Active Safety Envelopes using Light Curtains with Probabilistic Guarantees

Siddharth Ancha Gaurav Pathak Srinivasa G. Narasimhan David Held
Carnegie Mellon University, Pittsburgh PA 15213, USA
{sancha, gauravp, srinivas, dheld}@andrew.cmu.edu

Website: https://siddancha.github.io/projects/active-safety-envelopes-with-guarantees

***Abstract*—To safely navigate unknown environments, robots must accurately perceive dynamic obstacles. Instead of directly measuring the scene depth with a LiDAR sensor, we explore the use of a much cheaper and higher resolution sensor: *programmable light curtains*. Light curtains are controllable depth sensors that sense only along a surface that a user selects. We use light curtains to estimate the *safety envelope* of a scene: a hypothetical surface that separates the robot from all obstacles. We show that generating light curtains that sense *random* locations (from a particular distribution) can quickly discover the safety envelope for scenes with unknown objects. Importantly, we produce theoretical safety guarantees on the probability of detecting an obstacle using random curtains. We combine random curtains with a machine learning based model that forecasts and tracks the motion of the safety envelope efficiently. Our method accurately estimates safety envelopes while providing probabilistic safety guarantees that can be used to certify the efficacy of a robot perception system to detect and avoid dynamic obstacles. We evaluate our approach in a simulated urban driving environment and a real-world environment with moving pedestrians using a light curtain device and show that we can estimate safety envelopes efficiently and effectively.**[1]

I. Introduction

Consider a robot navigating in an unknown environment. The environment may contain objects that are arbitrarily distributed, whose motion is haphazard, and that may enter and leave the environment in an undetermined manner. This situation is commonly encountered in a variety of robotics tasks such as autonomous driving, indoor and outdoor robot navigation, mobile robotics, and robot delivery. How do we ensure that the robot moves safely in this environment and avoids collision with obstacles whose locations are unknown a priori? What guarantees can we provide about its perception system being able to discover these obstacles?

Given a LiDAR sensor, the locations of obstacles can be computed from the captured point cloud; however, LiDARs are typically expensive and low-resolution. Cameras are cheaper and high-resolution and 2D depth maps of the environment can be predicted from the images. However, depth estimation from camera images is prone to errors and does not guarantee safety.

[1]Please see our project website for (1) a web-based demo of random curtain analysis, (2) videos showing qualitative results of our method and (3) source code.

An alternative approach is to use *active perception* [3, 4], where only the important and required parts of the scene are accurately sensed, by actively guiding a controllable sensor in an intelligent manner. Specifically, a programmable light curtain [27, 5, 1] is a light-weight *controllable* sensor that detects objects intersecting any user-specified 2D vertically ruled surface (or a 'curtain'). Because they use an ordinary rolling shutter camera, light curtains combine the best of both worlds of passive cameras (high spatial-temporal resolution and lower cost) and LiDARs (accurate detection along the 2D curtain and robustness to scattered media like smoke/fog).

In this work, we propose to use light curtains to estimate the "safety envelope" of a scene. We define the safety envelope as an imaginary, vertically ruled surface that separates the robot from all obstacles in the scene. The region between the envelope and the robot is free space and is safe for the robot to occupy without colliding with any objects. Furthermore, the safety envelope "hugs" the closest object surfaces to maximize the amount of free space between the robot and the envelope. More formally, we define a safety envelope as a 1D depth map that is computed from a full 2D depth map by selecting the closest depth value along each column of the 2D depth map (ignoring points on the ground or above a maximal height). As long as the robot never intersects the safety envelope, it is guaranteed to not collide with any obstacle.

Realizing this concept requires addressing two novel and challenging questions: First, where do we place the curtains without *a priori* knowledge of objects in the scene? The light curtain will only sense the parts of the scene where the curtain is placed. Second, how do we evolve these curtains over time to capture dynamic objects? One approach is to place light curtains at random locations in the unknown scene. Previous work [5] has empirically shown that random light curtains can quickly discover unknown objects. In this work, we develop a systematic framework to generate random curtains that respect the physical constraints of the light curtain device. Importantly, we develop a method that produces theoretical guarantees on the probability of random curtains (from a given distribution) to detect unknown objects in the environment and discover the safety envelope. Such safety guarantees could be used to certify the efficacy of a robot perception system to detect and avoid obstacles.

Once a part of the safety envelope (such as an object's

surface) is discovered, it may be inefficient to keep exploring the scene randomly. Instead, a better strategy is to forecast how the identified safety envelope will move in the next timestep and track it by sensing at the predicted location. We achieve this by training a neural network to forecast the position of the envelope in the next timestep using previous light curtain measurements. However, it is difficult to provide theoretical guarantees for such learning-based systems. We overcome this challenge by combining the deep neural network with random light curtain placements. Using this combination, we are able to estimate the safety envelope efficiently, while furnishing probabilistic guarantees for discovering unknown obstacles. Our contributions are:

1) We develop a systematic framework to generate random curtains that respect the physical constraints of the light curtain device, by extending the "light curtain constraint graph" introduced in prior work [1] (Sec. IV, V-A).
2) We develop a dynamic-programming based approach to produce theoretical safety guarantees on the probability of random curtains discovering unknown objects in the environment (Sec. V-B, VII-A).
3) We combine random light curtains with a machine learning based forecasting model to efficiently estimate safety envelopes (Sec. VI).
4) We evaluate our approach on (1) a simulated autonomous driving environment, and (2) a real-world environment with moving pedestrians. We empirically demonstrate that our approach consistently outperforms multiple baselines and ablation conditions (Sec. VII-B).

II. Related Work

A. Active perception and light curtains

Active perception involves actively controlling a sensor for improved perception [3, 4], such as controlling camera parameters [3], moving a camera to look around occlusions [6], and next-best view planning [7]. The latter refers to approaches that select the best sensing action for specific tasks such as object instance classification [29, 11, 10, 25] and 3D reconstruction [15, 16, 26, 9]. Light curtains were introduced in prior work [27, 5] as an adaptive depth sensor. Prior work has also explored the use of light curtains. Ancha et al. [1] introduced the light curtain constraint graph to compute feasible light curtains. Bartels et al. [5] were the first to empirically use random curtains to quickly discover objects in a scene. However, there are several key differences from our work. First, we solve a very different problem: while Ancha et al. [1] use light curtains to perform active bounding-box object detection in static scenes, whereas we track the safety envelope of scenes with dynamic objects. Although we build upon their constraint graph framework, we make several significant and novel contributions. Our main contribution is the safety analysis of random light curtains, which uses dynamic programming (DP) to produce theoretical guarantees on the probability of discovering objects. Providing theoretical guarantees is essential to guarantee safety, and is typically a hard task for perception systems. These works [1, 5] do not provide any such guarantees. Additionally, we extend its constraint graph (that previously encoded only velocity constraints) to also incorporate acceleration constraints. Finally, we combine the discovery of safety envelopes using random curtains, with an ML approach that efficiently forecasts and tracks the envelope; this combination is novel, and we show that our method outperforms other approaches on this task.

