the structure of the light curtain and physical constraints of the device impose restrictions on how the light curtain can be placed. Our novel solution is to precompute a "constraint graph", which describes all possible light curtain placements that respect these physical constraints. We then use an optimization approach based on dynamic programming to efficiently search over all possible feasible paths in the constraint graph and maximize this objective (Sec. 4.4). This is a novel approach to constrained optimization of a controllable sensor's trajectory which takes advantage of the properties of the problem we are solving.

Our proposed active perception pipeline for 3D detection proceeds as follows. We initially record sparse data with an inexpensive single beam LIDAR sensor that performs fixed 3D scans. This data is input to a 3D point cloud object detector, which outputs an initial set of detections and confidence estimates. These confidence estimates are converted into uncertainty estimates, which are used by our dynamic programming algorithm to determine where to place the first light curtain. The output of the light curtain readings are again input to the 3D object detector to obtain refined detections and an updated uncertainty map. This process of estimating detections and placing new light curtains can be repeated multiple times (Fig. 3). Hence, we are able to sense the environment progressively, intelligently, and efficiently.

We evaluate our algorithm using two synthetic datasets of urban driving scenes [9,29]. Our experiments demonstrate that our algorithm leads to a monotonic improvement in performance with successive light curtain placements. We compare our proposed optimal light curtain placement strategy to multiple baseline strategies and find that they are significantly outperformed by our method. To summarize, our contributions are the following:

- We propose a method for using a deep learning based 3D object detector's prediction uncertainty as a guide for active sensing (Sec. 4.2).
- Given a network's uncertainty, we show how to compute a feasible light curtain that maximizes the coverage of uncertainty. Our novel contribution is to encode the physical constraints of the device into a graph and use dynamic-programming based graph optimization to efficiently maximize the objective while satisfying the physical constraints (Sec. 4.3, 4.4).
- We show how to train such an active detector using online light curtain data generation (Sec. 4.5).

- We empirically demonstrate that our approach leads to significantly improved detection performance compared to a number of baseline approaches (Sec. 5).

2 Related Work

2.1 Active Perception and Next-Best View Planning

Active Perception encompasses a variety of problems and techniques that involve actively controlling the sensor for improved perception [1,23]. Examples include actively modifying camera parameters [1], moving a camera to look around occluding objects [4], and next-best view (NBV) planning [5]. NBV refers to a broad set of problems in which the objective is to select the next best sensing action in order to solve a specific task. Typical problems include object instance classification [24,8,7,18] and 3D reconstruction [12,13,21,6,11]. Many works on next-best view formulate the objective as maximizing information gain (also known as mutual information) [24,7,12,13,21,6], using models such as probabilistic occupancy grids for beliefs over states [24,12,13,21,6]. Our method is similar in spirit to next-best view. One could consider each light curtain placement as obtaining a new "view" of the environment; we try to find the next best light curtain that aids object detection. In Sec. 4.3 and Appendix A, we derive an information-gain based objective to find the next best light curtain placement.

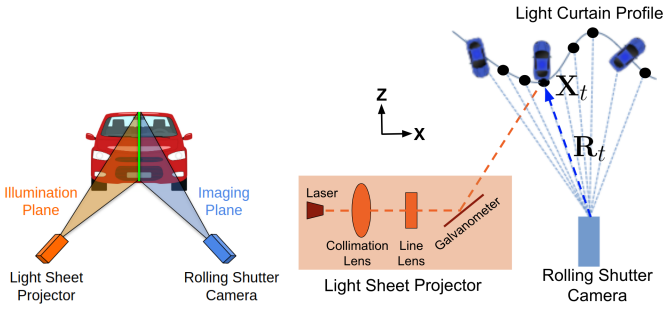
2.2 Object Detection from Point Clouds

There have been many recent advances in deep learning for 3D object detection. Approaches include representing LiDAR data as range images in LaserNet[16], using raw point clouds [19], and using point clouds in the bird's eye view such as AVOD [14], HDNet [26] and Complex-YOLO [20]. Most state-of-the-art approaches use voxelized point clouds, such as VoxelNet [27], PointPillars [15], SECOND [25], and CBGS [28]. These methods process an input point cloud by dividing the space into 3D regions (voxels or pillars) and extracting features from each of region using a PointNet [17] based architecture. Then, the volumetric feature map is converted to 2D features via convolutions, followed by a detection head that produces bounding boxes. We demonstrate that we can use such detectors, along with our novel light curtain placement algorithm, to process data from a single beam LiDAR combined with light curtains.

