Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Active Velocity Estimation using Light Curtains via Self-Supervised Multi-Armed Bandits

Siddharth Ancha* Gaurav Pathak† Ji Zhang‡ Srinivasa Narasimhan‡ David Held‡
*Massachusetts Institute of Technology, Cambridge, MA 02139, USA
†Adobe, San Jose, CA 95110, USA
‡Carnegie Mellon University, Pittsburgh, PA 15213, USA

Website: https://siddancha.github.io/projects/active-velocity-estimation[1]

***Abstract*—To navigate in an environment safely and autonomously, robots must accurately estimate where obstacles are and how they move. Instead of using expensive traditional 3D sensors, we explore the use of a much cheaper, faster, and higher resolution alternative: *programmable light curtains*. Light curtains are a controllable depth sensor that sense only along a surface that the user selects. We adapt a probabilistic method based on particle filters and occupancy grids to explicitly estimate the position and velocity of 3D points in the scene using partial measurements made by light curtains. The central challenge is to decide where to place the light curtain to accurately perform this task. We propose multiple curtain placement strategies guided by maximizing information gain and verifying predicted object locations. Then, we combine these strategies using an online learning framework. We propose a novel self-supervised reward function that evaluates the accuracy of current velocity estimates using future light curtain placements. We use a multi-armed bandit framework to intelligently switch between placement policies in real time, outperforming fixed policies. We develop a full-stack navigation system that uses position and velocity estimates from light curtains for downstream tasks such as localization, mapping, path-planning, and obstacle avoidance. This work paves the way for controllable light curtains to accurately, efficiently, and purposefully perceive and navigate complex and dynamic environments.**[1]

I. Introduction

Robots in the real world must navigate in the presence of moving objects like humans and vehicles whose motion is a priori unknown. This is a common challenge in many applications like autonomous driving, indoor and outdoor mobile robotics, and robot delivery. How should a robot sense and perceive such dynamic environments? How can it accurately estimate the motion of obstacles?

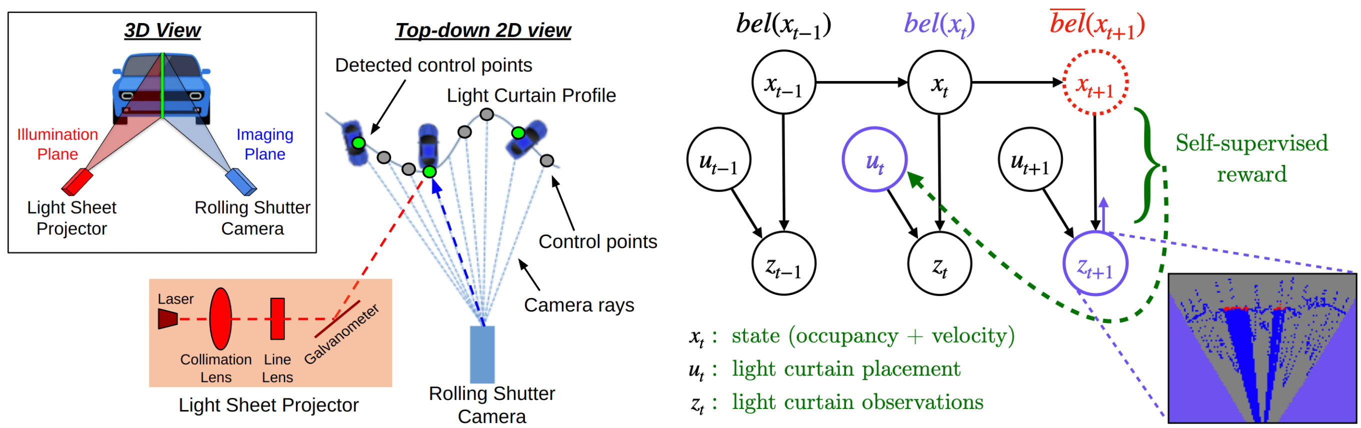
3D sensors such as LiDARs and depth cameras are conventionally used for robot navigation. However, LiDARs are typically expensive and low-resolution. Although cameras are cheaper and higher-resolution, depth estimates can be noisy and inaccurate. An alternative paradigm is *active perception* [7, 8] where a controllable sensor is actively guided to focus on only the relevant parts of the environment. Programmable light curtains [69, 9, 4, 58, 5] are a recently invented, lightweight 3D sensor that detects points intersecting any user-specified 2D surface ("curtain"). Light curtains combine the best of passive cameras (low cost, high resolution, and high speed) and LiDARs (accurate depth estimation along the curtain, robustness to bright lighting and scattered media like fog/smoke [69]). Compared to widely used commercial LiDARs like the Ouster OS1-128 [43], a lab-built light curtain prototype is relatively inexpensive (\$1,000 v.s. $\sim$\$20,000), higher vertical resolution (1280 rows/0.07° v.s. 128 rows/0.35°) and faster (45-60 Hz v.s. 10-20 Hz). See App. E for benefits of light curtains over conventional depth sensors. Because programmable light curtains are an active sensor, realizing these benefits requires actively deciding where to place the curtain at each timestep; this is the principal algorithmic challenge posed by programmable light curtains.