B. Multi-frame depth estimation

There is a large body of prior work on depth estimation across multiple frames [17, 31, 8, 30, 18, 28, 19]. Liu et al. [17] aggregate per-frame depth estimates across frames using Bayesian filtering. Matthies et al. [18] use a similar Bayesian approach, but their method is only applied to controlled scenes and restricted camera motion. Other works [31, 8, 28, 19] use RNNs for predicting depth maps at each frame. All of aforementioned works try to predict the full 2D depth map of the environment from monocular images. To the best of our knowledge, we are the first to use a controllable sensor to directly estimate the safety envelope of the scene.

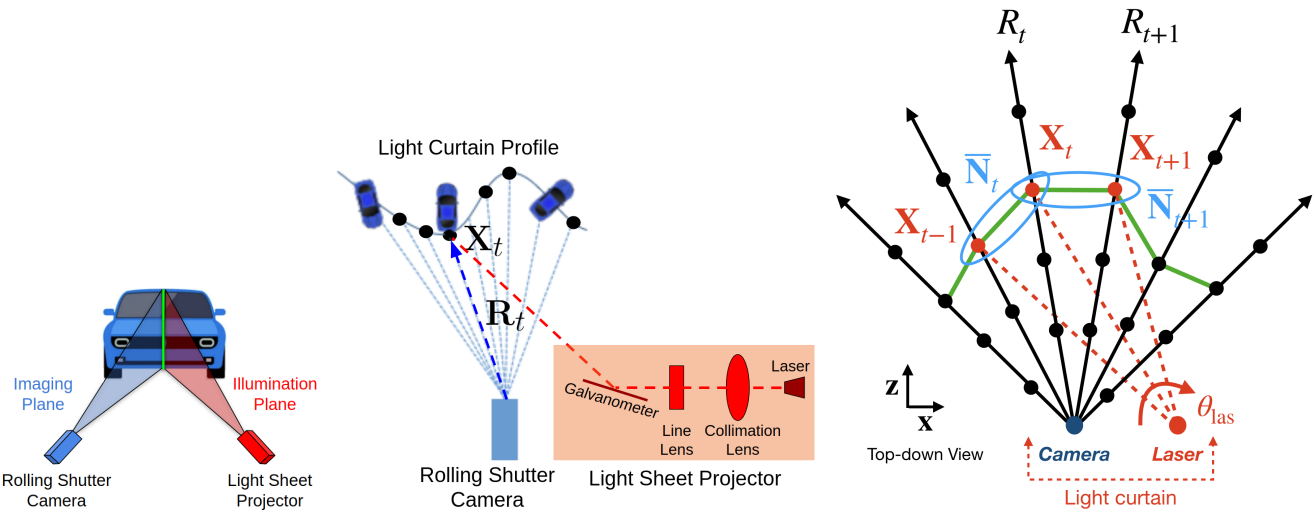
C. Safe navigation

Many approaches for safety guaranteed navigation use 3D sensors like LiDARs [22, 21, 24] and/or cameras [20, 2]. The sensor data is converted to occupancy grids/maps [22, 21, 2]; safety and collision avoidance guarantees are provided for planning under these representations. Other works use machine learning models to recognize unsafe, out-of-distribution inputs [20] or learning to predict collision probabilites [22, 21]. Our work of estimating the safety envelope using a light curtain is orthogonal to these works and can leverage those methods for path planning and obstacle avoidance.

III. Background on Light Curtains

Programmable *light curtains* [27, 5, 1] are a recently developed sensor for controllable depth sensing. "Light curtains" can be thought of as virtual surfaces placed in the environment that detect points on objects intersecting this surface. The working principle is illustrated in Fig. 1(a, b). The device sweeps a vertically ruled surface by rotating a light sheet laser using a galvo-mirror synchronously with the vertically aligned camera's rolling shutter. Object points intersecting a vertical line of the ruled surface are imaged brightly in the corresponding camera column. We denote the top-down projection of the imaging plane corresponding to the t-th pixel column as a "camera ray" R_t. The rolling shutter camera successively activates each image plane (column), corresponding to rays $R_1, \ldots, R_T$ from left to right, with a time difference of Δt between successive activations. The top-down projection of the vertical line intersecting the t-th imaging plane lies on R_t and will be referred to as a "control point" $\mathbf{X}_t$.

Input: A light curtain is uniquely defined by where it intersects each camera ray R_t in the top-down view, i.e. the set of control points $(\mathbf{X}_1, \cdots, \mathbf{X}_T)$, one for each camera ray. This is the input to the light curtain device. Then, to image $\mathbf{X}_t$



(a) Working principle (b) Optical schematic (top view) (c) Extended constraint Graph

Fig. 1: (a, b) Illustration of programmable light curtains adapted from [1]. (a) An illumination plane (from the projector) and an imaging plane (of the camera) intersect to produce a light curtain. (b) A controllable galvanometer mirror rotates synchronously with a rolling shutter camera and images the points of intersection. (c) Light curtain constraint graph with our proposed extension. Black dots are control points that can be imaged, and blue ovals are extended nodes that contain two control points. Any path in this graph (shown in green) is a valid light curtain that satisfies velocity and acceleration constraints.

on camera ray R_t, the galvo-mirror is programmed to rotate by an angle of $\theta(\mathbf{X}_t)$ that is required for the laser sheet to intersect R_t at $\mathbf{X}_t$. By specifying a control point $\mathbf{X}_t$ for each camera ray, the light curtain device can be made to image any vertically ruled surface [27, 5].

Output: The light curtain outputs an intensity value for each camera pixel. Since a light curtain profile is specified by a control point $\mathbf{X}_t$ for every camera ray R_t in the top-down view, we compute the maximum pixel intensity value $\mathbf{I}_t$ of the t-th pixel column and treat this as the output of the light curtain for the corresponding ray R_t.