3 Background on Light Curtains

Programmable *light curtains* [22,2] are a sensor for adaptive depth sensing. "Light curtains" can be thought of as virtual surfaces placed in the environment. They can detect points on objects that intersect this surface. Before explaining how the curtain is created, we briefly describe our coordinate system and the basics of a rolling shutter camera.

Coordinate system: Throughout the paper, we will use the standard camera



(a) Working principle (b) Optical schematic (top view)

Fig. 2: Illustration of programmable light curtains adapted from [2,22]. a) The light curtain is placed at the intersection of the illumination plane (from the projector) and the imaging plane (from the camera). b) A programmable galvanometer and a rolling shutter camera create multiple points of intersection, $\mathbf{X}_t$.

coordinate system centered at the sensor. We assume that the z axis corresponds to depth from the sensor pointing forward, and that the y vector points vertically downwards. Hence the xz-plane is parallel to the ground and corresponds to a top-down view, also referred to as the bird's eye view.

Rolling shutter camera: A rolling shutter camera contains pixels arranged in T number of vertical columns. Each pixel column corresponds to a vertical imaging plane. Readings from only those visible 3D points that lie on the imaging plane get recorded onto its pixel column. We will denote the xz-projection of the imaging plane corresponding to the t-th pixel column by ray $\mathbf{R}_t$, shown in the top-down view in Fig. 2(b). We will refer to these as "camera rays". The camera has a rolling shutter that successively activates each pixel column and its imaging plane one at a time from left to right. The time interval between the activation of two adjacent pixel columns is determined by the pixel clock.

Working principle of light curtains: The latest version of light curtains [2] works by rapidly rotating a light sheet laser in synchrony with the motion of a camera's rolling shutter. A laser beam is collimated and shaped into a line sheet using appropriate lenses and is reflected at a desired angle using a controllable galvanometer mirror (see Fig. 2(b)). The illumination plane created by the laser intersects the active imaging plane of the camera in a vertical line along the curtain profile (Fig. 2(a)). The xz-projection of this vertical line intersecting the t-th imaging plane lies on $\mathbf{R}_t$, and we call this the t-th "control point", denoted by $\mathbf{X}_t$ (Fig. 2(b)).

Light curtain input: The shape of a light curtain is uniquely defined by where it intersects each camera ray in the xz-plane, i.e. the control points $\{\mathbf{X}_1, \ldots, \mathbf{X}_T\}$. These will act as inputs to the light curtain device. In order to produce the light curtain defined by $\{\mathbf{X}_t\}_{t=1}^{T}$, the galvanometer is programmed to compute and rotate at, for each camera ray $\mathbf{R}_t$, the reflection angle $\theta_t(\mathbf{X}_t)$ of the laser beam

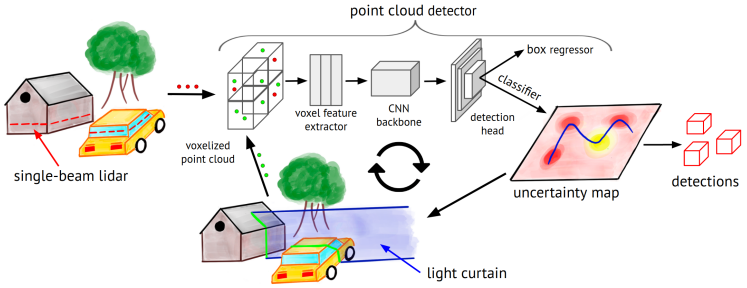


Fig. 3: *Our method for detecting objects using light curtains.* An inexpensive single-beam lidar input is used by a 3D detection network to obtain rough initial estimates of object locations. The uncertainty of the detector is used to optimally place a light curtain that covers the most uncertain regions. The points detected by the light curtain (shown in green in the bottom figure) are input back into the detector so that it can update its predictions as well as uncertainty. The new uncertainty maps can again be used to place successive light curtains in an iterative manner, closing the loop.

such that the laser sheet intersects $\mathbf{R}_t$ at $\mathbf{X}_t$. By selecting a control point on each camera ray, the light curtain device can be made to image any vertical ruled surface [2,22].

Light curtain output: The light curtain outputs a point cloud of all 3D visible points in the scene that intersect the light curtain surface. The density of light curtain points on the surface is usually much higher than LiDAR points.