Previously, light curtains have been used for object detection [4], depth estimation [58], and estimating safety regions [5]. However, light curtains have not been used to explicitly estimate velocities of dynamic objects. Velocity estimation is crucial for many tasks in robotics such as trajectory forecasting, obstacle avoidance, motion planning, and dynamic object removal for SLAM [65].

The focus of this paper is to develop light curtain placement strategies that improve velocity estimates. We use dynamic occupancy grids [23] to estimate velocities and occupancies from points detected by light curtains without requiring point cloud segmentation or explicit data association across frames. First, we extend light curtain placement strategies from previous works [4, 5] to integrate dynamic occupancy grids. Then, we propose a novel method to switch between multiple light curtain placement strategies using a multi-armed bandits approach. The feedback for the multi-armed bandits is obtained using a novel *self-supervised* reward function that evaluates the current estimates of occupancy and velocity using future light curtain placements, without requiring ground truth or additional sensors. We obtain this supervision by reusing intermediate quantities computed during recursive Bayes estimation of dynamic occupancy grids; thus the self-supervised rewards do not require extra light curtain placements or additional computations. We evaluate our approach on challenging simulated and real-world environments with complex and fast object motion. We integrate our method into a full-stack navigation pipeline and show that

[1]Please see our project website for (1) the appendix, (2) an overview video, (3) videos showing qualitative results of our method, and (4) source code.

*Corresponding author. E-mail: sancha@mit.edu.
*†This work was performed when SA and GP were affiliated with CMU.



(a) Light curtain working principle

(b) Bayes filter with self-supervised reward

Figure 1: **(a)** Illustration of programmable light curtains adapted from [4]. An illumination plane (from the projector) and an imaging plane (of the camera) intersect to produce a light curtain. A controllable galvanometer mirror rotates synchronously with the camera's rolling shutter and images the points of intersection. See Sec. II-A for more details. **(b)** A *Dynamic Bayes network* [65] for controllable sensing. At timestep t, x_t corresponds to the state of the world, u_t corresponds to the action i.e. the location of light curtain placement, z_t corresponds to light curtain measurements, and $\overline{bel}(x_t)$ and $bel(x_t)$ are the inferred distributions over states before and after incorporating measurements z_t, respectively. This is a slightly modified graphical model for controllable sensing where actions u_t don't affect state x_t but directly affect observations z_t.

the multi-armed bandits approach is able to outperform each individual strategy.

Our contributions include:

1) We re-derive the dynamic occupancy grid method [23] using a more rigorous mathematical analysis grounded in Bayesian filtering [65] (Sec. IV, App. B).
2) We design curtain placement strategies for dynamic occupancy grids to verify predicted object locations (Sec. V-A) and maximize information gain in hybrid discrete-continuous spaces (Sec. V-B, App. D).
3) We propose a novel *self-supervised* reward function that evaluates current velocity estimates using future light curtain placements without requiring additional supervision. Using the self-supervised reward, we learn to combine multiple curtain placement policies using a multi-armed bandit framework (Sec. VI).
4) We evaluate this approach in simulated and real-world environments with fast-moving obstacles and demonstrate that it outperforms individual placement strategies (Sec. VIII).
5) We develop an efficient and parallelized pipeline where light curtain sensing, grid estimation and computing curtain placement are tightly coupled and continuously interact with each other at ∼45 Hz (Sec. VII, Fig. 4, App. F).
6) We integrate our method into a full-stack navigation pipeline that uses position and velocity estimates to perform localization, mapping and obstacle avoidance in real-world dynamic environments (App. K).