IV. Generating Feasible Light Curtains

The set of realizable curtains depend on the physical constraints imposed by the real device. The rotating galvo-mirror can operate at a maximum angular velocity $\omega_{\max}$ and a maximum angular acceleration $\alpha_{\max}$. Let $\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}$ be the control points imaged by the light curtain on three consecutive camera rays. These induce laser angles $\theta(\mathbf{X}_{t-1}), \theta(\mathbf{X}_t), \theta(\mathbf{X}_{t+1})$ respectively. Let $\mathbf{\Omega}_t = (\theta(\mathbf{X}_t) - \theta(\mathbf{X}_{t-1}))/\Delta t$ be the angular velocity of the galvo-mirror at R_t. Its angular acceleration at R_t is $(\mathbf{\Omega}_{t+1} - \mathbf{\Omega}_t)/\Delta t = (\theta(\mathbf{X}_{t+1}) + \theta(\mathbf{X}_{t-1}) - 2\cdot\theta(\mathbf{X}_t))/(\Delta t)^2$. Then, the light curtain velocity and acceleration constraints, in terms of the control points $\mathbf{X}_t$, are:

$$|\theta(\mathbf{X}_t) - \theta(\mathbf{X}_{t-1})| \leq \omega_{\max} \cdot \Delta t \qquad 2 \leq t \leq T \tag{1}$$

$$|\theta(\mathbf{X}_{t+1}) + \theta(\mathbf{X}_{t-1}) - 2\theta(\mathbf{X}_t)| \leq \alpha_{\max} \cdot (\Delta t)^2 \qquad 2 \leq t < T \tag{2}$$

In order to compute feasible light curtains that satisfy physical constraints, Ancha et al. [1] introduced a "light curtain constraint graph," denoted by $\mathcal{G}$. $\mathcal{G}$ has two components: a set of nodes $\mathbf{N}_t$ associated with each camera ray and edges between nodes $\mathbf{N}_t$ and $\mathbf{N}_{t+1}$ on consecutive camera rays. Nodes are designed to store information that fully captures the *state* of the galvo-mirror when imaging the ray R_t. An edge exists from $\mathbf{N}_t$ to $\mathbf{N}_{t+1}$ *iff* the galvo-mirror is able to transition from the state defined by $\mathbf{N}_t$ to the state defined by $\mathbf{N}_{t+1}$ without violating any of velocity constraints (Eqn. (1)).

Ancha et al. [1] defined the state to be $\mathbf{N}_t = \mathbf{X}_t$ (i.e. contain only one control point). But this representation does not ensure that acceleration constraints of Eqn. 2, that depend on three consecutive control points, are satisfied. This can produce light curtain profiles which require the galvo-mirror to change its angular velocity more abruptly than its physical torque limits can allow, resulting in hardware errors. Thus, we extend the definition of a node $\mathbf{N}_t$ at t to store control points on the current ray *and* the previous ray, as $\overline{\mathbf{N}}_t = (\mathbf{X}_{t-1}, \mathbf{X}_t)$ (see Fig. 1 (c)). Intuitively, $\overline{\mathbf{N}}$ contains information about the angular position and velocity of the galvo-mirror. This allows us to incorporate acceleration constraints by creating an edge between nodes $(\mathbf{X}_{t-1}, \mathbf{X}_t)$ and $(\mathbf{X}_t, \mathbf{X}_{t+1})$, *iff* $\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}$ satisfy the velocity and acceleration constraints defined in Equations (1, 2). Thus, any path in the extended graph $\overline{\mathcal{G}}$ represents a feasible light curtain that satisfies both constraints. The acceleration constraints also serve to limit the increase in the number of nodes with feasible edges, keeping the graph size manageable.

V. Random curtains & theoretical guarantees

Recall that the light curtain will only sense the parts of the scene where the curtain is placed. Thus we must decide where to place the curtain in order to sense the scene and estimate the safety envelope. Our proposed method uses a combination of random curtains as well as learned forecasting to estimate the safety envelope of an unknown scene. In this section, we show how a random curtain can be sampled from the extended constraint graph $\overline{\mathcal{G}}$ and how to analytically compute the probability of a random curtain detecting an obstacle, which helps to probabilistically guarantee obstacle detection of our overall method.

A. Sampling random curtains from the constraint graph

First, we need to define a probability distribution over the set of valid curtains in order to sample from it. We do so by defining, for each node $\overline{\mathbf{N}}_t = (\mathbf{X}_{t-1}, \mathbf{X}_t)$, a *transition probability distribution* $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$. This denotes the probability of transitioning from imaging the control points $\mathbf{X}_{t-1}, \mathbf{X}_t$ on the previous and current camera rays to the control point $\mathbf{X}_{t+1}$ on the next ray. We constrain $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ to equal 0 if there is no edge from node $(\mathbf{X}_{t-1}, \mathbf{X}_t)$ to node $(\mathbf{X}_t, \mathbf{X}_{t+1})$; an edge will exist *iff* the transition $\mathbf{X}_{t-1} \to \mathbf{X}_t \to \mathbf{X}_{t+1}$ satisfies the light curtain constraints. Thus, $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ defines a probability distribution over the neighbors of $\overline{\mathbf{N}}_t$ in the constraint graph.

The transition probability distribution enables an algorithm to sequentially generate a random curtain. We begin by sampling the control points $\overline{\mathbf{N}}_2 = (\mathbf{X}_1, \mathbf{X}_2)$ according to an initial probability distribution $P(\overline{\mathbf{N}}_2)$. At the t-th iteration, we sample $\mathbf{X}_{t+1}$ according to the transition probability distribution $\mathbf{X}_{t+1} \sim P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ and add $\mathbf{X}_{t+1}$ to the current set of sampled control points. After $(T-1)$ iterations, this process generates a full random curtain. Pseudo-code for this process is found in Algorithm 1 in Appendix A. Our random curtain sampling process provides the flexibility to design any initial and transition probability distribution. See Appendix A for a discussion on various choices of the transition probability distribution, where we also provide a theoretical and empirical justification to use one distribution in particular.

B. Theoretical guarantees for random curtains

In this section, we first describe a procedure to detect objects in a scene using the output of a random light curtain placement. Then, we develop a method that runs dynamic programming on $\overline{\mathcal{G}}$ to analytically compute a random curtain's probability of detecting a specific object. This provides probabilistic safety guarantees on how well a random curtain can discover the safety envelope of an object.

Detection using light curtains: Consider an object in the scene whose visible surface intersects each camera ray at the positions $O_{1:T}$. This representation captures the position, shape and size of the object from the top-down view. Let $\{\mathbf{X}_t\}_{t=1}^T$ be the set of control points for a light curtain placed in the scene. The light curtain will produce an intensity $\mathbf{I}_t(\mathbf{X}_t, O_t)$ at each control point $\mathbf{X}_t$ that is sampled by the light curtain device. Note that $\mathbf{I}_t$ is a function of the position of the object as well as the position of the light curtain; the intensity increases as the distance between $\mathbf{X}_t$ and O_t reduces and is the highest when $\mathbf{X}_t$ and O_t coincide. We say that an object has been detected at control point $\mathbf{X}_t$ if the intensity $\mathbf{I}_t$ is above a threshold τ; the intensity threshold is used to account for noise in the image. We define a binary detection variable to indicate whether a detection of an object occurred at position O_t at control point $\mathbf{X}_t$ as $\mathbf{D}_t(\mathbf{X}_t, O_t) = [\mathbf{I}_t(\mathbf{X}_t, O_t) > \tau]$, where $[\cdot]$ is the indicator function. We declare that an object has been detected by a light curtain if it is detected on any of its control points. Formally, we define a binary detection variable to indicate whether a detection of object $O_{1:T}$ occurred at any of its control point $\mathbf{X}_{1:T}$ as $\mathbf{D}(\mathbf{X}_{1:T}, O_{1:T}) = \bigvee_{t=1}^T \mathbf{D}_t(\mathbf{X}_t, O_t)$, (where $\bigvee$ is 'logical or' operator). Our objective is then to compute the *detection probability*, denoted as $P(\mathbf{D}(\mathbf{X}_{1:T}, O_{1:T}))$, which is the probability that a curtain sampled from $\overline{\mathcal{G}}$ (using the sampling procedure described in Sec. V-A) will detect the object $O_{1:T}$; below we will use the simpler notation $P(\mathbf{D})$ to denote the detection probability.