Light curtain constraints: The rotating galvanometer can only operate at a maximum angular velocity $\omega_{\max}$. Let $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$ be the control points on two consecutive camera rays $\mathbf{R}_t$ and $\mathbf{R}_{t+1}$. These induce laser angles $\theta(\mathbf{X}_t)$ and $\theta(\mathbf{X}_{t+1})$ respectively. If Δt is the time difference between when the t-th and $(t+1)$-th pixel columns are active, the galvanometer needs to rotate by an angle of $\Delta\theta(\mathbf{X}_t) = \theta(\mathbf{X}_{t+1}) - \theta(\mathbf{X}_t)$ within Δt time. Denote $\Delta\theta_{\max} = \omega_{\max} \cdot \Delta t$. Then the light curtain can only image control points subject to $|\theta(\mathbf{X}_{t+1}) - \theta(\mathbf{X}_t)| \leq \Delta\theta_{\max},\ \forall 1 \leq t < T$.

4 Approach

4.1 Overview

Our aim is to use light curtains for detecting objects in a 3D scene. The overall approach is illustrated in Fig. 3. We use a voxel-based point cloud detector [25] and train it to use light curtain data without any architectural changes. The pipeline illustrated in Fig. 3 proceeds as follows.

To obtain an initial set of object detections, we use data from an inexpensive single-beam LiDAR as input to the detector. This produces rough estimates of object locations in the scene. Single-beam LiDAR is inexpensive because it

consists of only one laser beam as opposed to 64 or 128 beams that are common in autonomous driving. The downside is that the data from the single beam contains very few points; this results in inaccurate detections and high uncertainty about object locations in the scene (see Fig. 1b).

Alongside bounding box detections, we can also extract from the detector an "uncertainty map" (explained in Sec. 4.2). We then use light curtains, placed in regions guided by the detector's uncertainty, to collect more data and iteratively refine the object detections. In order to get more data from the regions the detector is most uncertain about, we derive an information-gain based objective function that sums the uncertainties along the light curtain control points (Sec. 4.3 and Appendix A), and we develop a constrained optimization algorithm that places the light curtain to maximize this objective (Sec. 4.4).

Once the light curtain is placed, it returns a dense set of points where the curtain intersects with visible objects in the scene. We maintain a *unified point cloud*, which we define as the union of all points observed so far. The unified point cloud is initialized with the points from the single-beam LiDAR. Points from the light curtain are added to the unified point cloud and this data is input back into the detector. Note that the input representation for the detector remains the same (point clouds), enabling the use of existing state-of-the-art point cloud detection methods without any architectural modifications.

As new data from the light curtains are added to the unified point cloud and input to the detector, the detector refines its predictions and improves its accuracy. Furthermore, the additional inputs cause the network to update its uncertainty map; the network may no longer be uncertain about the areas that were sensed by the light curtain. Our algorithm uses the new uncertainty map to generate a new light curtain placement. We can iteratively place light curtains to cover the current uncertain regions and input the sensed points back into the network, closing the loop and iteratively improving detection performance.

4.2 Extracting uncertainty from the detector

The standard pipeline for 3D object detection [27,25,15] proceeds as follows. First, the ground plane (parallel to the xz-plane) is uniformly tiled with "anchor boxes"; these are reference boxes used by a 3D detector to produce detections. They are located on points in a uniformly discretized grid $G = [x_{\min}, x_{\max}] \times [z_{\min}, z_{\max}]$. For example, a $[-40\text{m}, 40\text{m}] \times [0\text{m}, 70.4\text{m}]$ grid is used for detecting cars in KITTI [10]. A 3D detector, which is usually a binary detector, takes a point cloud as input, and produces a binary classification score $p \in [0, 1]$ and bounding box regression offsets for every anchor box. The score p is the estimated probability that the anchor box contains an object of a specific class (such as car/pedestrian). The detector produces a detection for that anchor box if p exceeds a certain threshold. If so, the detector combines the fixed dimensions of the anchor box with its predicted regression offsets to output a detection box.

We can convert the confidence score to binary entropy $H(p) \in [0, 1]$ where $H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$. Entropy is a measure of the detector's uncertainty about the presence of an object at the anchor location. Since we

have an uncertainty score at uniformly spaced anchor locations parallel to the xz-plane, they form an "uncertainty map" in the top-down view. We use this uncertainty map to place light curtains.