Theoretical guarantees using dynamic programming: The simplest method to compute the detection probability for a given object is to sample a large number of random curtains and output the average number of curtains that detect the object. However, a large number of samples would be needed to provide accurate estimates of the detection probability; further, this procedure is stochastic and the probability estimate will only be approximate. Instead, we propose utilizing the known structure of the constraint graph and the transition probabilities; we will apply dynamic programming to compute the detection probability both efficiently and analytically.

Our analytic method for computing the detection probability proceeds as follows: we first compute the value of the detection event $\mathbf{D}_t(\mathbf{X}_t, O_t)$ at every possible control point $\mathbf{X}_t$ that is part of a node in the constraint graph $\overline{\mathcal{G}}$ i.e. we compute whether or not a curtain placed at $\mathbf{X}_t$ is able to detect the object. Given the positions $O_{1:T}$ of an object, as well as the physical properties of the light curtain device (intrinsics of the camera and the power, thickness and divergence of the laser beam), we use a light curtain simulator to compute $\mathbf{I}_t(\mathbf{X}_t, O_t)$ using standard raytracing and rendering, for any arbitrary control point $\mathbf{X}_t$.

To compute the detection probability $P(\mathbf{D})$, we first define the notion of a "sub-curtain," which is a subset of the control points $\mathbf{X}_{t:T}$ which start at ray R_t and ends on ray R_T. We can decompose the overall problem of computing $P(\mathbf{D})$ into simpler sub-problems by defining the *sub-curtain detection probability* $P_{\text{det}}(\mathbf{X}_{t-1}, \mathbf{X}_t)$. This is the probability that any random sub-curtain starting at $(\mathbf{X}_{t-1}, \mathbf{X}_t)$ and ending on the last camera ray R_T detects the object O at some point between rays R_t and R_T. Using this definition, we can write the sub-curtain detection probability as $P_{\text{det}}(\mathbf{X}_{t-1}, \mathbf{X}_t) = P(\bigvee_{t'=t}^T \mathbf{D}_{t'}(\mathbf{X}_{t'}, O_{t'}) \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$.

Note that the overall curtain detection probability can be written in terms of the sub-curtain detection probabilities of

the second ray (the first set of nodes in the graph) as

$$P(\mathbf{D}) = \sum_{\mathbf{X}_1, \mathbf{X}_2} P_{\text{det}}(\mathbf{X}_1, \mathbf{X}_2)\ P(\mathbf{X}_1, \mathbf{X}_2). \tag{3}$$

This is the sum of the detection probabilities of a random curtain starting from the initial nodes $P_{\text{det}}(\mathbf{X}_1, \mathbf{X}_2)$, weighted by the probability of the nodes being sampled from the initial distribution $P(\mathbf{X}_1, \mathbf{X}_2)$. Conveniently, the sub-curtain detection probabilities satisfy a simple recursive equation:

$$P_{\text{det}}(\mathbf{X}_{t-1}, \mathbf{X}_t) = \begin{cases} 1 & \text{if } \mathbf{D}_t(\mathbf{X}_t, O_t) = 1 \\ \sum_{\mathbf{X}_{t+1}} P_{\text{det}}(\mathbf{X}_t, \mathbf{X}_{t+1})\ P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t) & \text{otherwise} \end{cases} \tag{4}$$

Intuitively, if the control point $\mathbf{X}_t$ is able to detect the object, then the sub-curtain detection probability is 1 regardless of how the sub-curtain is placed on the later rays. If not, then the detection probability should be equal to the sum of the sub-curtain detection probabilities of the nodes the curtain transitions to, weighted by the transition probabilities.

This recursive relationship can be exploited by successively computing the sub-curtain detection probabilities from the last ray to the first. The sub-curtain detection probabilities on the last ray will simply be either 1 or 0, based on whether the object is detected there or not. After having computed sub-curtain detection probabilities for all rays between $t+1$ and T, the probabilities for nodes at ray t can be computed using the above recursive formula (Eqn. 4). Finally, after obtaining the probabilities for nodes on the second ray, the overall curtain detection probability can be computed as described using Eqn. 3. Pseudocode for this method can be found in Algorithm 2 in Appendix B. A discussion on the computational complexity of the extended constraint graph $\overline{\mathcal{G}}$ (which contains more nodes and edges than $\mathcal{G}$) can be found in Appendix C.

We have created a web-based demo (available on the project website) that computes the probability of a random curtain detecting an object with a user-specified shape in the top-down view. It also performs analysis of the detection probability as a function of the number of light curtains placed. In Section VII-A, we use this method to analyze the detection probability of random curtains as a function of the object size and number of curtain placements. We compare it against a sampling-based approach and show that our method gives the same results but with an efficient analytical computation.

VI. Learning to Forecast Safety Envelopes

Random curtains can help discover the safety envelope of unknown objects in a scene. However, once a part of the envelope is discovered, an efficient way to estimate the safety envelope in future timesteps is to *forecast* how the envelope will move with time and *track* the envelope by placing a new light curtain at the forecasted locations. In this section, we describe how to train a deep neural network to forecast safety envelopes and combine them with random curtains.

Problem setup: We call any algorithm that attempts to forecast the safety envelope as a "forecasting policy". We assume that a forecasting policy is provided with the ground truth safety envelope of the scene in the first timestep. In the real world, this can be done by running one of the less efficient baseline methods once when the light curtain is first started, until the light curtain is initialized. After the initialization, the learning-based method is used for more efficient light curtain tracking. The policy always has access to all previous light curtain measurements. At every timestep, the policy is required to forecast the location of the safety envelope for the next timestep. Then, the next light curtain will be placed at the forecasted location. To leverage the benefits of random curtains that can discover unknown objects, we place random light curtains while the forecasting method predicts the safety envelope of the next timestep. We allow random curtains to override the forecasted curtain: if the random curtain obtains an intensity $\mathbf{I}_t$ on camera ray R_t that is above a threshold τ, the control point of the forecasted curtain for ray R_t is immediately updated to that of the random curtain, i.e. the random curtain overrides the forecasted curtain if the random curtain detects an object. See Appendix E for details of our efficient, parallelized implementation of random curtain placement and forecasting, as well as an analysis of the pipeline's runtime.

Handcrafted policy: First, we define a simple, hand-specified light curtain placement policy; this policy will serve both as a baseline and as an input to our neural network, described below. The policy conservatively forecasts a fixed decrease in the depth of the safety envelope for ray R_t if the ray's intensity $\mathbf{I}_t$ is above a threshold (indicative of the presence of an object), and forecasts a fixed increase in depth otherwise. By alternating between increasing and decreasing the depth of the forecasted curtain, this policy can roughly track the safety envelope. However, since the forecasted changes in depth are hand-defined, it is not designed to handle large object motions, nor will it accurately converge to the correct depth for stationary objects.

Neural network forecasting policy: We use a 2D convolutional neural network to forecast safety envelopes in the next timestep. It takes as input (1) the intensities $\mathbf{I}_t$ returned by previous k light curtain placements, (2) the positions of the previous k light curtain placements, and (3) the outputs of the handcrafted policy described above (this helps avoid local minima during training and provides useful information to the network). For more details about the architecture of our network, please see Appendix D.

We assume access to ground truth safety envelopes at training time. This can be directly obtained in simulated environments or from auxiliary sensors such as LiDAR in the real world. Because a light curtain is an active sensor, the data that it collects depends on the forecasting policy. Thus to train our network, we use DAgger [23], a widely-used imitation learning algorithm to train a policy with expert or ground-truth supervision across multiple timesteps. We use the Huber loss [14] between the predicted and ground truth safety

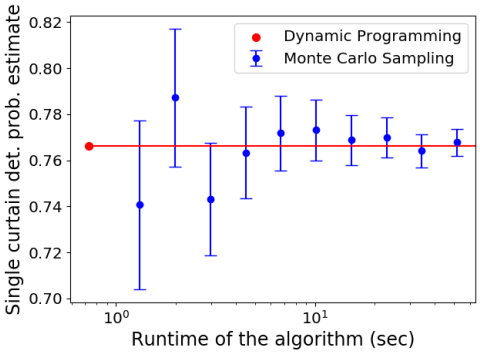


Fig. 2: Comparison of our dynamic programming approach with a Monte Carlo sampling based approach to estimate the probability of a random curtain detecting an object of size $2m \times 2m$. The x-axis shows the runtime of the methods (in seconds), and the y-axis shows estimated detection probability. Our method (in red) is both *exact* and fast; it quickly produces a single and precise point estimate. Monte Carlo sampling (in blue) samples a large number of random curtain and returns the average number of curtains that intersect the object. The estimate is stochastic and the 95%-confidence intervals are shown in blue. While the Monte Carlo estimate eventually converges to the true probability, it is inherently noisy and orders of magnitude slower than our method.

envelopes as our training loss. The Huber loss is designed to produce stable gradients while being robust to outliers.

VII. Experiments

A. Random curtain analysis

In this section, we use the dynamic programming approach introduced in Section V-B to analyze the detection probability of random curtains. First, we compare our dynamic programming method to an alternate approach to compute detection probabilities: Monte Carlo sampling. This method involves sampling a large number of random curtains and returning the average number of curtains that were able to detect the object. This produces an unbiased estimate of the single-curtain detection probability, with a variance based on the number of samples. However, our dynamic programming approach has multiple advantages over such a sampling-based approach:

1) Dynamic programming produces an *analytic* estimate of the detection probability, whereas sampling produces a *stochastic*, noisy estimate of the probability. Analytic estimates are useful for reliably evaluating the safety and robustness of perception systems.
2) Dynamic programming is significantly more efficient than sampling based approaches. The former only involves one pass through the constraint graph. In contrast, a large number of samples may be required to provide a reasonable estimate of the detection probability.

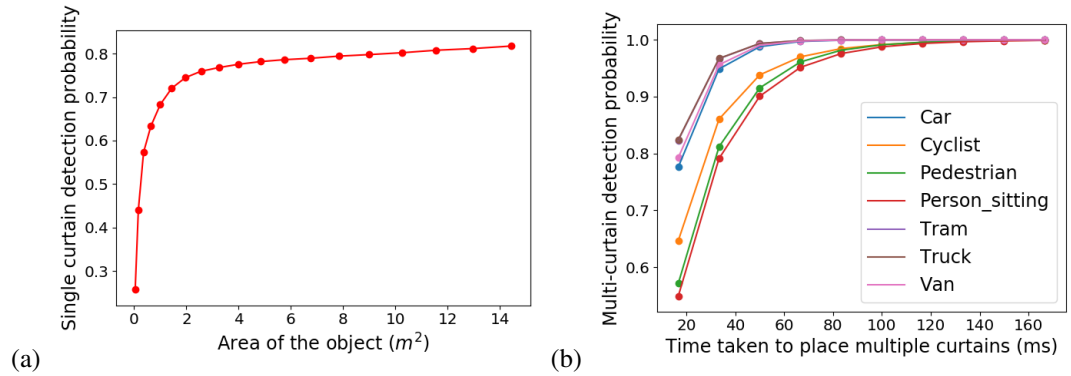
The two methods are compared in Figure 2, which shows the estimated single-curtain detection probabilities of both methods as a function of the runtime of each method (the runtimes include pre-processing steps such as raycasting, and hence are directly comparable between the two methods). Dynamic programming (shown in red) produces an analytic estimate very efficiently (around 0.8 seconds). For Monte Carlo sampling, we show the probability estimate for a varying number of Monte Carlo samples. Each run shows the mean estimate of the detection probability (blue dots), as well as its corresponding 95%-confidence intervals (blue bars). Using more samples produces more accurate estimates with smaller confidence intervals, at the cost of increased runtime. The sampling approach will eventually converge to the point estimate output by dynamic programming in the limit of an infinite number of samples. This experiment shows that dynamic programming produces precise estimates (i.e. there is zero uncertainty in its estimate) while being orders of magnitude faster than Monte Carlo sampling.

Next, we investigate how the size of an object affects the detection probability. We generate objects of varying sizes and run our dynamic programming algorithm to compute their detection probabilities. Figure 3 (a) shows a plot of the detection probability of a single curtain as a function of the area of the object (averaged over multiple object orientations). As one would expect, the figure shows that larger objects are detected with higher probability.

Last, we analyze the detection probability as a function of the number of light curtains placed. The motivation for using multiple curtains to detect objects is the following. A single curtain might have a low detection probability p, especially for a small object. However, we could place multiple (say n) light curtains and report a successful detection if at least one of the n curtains detects the object. Then, the probability of detection increases exponentially by $1-(1-p)^n$. We call this the "multi-curtain" detection probability. Figure 3 (b) shows the multi-curtain detection probabilities for objects from the KITTI [13] dataset, as a function of the time taken to place those curtains (at 60 Hz). For each object class, we construct a "canonical object" by averaging the dimensions of all labeled instances of that class in the KITTI dataset. We can see that larger object classes are detected with a higher probability, as expected. The figure also shows that the probability increases rapidly with the number of random curtains for all object classes. Four random curtains (which take about 67ms to image) are sufficient to detect objects from all classes with at least 90% probability. Note that there is a tradeoff between detection probability and runtime of multiple curtains; guaranteeing a high probability requires more time for curtains to be placed.