4.3 Information gain objective

Based on the uncertainty estimates given by Sec. 4.2, our method determines how to place the light curtain to sense the most uncertain/ambiguous regions. It seems intuitive that sensing the locations of highest detector uncertainty can provide the largest amount of information from a single light curtain placement, towards improving detector accuracy. As discussed in Sec. 3, a single light curtain placement is defined by a set of T control points $\{\mathbf{X}_t\}_{t=1}^T$. The light curtain will be placed to lie vertically on top of these control points. To define an optimization objective, we use the framework of information gain (commonly used in next-best view methods; see Sec. 2.1) along with some simplifying assumptions (see Appendix A). We show that under these assumptions, placing a light curtain to maximize information gain (a mathematically defined information-theoretic quantity) is equivalent to maximizing the objective $J(\mathbf{X}_1, \ldots, \mathbf{X}_T) = \sum_{t=1}^T H(\mathbf{X}_t)$, where $H(\mathbf{X})$ is the binary entropy of the detector's confidence at the anchor location of $\mathbf{X}$. When the control point $\mathbf{X}$ does not exactly correspond to an anchor location, we impute $H(\mathbf{X})$ by nearest-neighbor interpolation from the uncertainty map. Please see Appendix A for a detailed derivation.

4.4 Optimal light curtain placement

In this section, we will describe an exact optimization algorithm to maximize the objective function $J(\mathbf{X}_1, \ldots, \mathbf{X}_T) = \sum_{t=1}^T H(\mathbf{X}_t)$.

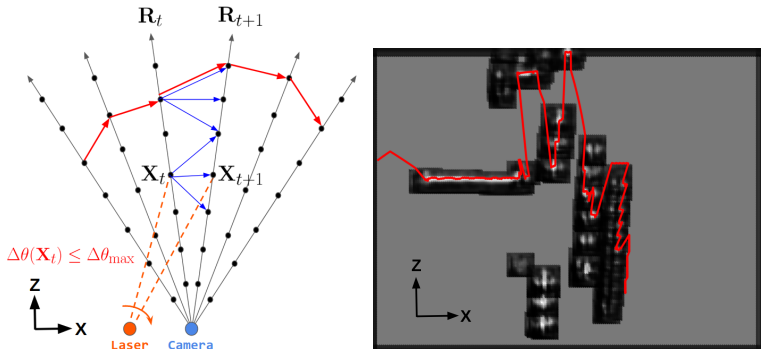
Constrained optimization: The control points $\{\mathbf{X}_t\}_{t=1}^T$, where each $\mathbf{X}_t$ lies on the the camera ray $\mathbf{R}_t$, must be chosen to satisfy the physical constraints of the light curtain device: $|\theta(\mathbf{X}_{t+1}) - \theta(\mathbf{X}_t)| \leq \Delta\theta_{\max}$ (see Sec. 3: light curtain constraints). Hence, this is a constrained optimization problem. We discretize the problem by considering a dense set of N discrete, equally spaced points $\mathcal{D}_t = \{\mathbf{X}_t^{(n)}\}_{n=1}^N$ on each ray $\mathbf{R}_t$. We will assume that $\mathbf{X}_t \in \mathcal{D}_t$ for all $1 \leq t \leq T$ henceforth unless stated otherwise. We use $N = 80$ in all our experiments which we found to be sufficiently large. Overall, the optimization problem can be formulated as:

$$\arg \max_{\{\mathbf{X}_t\}_{t=1}^T} \sum_{t=1}^T H(\mathbf{X}_t) \tag{1}$$

$$\text{where } \mathbf{X}_t \in \mathcal{D}_t \ \forall 1 \leq t \leq T \tag{2}$$

$$\text{subject to } |\theta(\mathbf{X}_{t+1}) - \theta(\mathbf{X}_t)| \leq \Delta\theta_{\max}, \ \forall 1 \leq t < T \tag{3}$$

Light Curtain Constraint Graph: we encode the light curtain constraints into a graph, as illustrated in Figure 4. Each black ray corresponds to a camera ray. Each black dot on the ray is a vertex in the constraint graph. It represents a



(a) (b)

Fig. 4: (a) Light curtain constraint graph. Black dots are nodes and blue arrows are the edges of the graph. The optimized light curtain profile is depicted as red arrows. (b) Example uncertainty map from the detector, and optimized light curtain profile in red. Black is lowest uncertainty and white is highest uncertainty. The optimized light curtain covers the most uncertain regions.

candidate control point and is associated with an uncertainty score. Exactly one control point must be chosen per camera ray. The optimization objective is to choose such points to maximize the total sum of uncertainties. An edge between two control points indicates that the light curtain is able to transition from one control point $\mathbf{X}_t$ to the next, $\mathbf{X}_{t+1}$ without violating the maximum velocity light curtain constraints. Thus, the maximum velocity constraint (Eqn. 3) can be specified by restricting the set of edges (depicted using blue arrows). We note that the graph only needs to be constructed once and can be done offline.