B. Estimating safety envelopes

Environments: In this section, we evaluate our approach to estimate safety envelopes using light curtains, in two environments. First, we use SYNTHIA [33], a large, simulated dataset containing photorealistic scenes of urban driving scenarios. It consists of 191 training scenes ($\sim 96K$ frames) and 97 test scenes ($\sim 45K$) frames and provides ground truth depth maps. Second, we perform safety envelope estimation in a



(a) (b)

Fig. 3: (a) The probability of random curtains detecting objects of various areas. For an object of a fixed area, we average the probability across various orientations of the object. Larger objects are detected with higher probability. (b) We show the detection probability of canonical objects from classes in the KITTI [13] dataset. For each object class, we construct a "canonical object" by averaging the dimensions of all labeled instances of that class in the KITTI dataset. Larger object classes are detected with a higher probability, as expected. We also show the detection probability as a function of the number of light curtains placed. The detection probability increases exponentially with the number of light curtain placements.

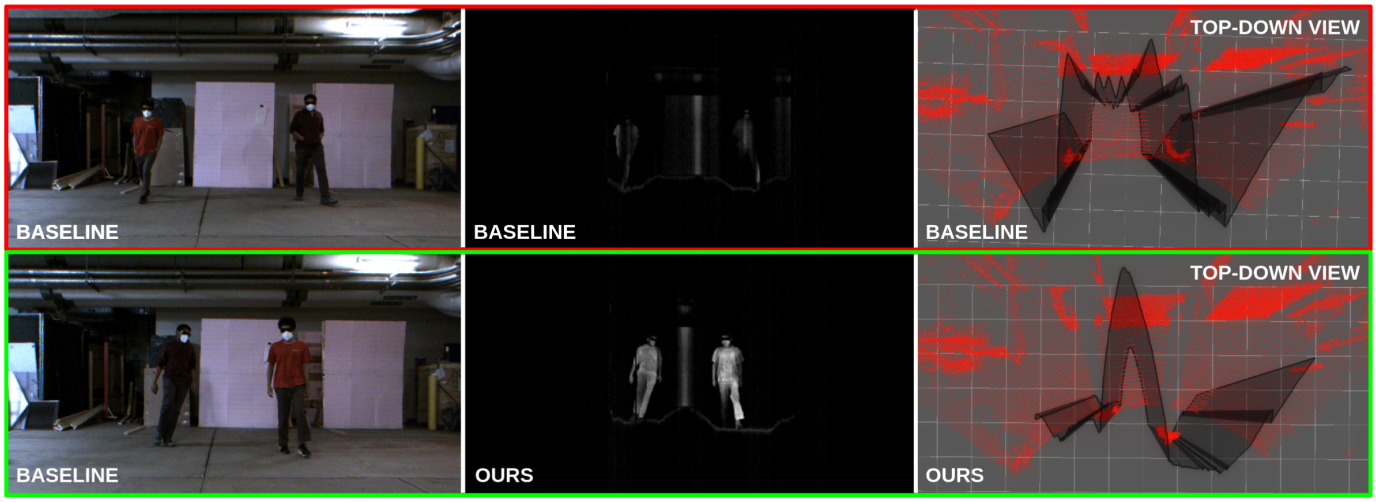


Fig. 4: Qualitative results in a real-world environment with two walking pedestrians, comparing a hand-crafted baseline policy (top-row) with our method (bottom-row). *Left column:* contains RGB scene images. *Middle column:* contains the light curtain images, where higher intensity means a closer intersection between the light curtain and the object surfaces (i.e. a better estimation of the safety envelope). Since our method learns to forecast the safety envelope, it estimates the envelope more accurately and produces higher light curtain intensity. *Right column* (top-down view): the black surfaces are the estimated safety envelopes, and red points show a LiDAR point cloud (only used to aid visualization). Our forecasting method's estimate of the safety envelope hugs the pedestrians more tightly and looks smoother. The hand-crafted baseline works by continuously moving the curtain back and forth, creating a jagged profile and preventing it from enveloping objects tightly.

	Huber loss	RMSE Linear	RMSE Log	RMSE Log Scale-Inv.	Absolute Relative Diff.	Squared Relative Diff.	Thresh (1.25)	Thresh (1.25^2)	Thresh (1.25^3)
	↓	↓	↓	↓	↓	↓	↑	↑	↑
Handcrafted baseline	0.1145	1.9279	0.1522	0.0721	0.1345	1.0731	0.6847	0.7765	0.8022
Random curtain only	0.1484	2.2708	0.1953	0.0852	0.1698	1.2280	0.6066	0.7392	0.7860
1D-CNN	0.0896	1.7124	0.1372	0.0731	0.1101	0.7076	0.7159	0.7900	0.8138
1D-GNN	0.1074	1.6763	0.1377	0.0669	0.1256	0.8916	0.7081	0.7827	0.8037
Ours w/o Random curtains	0.1220	2.0332	0.1724	0.0888	0.1411	0.9070	0.6752	0.7450	0.7852
Ours w/o Forecasting	0.0960	1.7495	0.1428	0.0741	0.1163	0.6815	0.7010	0.7742	0.8024
Ours w/o Baseline input	0.0949	1.8569	0.1600	0.0910	0.1148	0.7315	0.7082	0.7740	0.7967
Ours	**0.0567**	**1.4574**	**0.1146**	**0.0655**	**0.0760**	**0.3662**	**0.7419**	**0.8035**	**0.8211**

TABLE I: Performance of safety envelope estimation on the SYNTHIA [33] urban driving dataset under various metrics.

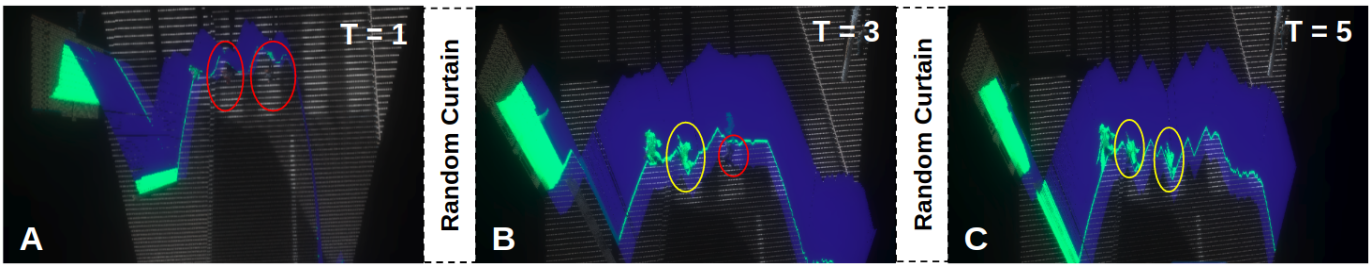


Fig. 5: We illustrate the benefits of placing random curtains (that come with probabilistic guarantees of obstacle detection) while estimating safety envelopes, shown in SYNTHIA [33], a simulated urban driving environment. The blue surfaces are the estimated safety envelopes, and the green points show regions of high light curtain intensity (higher intensity corresponds to better estimation). There are three pedestrians in the scene. (a) Our forecasting model fails to locate two pedestrians (red circles). (b) The first random curtain leads to the discovery of one pedestrian (yellow). (c) The second random curtain helps discover the other pedestrian (second yellow circle). The safety envelope of all pedestrians has now been detected.