Dynamic programming for constrained optimization: The number of possible light curtain placements, $|\mathcal{D}_1 \times \cdots \times \mathcal{D}_T| = N^T$, is exponentially large, which prevents us from searching for the optimal solution by brute force. However, we observe that the problem can be decomposed into simpler subproblems. In particular, let us define $J_t^*(\mathbf{X}_t)$ as the optimal sum of uncertainties of the *tail subproblem* starting from $\mathbf{X}_t$ i.e.

$$J_t^*(\mathbf{X}_t) = \max_{\mathbf{X}_{t+1},\ldots,\mathbf{X}_T} H(\mathbf{X}_t) + \sum_{k=t+1}^{T} H(\mathbf{X}_k); \tag{4}$$

$$\text{subject to } |\theta(\mathbf{X}_{k+1}) - \theta(\mathbf{X}_k)| \leq \Delta\theta_{\max},\ \forall\ t \leq k < T \tag{5}$$

If we were able to compute $J_t^*(\mathbf{X}_t)$, then this would help in solving a more complex subproblem using recursion: we observe that $J_t^*(\mathbf{X}_t)$ has the property of *optimal substructure*, i.e. the optimal solution of $J_{t-1}^*(\mathbf{X}_{t-1})$ can be computed from the optimal solution of $J_t^*(\mathbf{X}_t)$ via

$$\begin{aligned} J_{t-1}^*(\mathbf{X}_{t-1}) = H(\mathbf{X}_{t-1}) + \max_{\mathbf{X}_t \in \mathcal{D}_t} & J_t^*(\mathbf{X}_t) \\ \text{subject to } & |\theta(\mathbf{X}_t) - \theta(\mathbf{X}_{t-1})| \leq \Delta\theta_{\max} \end{aligned} \tag{6}$$

Because of this optimal substructure property, we can solve for $J^*_{t-1}(\mathbf{X}_{t-1})$ via dynamic programming. We also note that the solution to $\max_{\mathbf{X}_1} J^*_1(\mathbf{X}_1)$ is the solution to our original constrained optimization problem (Eqn. 1-3).

We thus perform the dynamic programming optimization as follows: the recursion from Eqn. 6 can be implemented by first performing a backwards pass, starting from T and computing $J^*_t(\mathbf{X}_t)$ for each $\mathbf{X}_t$. Computing each $J^*_t(\mathbf{X}_t)$ takes only $O(B_{\text{avg}})$ time where B_{avg} is the average degree of a vertex (number of edges starting from a vertex) in the constraint graph, since we iterate once over all edges of $\mathbf{X}_t$ in Eqn. 6. Then, we do a forward pass, starting with $\arg\max_{\mathbf{X}_1 \in \mathcal{D}_1} J^*_1(\mathbf{X}_1)$ and for a given $\mathbf{X}^*_{t-1}$, choosing $\mathbf{X}^*_t$ according to Eqn. 6. Since there are N vertices per ray and T rays in the graph, the overall algorithm takes $O(NTB_{\text{avg}})$ time; this is a significant reduction from the $O(N^T)$ brute-force solution. We describe a simple extension of this objective that encourages smoothness in Appendix B .

4.5 Training active detector with online training data generation

The same detector is used to process data from the single beam LiDAR and all light curtain placements. Since the light curtains are placed based on the output (uncertainty maps) of the detector, the input point cloud for the next iteration depends on the current weights of the detector. As the weights change during training, so does the input data distribution. We account for non-stationarity of the training data by generating it online during the training process. This prevents the input distribution from diverging from the network weights during training. See Appendix C for algorithmic details and ablation experiments.

5 Experiments

To evaluate our algorithm, we need dense ground truth depth maps to simulate an arbitrary placement of a light curtain. However, standard autonomous driving datasets, such as KITTI [10] and nuScenes [3], contain only sparse LiDAR data, and hence the data is not suitable to accurately simulate a dense light curtain to evaluate our method. To circumvent this problem, we demonstrate our method on two synthetic datasets that provide dense ground truth depth maps, namely the Virtual KITTI [9] and SYNTHIA [29] datasets. Please find more details of the datasets and the evaluation metrics in Appendix D.

Our experiments demonstrate the following: First, we show that our method for successive placement of light curtains improves detection performance; particularly, there is a significant increase between the performance of single-beam LiDAR and the performance after placing the first light curtain. We also compare our method to multiple ablations and alternative placement strategies that demonstrate that each component of our method is crucial to achieve good performance. Finally, we show that our method can generalize to many more light curtain placements at test time than the method was trained on. In the appendix, we perform further experiments that include evaluating the generalization of our method to noise in the light curtain data, an ablation experiment for training with online data generation (Sec. 4.5), and efficiency analysis.

5.1 Comparison with varying number of light curtains

We train our method using online training data generation simultaneously on data from single-beam LiDAR and one, two, and three light curtain placements. We perform this experiment for both the Virtual KITTI and SYNTHIA datasets. The accuracies on their tests sets are reported in Table 1.

	Virtual KITTI				**SYNTHIA**			
	3D mAP		BEV mAP		3D mAP		BEV mAP	
	0.5 IoU	0.7 IoU	0.5 IoU	0.7 IoU	0.5 IoU	0.7 IoU	0.5 IoU	0.7 IoU
Single Beam Lidar	39.91	15.49	40.77	36.54	60.49	47.73	60.69	51.22
Single Beam Lidar (separate model)	42.35	23.66	47.77	40.15	60.69	48.23	60.84	57.98
1 Light Curtain	58.01	35.29	58.51	47.05	68.79	55.99	68.97	59.63
2 Light Curtains	60.86	37.91	61.10	49.84	69.02	57.08	69.17	67.14
3 Light Curtains	**68.52**	**38.47**	**68.82**	**50.53**	**69.16**	**57.30**	**69.25**	**67.25**

Table 1: Performance of the detector trained with single-beam LiDAR and up to three light curtains. Performance improves with more light curtain placements, with a significant jump at the first light curtain placement.

Note that there is a significant and consistent increase in the accuracy between single-beam LiDAR performance and the first light curtain placement (row 1 and row 3). This shows that actively placing light curtains on the most uncertain regions can improve performance over a single-beam LiDAR that performs fixed scans. Furthermore, placing more light curtains consistently improves detection accuracy.

As an ablation experiment, we train a separate model only on single-beam LiDAR data (row 2), for the same number of training iterations. This is different from row 1 which was trained with both single beam LiDAR and light curtain data but evaluated using only data for a single beam LiDAR. Although training a model with only single-beam LiDAR data (row 2) improves performance over row 1, it is still significantly outperformed by our method which uses data from light curtain placements.

Noise simulations: In order to simulate noise in the real-world sensor, we perform experiments with added noise in the light curtain input. We demonstrate that the results are comparable to the noiseless case, indicating that our method is robust to noise and is likely to transfer well to the real world. Please see Appendix E for more details.

5.2 Comparison with alternative light curtain placement strategies

In our approach, light curtains are placed by maximizing the coverage of uncertain regions using a dynamic programming optimization. How does this compare to other strategies for light curtain placement? We experiment with several baselines:

1. *Random*: we place frontoparallel light curtains at a random z-distance from the sensor, ignoring the detector's uncertainty map.
2. *Fixed depth*: we place a frontoparallel light curtain at a fixed z-distance (15m, 30m, 45m) from the sensor, ignoring the detector's uncertainty map.
3. *Greedy optimization*: this baseline tries to evaluate the benefits of using a dynamic programming optimization. Here, we use the same light curtain constraints described in Section 4.4 (Figure 4(a)). We greedily select the next control point based on local uncertainty instead of optimizing for the future sum of uncertainties. Ties are broken by (a) choosing smaller laser angle changes, and (b) randomly.
4. *Frontoparallel + Uncertainty*: Our optimization process finds light curtains with flexible shapes. What if the shapes were constrained to make the optimization problem easier? If we restrict ourselves to frontoparallel curtains, we can place them at the z-distance of maximum uncertainty by simply summing the uncertainties for every fixed value of z.

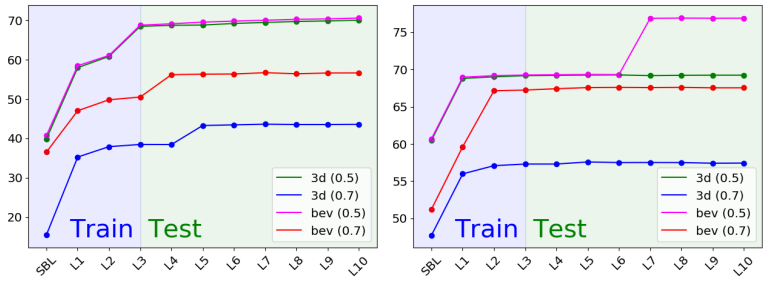
The results on the Virtual KITTI and SYNTHIA datasets are shown in Table 2. Our method significantly and consistently outperforms all baselines. This empirically demonstrates the value of using dynamic programming for light curtain placement to improve object detection performance.