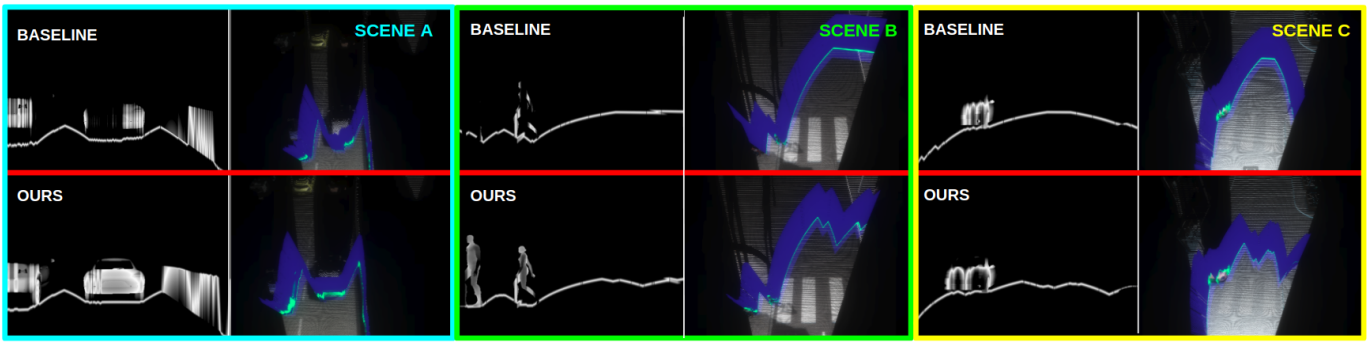


Fig. 6: Comparison of the safety envelope estimation between a hand-crafted baseline policy (top row) and a trained neural network (bottom row), in three simulated urban driving scenes from the SYNTHIA [33] dataset. For each scene and method, the left column shows the intensity image of the light curtain; higher intensities correspond to closer intersection of the light curtain and object surfaces, implying better estimation of the safety envelope. The right column shows the light curtain profile in the scene. The trained network estimates the safety envelopes more accurately than the handcrafted baseline policy.

		Huber loss	RMSE Linear	RMSE Log	RMSE Log Scale-Inv.	Absolute Relative Diff.	Squared Relative Diff.	Thresh (1.25)	Thresh (1.25^2)	Thresh (1.25^3)
		↓	↓	↓	↓	↓	↓	↑	↑	↑
Slow Walking	Handcrafted baseline	0.0997	0.9908	0.1881	**0.1015**	0.1371	0.2267	0.8336	**0.9369**	**0.9760**
	Ours	**0.0630**	**0.9115**	**0.1751**	0.1083	**0.0909**	**0.1658**	**0.8660**	0.9228	0.9694
Fast Walking	Handcrafted baseline	0.1473	1.2425	0.2475	0.1508	0.1824	0.3229	0.6839	0.8774	**0.9702**
	Ours	**0.0832**	**0.9185**	**0.1870**	**0.1201**	**0.1132**	**0.2093**	**0.8575**	**0.9165**	0.9610

TABLE II: Performance of safety envelope estimation in a real-world dataset with moving pedestrians. The environment consisted of two people walking in both back-and-forth and sideways motions.

real-world environment with moving pedestrians. These scenes consist of two people walking in front of the device in complicated, overlapping trajectories. We perform evaluations in two settings: a *"Slow Walking"* setting, and a harder *"Fast Walking"* setting where forecasting the motion of the safety envelope is naturally more challenging. We use an Ouster OS2 128-beam LiDAR (and ground-truth depth maps for the SYNTHIA dataset) to compute ground truth safety envelopes for training and evaluation. We evaluate policies over a horizon of 50 timesteps in both environments.

Evaluation metrics: Safety envelopes can be thought of as 1D depth maps computed from a full 2D depth map, since the safety envelope is constrained to be a vertically ruled surface that always "hugs" the closest obstacle. Thus, the safety curtain can be computed by selecting the closest depth value along each column of a 2D depth map (ignoring points on the ground or above a maximal height). Because of the relationship between the safety envelope and the depth map, we evaluate our method using a variety of standard metrics from the single-frame depth estimation literature [32, 12]. The metrics are averaged over multiple timesteps to evaluate the policy's performance across time.

Baselines: In Table I, we compare our method to the hand-crafted policy described in Sec. VI. A 'random curtain only' baseline tests the performance of random curtains for safety envelope estimation in the absence of any forecasting policy.

We also compare our method against two other neural network architectures that forecast safety envelopes: a CNN that performs 1D convolutions, and a graph neural network with nodes corresponding to camera rays. Please see Appendix D for more details about their network architectures. See Table I for a comparison of our method with the baselines in the SYNTHIA environment, and Table II for the real-world environment. The arrows below each metric in the second row denote whether a higher value (↑) or lower value (↓) is better. In both environments (simulated and real), our method outperforms the baselines on most metrics, often by a significant margin.

Ablations: We also perform multiple ablation experiments. First, we train and evaluate our without using random curtains (Tab. I, "Ours w/o Random Curtains"). This reduces the performance by a significant margins, suggesting that it is crucial to combine forecasting with random curtains for increased robustness. See Appendix F for more experiments performed without using random curtains for all the other baselines and ablation conditions. Second, we perform an ablation in which we train our model without forecasting to the next timestep i.e. the network is only trained to predict the safety envelope of the current timestep (Tab. I, "Ours w/o Forecasting"). This leads to a drop in performance, suggesting that it is important to place light curtains at the locations where the safety envelope is expected to move to, not where it currently is. Finally, we modify our method to not take the output of the hand-crafted policy as input (Tab. I, "Ours w/o Baseline input"). The drop in performance shows that providing the neural network access to another policy that performs reasonably well helps with training and improves performance.