5.3 Generalization to successive light curtain placements

If we train a detector using our online light curtain data generation approach for k light curtains, can the performance generalize to more than k light curtains? Specifically, if we continue to place light curtains beyond the number trained for,

	Virtual KITTI				**SYNTHIA**			
	3D mAP		BEV mAP		3D mAP		BEV mAP	
	.5 IoU	.7 IoU	.5 IoU	.7 IoU	.5 IoU	.7 IoU	.5 IoU	.7 IoU
Random	41.29	17.49	46.65	38.09	60.43	47.09	60.66	58.14
Fixed depth - 15m	44.99	22.20	46.07	38.05	60.74	48.16	60.89	58.48
Fixed depth - 30m	39.72	19.05	45.21	35.83	60.02	47.88	60.23	57.89
Fixed depth - 45m	39.86	20.02	40.61	36.87	60.23	48.12	60.43	57.77
Greedy Optimization (Randomly break ties)	37.40	19.93	42.80	35.33	60.62	47.46	60.83	58.22
Greedy Optimization (Min laser angle change)	39.20	20.19	44.80	36.94	60.61	47.05	60.76	58.07
Frontoparallel + Uncertainty	39.41	21.25	45.10	37.80	60.36	47.20	60.52	58.00
Ours	**58.01**	**35.29**	**58.51**	**47.05**	**68.79**	**55.99**	**68.97**	**59.63**

Table 2: Baselines for alternate light curtain placement strategies, trained and tested on (a) Virtual KITTI and (b) SYNTHIA datasets. Our dynamic programming optimization approach significantly outperforms all other strategies.



(a) Generalization in Virtual KITTI (b) Generalization in SYNTHIA

Fig. 5: *Generalization to many more light curtains than what the detector was trained for*. We train using online data generation on single-beam lidar and only 3 light curtains. We then test with placing 10 curtains, on (a) Virtual KITTI, and (b) SYNTHIA. Performance continues to increase monotonically according to multiple metrics. Takeaway: one can safely place more light curtains at test time and expect to see sustained improvement in accuracy.

will the accuracy continue improving? We test this hypothesis by evaluating on 10 light curtains, many more than the model was trained for (3 light curtains). Figure 5 shows the performance as a function of the number of light curtains. We find that in both Virtual KITTI and SYNTHIA, the accuracy monotonically improves with the number of curtains.

This result implies that a priori one need not worry about how many light curtains will be placed at test time. If we train on only 3 light curtains, we can place many more light curtains at test time; our results indicate that the performance will keep improving.

5.4 Qualitative analysis

We visualized a successful case of our method in Fig. 1. This is an example where our method detects false negatives missed by the single-beam LiDAR. We also show two other types of successful cases where light curtains remove false positive detections and fix misalignment errors in Figure 6. In Figure 7, we show the predominant failure case of our method. See captions for more details.

6 Conclusions

In this work, we develop a method to use light curtains, an actively controllable resource-efficient sensor, for object recognition in static scenes. We propose to use a 3D object detector's prediction uncertainty as a guide for deciding where to sense. By encoding the constraints of the light curtain into a graph, we show how to optimally and feasibly place a light curtain that maximizes the coverage of uncertain regions. We are able to train an active detector that interacts with light

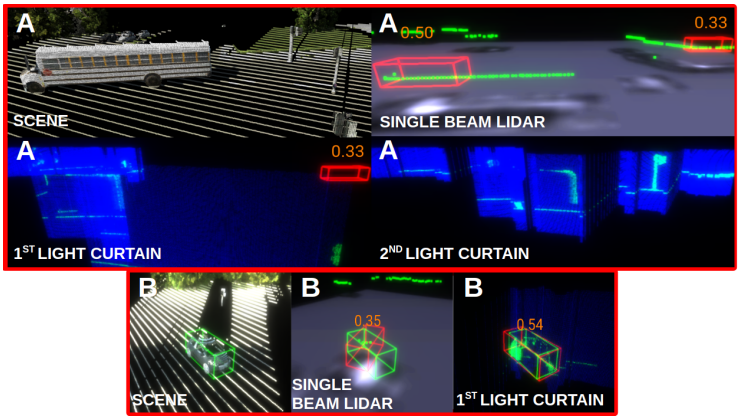


Fig. 6: *Successful cases:* Other type of successful cases than Fig. 1. In (A), the single-beam LiDAR incorrectly detects a bus and a piece of lawn as false positives. They get eliminated successively after placing the first and second light curtains. In (B), the first light curtain fixes misalignment in the bounding box predicted by the single beam LiDAR.

curtains to iteratively and efficiently sense parts of scene in an uncertainty-guided manner, successively improving detection accuracy. We hope this works pushes towards designing perception algorithms that integrate sensing and recognition, towards intelligent and adaptive perception.