Qualitative anaysis: We perform qualitative analysis of our method in the real-world environment with moving pedestrians in Fig. 4, and in the SYNTHIA [33] simulated environment in Figs. 5, 6. We compare our method against the hand-crafted baseline, as well as show how placing random curtains can discover objects and improve the estimation of safety envelopes. Please see captions for more details. Our project website contains videos demonstrating the qualitative performance of our method in the real-world pedestrian environment. They show that our method can generalize to multiple obstacles (as many as five pedestrians) and extremely fast and spontaneous motion, even though such examples were not part of the training set.

VIII. Conclusion

In this work, we develop a method to estimate the safety envelope of a scene, which is a hypothetical vertical surface that separates a robot from all obstacles in the environment. We use light curtains, an actively controllable, resource-efficient sensor to directly estimate the safety envelope. We describe a method to generate random curtains that respect the physical constraints of the device, in order to quickly discover the safety envelope of an unknown object. Importantly, we develop a dynamic-programming based approach to produce theoretical safety guarantees on the probability of random curtains detecting objects in the scene. We combine this method with a machine-learning based model that forecasts the motion of already-discovered safety envelopes to efficiently track them. This enables our robot perception system to accurately estimate safety envelopes, while our probabilistic guarantees help certify its accuracy and safety towards obstacle detection and avoidance.

Acknowledgements

We thank Adithya Pediredla, N. Dinesh Reddy and Zelin Ye for help with real-world experiments. 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.

References

[1] Siddharth Ancha, Yaadhav Raaj, Peiyun Hu, Srinivasa G. Narasimhan, and David Held. Active perception using light curtains for autonomous driving. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 751–766, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58558-7. URL http://siddancha.github.io/projects/active-perception-light-curtains/.

[2] Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE, 2019. URL https://ieeexplore.ieee.org/abstract/document/9030133.

[3] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988. URL https://ieeexplore.ieee.org/abstract/document/5968.

[4] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2): 177–196, 2018. URL https://link.springer.com/article/10.1007/s10514-017-9615-3.

[5] Joseph R Bartels, Jian Wang, William Whittaker, Srinivasa G Narasimhan, et al. Agile depth sensing using triangulation light curtains. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7900–7908, 2019. URL http://www.cs.cmu.edu/~ILIM/agile_depth_sensing/html/index.html.

[6] Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *Conference on Robot Learning*, pages 422–431. PMLR, 2018. URL http://proceedings.mlr.press/v87/cheng18a.html.

[7] Cl Connolly. The determination of next best views. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 432–435. IEEE, 1985. URL https://ieeexplore.ieee.org/abstract/document/1087372.

[8] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. Depthnet: A recurrent neural network architecture for monocular depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 283–291, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w9/Kumar_DepthNet_A_Recurrent_CVPR_2018_paper.pdf.

[9] Jonathan Daudelin and Mark Campbell. An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects. *IEEE Robotics and Automation Letters*, 2(3):1540–1547, 2017. URL https://scholar.google.com/scholar?cluster=7456760468603259697&hl=en&as_sdt=5,39&sciodt=0,39.

[10] Joachim Denzler and Christopher M Brown. 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. URL https://ieeexplore.ieee.org/abstract/document/982896.

[11] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3583–3592, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Doumanoglou_Recovering_6D_Object_CVPR_2016_paper.html.

[12] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/7bccfde7714a1ebadf06c5f4cea752c1-Paper.pdf.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. URL http://www.cvlibs.net/datasets/kitti/.

[14] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992. URL https://www.jstor.org/stable/2238020?seq=1.

[15] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3477–3484. IEEE, 2016. URL https://ieeexplore.ieee.org/abstract/document/7487527.

[16] Simon Kriegel, Christian Rink, Tim Bodenmüller, and Michael Suppa. Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. *Journal of Real-Time Image Processing*, 10(4): 611–631, 2015. URL https://link.springer.com/article/10.1007/s11554-013-0386-6.

[17] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10986–10995, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Liu_Neural_RGBrD_Sensing_Depth_and_Uncertainty_From_a_Video_Camera_CVPR_2019_paper.pdf.

[18] Larry Matthies, Richard Szeliski, and Takeo Kanade. Depth maps from image sequences1. URL https://www.ri.cmu.edu/pub_files/pub2/matthies_l_1988_1/matthies_l_1988_1.pdf.

[19] Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. Dont forget the past: Recurrent depth estimation from monocular video. *IEEE Robotics and*

Automation Letters, 5(4):6813–6820, 2020. URL https://arxiv.org/pdf/2001.02613.pdf.

[20] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. 2017. URL https://dspace.mit.edu/handle/1721.1/115978.

[21] Charles Richter, John Ware, and Nicholas Roy. High-speed autonomous navigation of unknown environments using learned probabilities of collision. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6114–6121. IEEE, 2014. URL https://ieeexplore.ieee.org/abstract/document/6907760.

[22] Charles Richter, William Vega-Brown, and Nicholas Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Robotics Research*, pages 325–341. Springer, 2018. URL https://link.springer.com/chapter/10.1007/978-3-319-60916-4_19.

[23] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. URL http://proceedings.mlr.press/v15/ross11a.

[24] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018. URL https://www.annualreviews.org/doi/abs/10.1146/annurev-control-060117-105157.

[25] William R Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)*, 35(1):64–96, 2003. URL https://dl.acm.org/doi/abs/10.1145/641865.641868.

[26] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. Volumetric next-best-view planning for 3d object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11(10):159, 2014. URL https://journals.sagepub.com/doi/full/10.5772/58759.

[27] Jian Wang, Joseph Bartels, William Whittaker, Aswin C Sankaranarayanan, and Srinivasa G Narasimhan. Programmable triangulation light curtains. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. URL http://www.cs.cmu.edu/~ILIM/programmable_light_curtain/html/index.html.

[28] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5555–5564, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_Recurrent_Neural_Network_for_Un-Supervised_Learning_of_Monocular_Video_Visual_CVPR_2019_paper.pdf.

[29] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. URL https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Wu_3D_ShapeNets_A_2015_CVPR_paper.html.

[30] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/papers/Zhan_Unsupervised_Learning_of_CVPR_2018_paper.pdf.

[31] Haokui Zhang, Chunhua Shen, Ying Li, Yuanzhouhan Cao, Yu Liu, and Youliang Yan. Exploiting temporal consistency for real-time video depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1725–1734, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/papers/Zhang_Exploiting_Temporal_Consistency_for_Real-Time_Video_Depth_Estimation_ICCV_2019_paper.pdf.

[32] ChaoQiang Zhao, QiYu Sun, ChongZhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, pages 1–16, 2020. URL https://link.springer.com/article/10.1007/s11431-020-1582-8.

[33] Javad Zolfaghari Bengar, Abel Gonzalez-Garcia, Gabriel Villalonga, Bogdan Raducanu, Hamed H Aghdam, Mikhail Mozerov, Antonio M Lopez, and Joost van de Weijer. Temporal coherence for active learning in videos. *arXiv preprint arXiv:1908.11757*, 2019. URL https://synthia-dataset.net/.