Acknowledgements

We thank Matthew O'Toole for feedback on the initial draft of this paper. This material is based upon work supported by the National Science Foundation under Grants No. IIS-1849154, IIS-1900821 and by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092.

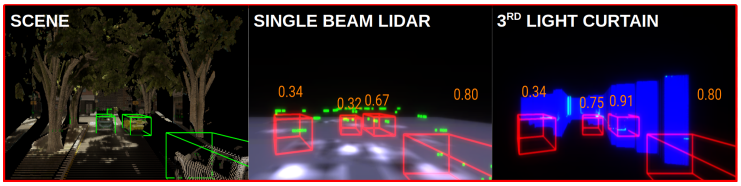


Fig. 7: *Failure cases:* The predominant failure mode is that the single beam LiDAR detects a false positive which is not removed by light curtains because the detector is overly confident in its prediction (so the estimated uncertainty is low). *Middle*: Falsely detecting a tree to be a car. *Right*: After three light curtains, the detection persists because light curtains do not get placed on this false positive. False positive gets removed eventually only after six light curtain placements.

References

1. Bajcsy, R.: Active perception. Proceedings of the IEEE **76**(8), 966–1005 (1988)
2. Bartels, J.R., Wang, J., Whittaker, W.R., Narasimhan, S.G.: Agile depth sensing using triangulation light curtains. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)
4. Cheng, R., Agarwal, A., Fragkiadaki, K.: Reinforcement learning of active vision for manipulating objects under occlusions. arXiv preprint arXiv:1811.08067 (2018)
5. Connolly, C.: The determination of next best views. In: Proceedings. 1985 IEEE international conference on robotics and automation. vol. 2, pp. 432–435. IEEE (1985)
6. Daudelin, J., Campbell, M.: An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects. IEEE Robotics and Automation Letters **2**(3), 1540–1547 (2017)
7. Denzler, J., Brown, C.M.: Information theoretic sensor data selection for active object recognition and state estimation. IEEE Transactions on pattern analysis and machine intelligence **24**(2), 145–157 (2002)
8. Doumanoglou, A., Kouskouridas, R., Malassiotis, S., Kim, T.K.: Recovering 6d object pose and predicting next-best-view in the crowd. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3583–3592 (2016)
9. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: CVPR (2016)
10. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
11. Haner, S., Heyden, A.: Covariance propagation and next best view planning for 3d reconstruction. In: European Conference on Computer Vision. pp. 545–556. Springer (2012)
12. Isler, S., Sabzevari, R., Delmerico, J., Scaramuzza, D.: An information gain formulation for active volumetric 3d reconstruction. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 3477–3484. IEEE (2016)
13. Kriegel, S., Rink, C., Bodenmüller, T., Suppa, M.: Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. Journal of Real-Time Image Processing **10**(4), 611–631 (2015)
14. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3d proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–8. IEEE (2018)
15. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
16. Meyer, G.P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., Wellington, C.K.: Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12677–12686 (2019)
17. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)

18. Scott, W.R., Roth, G., Rivest, J.F.: View planning for automated three-dimensional object reconstruction and inspection. ACM Computing Surveys (CSUR) **35**(1), 64–96 (2003)
19. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019)
20. Simony, M., Milzy, S., Amendey, K., Gross, H.M.: Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 0–0 (2018)
21. Vasquez-Gomez, J.I., Sucar, L.E., Murrieta-Cid, R., Lopez-Damian, E.: Volumetric next-best-view planning for 3d object reconstruction with positioning error. International Journal of Advanced Robotic Systems **11**(10), 159 (2014)
22. Wang, J., Bartels, J., Whittaker, W., Sankaranarayanan, A.C., Narasimhan, S.G.: Programmable triangulation light curtains. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)
23. Wilkes, D.: Active object recognition (1994)
24. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
25. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
26. Yang, B., Liang, M., Urtasun, R.: Hdnet: Exploiting hd maps for 3d object detection. In: Conference on Robot Learning. pp. 146–155 (2018)
27. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4490–4499 (2018)
28. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection. arXiv preprint arXiv:1908.09492 (2019)
29. Zolfaghari Bengar, J., Gonzalez-Garcia, A., Villalonga, G., Raducanu, B., Aghdam, H.H., Mozerov, M., Lopez, A.M., van de Weijer, J.: Temporal coherence for active learning in videos. arXiv preprint arXiv:1908.11757 (2019)