II. Background

A. Light curtain working principle

Programmable *light curtains* [69, 9, 4, 58, 5] are a recently developed controllable depth sensor that image any user-specified vertically-ruled 2D surface in the environment. The device contains two main components: a rolling-shutter camera and a rotating light sheet laser (see illustration in Fig. 1a). The camera activates one pixel column at a time, from left to right, via the rolling shutter. We refer to the top-down projection of the imaging plane corresponding to each pixel column as a "camera ray" (shown in Fig. 1a). The shape of the light curtain is entirely specified by a 2D *control point* selected on each camera ray (shown as gray and green circles). The set of control points forms the input to the light curtain device. The laser is vertically aligned and synchronized with the camera's rolling shutter. A controllable galvo-mirror rotates the light sheet to point it at the control point corresponding to the currently active pixel column. Triangulated 3D scene points that both (1) intersect the laser light sheet and (2) are visible in the currently active pixel column, get detected by the device. If there exists an object in the environment at the surface of this intersection, then the point will have a large intensity in the camera reading; otherwise it will not; thus the device outputs the subset of control points (shown as green circles in Fig. 1a) that correspond to 3D object surfaces. Importantly, light curtains form a partial observation on the scene, since only control points can be detected. Please see [9, 69] for further details on the mechanism behind a programmable light curtain.

B. Bayes filtering

This section provides a brief background on Bayes filtering and introduces notation used throughout the paper. A *dynamic Bayes filter* [65], also known as a *hidden Markov model* or a *state space model* is represented by a probabilistic graphical model shown in Fig. 1b. The state of the world at timestep

t is denoted by x_t (in our case, x_t is the occupancy and velocity of a set of cells arranged in a 2D grid from the top-down view; more details in App. B-A). The control actions are denoted by u_t (the locations where the light curtain is placed). Observations obtained from the sensor are denoted by z_t. Fig. 1b is a slight modification of the standard model for the task of active perception, where actions don't affect the state of the world x_t but directly affect the observations z_t.

The goal is to infer at each timestep t the posterior distribution (a.k.a "belief") $bel(x_t) = P(x_t \mid u_{1:t}, z_{1:t})$ over the current state x_t from the sequence of sensor observations $z_{1:t}$ and the known sequence of actions $u_{1:t}$. This is computed using *recursive Bayesian estimation* [65] that alternates between two steps.

$$\overline{bel}(x_t) = \int_{x_{t-1}} bel(x_{t-1})\ P(x_t \mid x_{t-1})\ \mathrm{d}x_{t-1} \tag{1}$$

$$bel(x_t) \propto P(z_t \mid x_t, u_t)\ \overline{bel}(x_t) \tag{2}$$

First, the *motion update* step computes an intermediate prior belief $\overline{bel}(x_t)$ by applying a motion model $P(x_t \mid x_{t-1})$ that encodes the dynamics of the environment. Then, the *measurement update* step computes the updated posterior belief $bel(x_t)$ by incorporating sensor observations from the current timestep. To make this paper self-contained, we provide a detailed mathematical derivation of these steps in App. A.

III. Related Work

A. Active perception and light curtains

Active perception involves actively controlling a sensor such as camera parameters [7], moving a camera to look around occlusions [19], and next-best view planning [21] for object instance classification [76, 28, 26, 60] and 3D reconstruction [44, 47, 67, 24]. Programmable light curtains [69, 9, 16] are a controllable depth sensor that have been used for active perception tasks such as active object detection [4], active depth estimation [58], and actively estimating safety regions [5]. However, most prior light curtain work has only focused on estimating object positions. They either place curtains with fixed scan patterns [69, 16] or adaptive curtains for static scenes without taking object motion into account [9, 4, 58]. Ancha et al. [5] track safety regions by learning to forecast future locations; this could be interpreted as *implicit* velocity estimation. However, we are the first to explicitly estimate obstacle velocities which can be used for other downstream tasks like trajectory forecasting, obstacle avoidance and motion planning. Furthermore, we combine multiple adaptive strategies like random curtains [9, 5], maximizing information gain [4], and verifying predicted object locations [5] using a novel multi-armed bandit framework for estimating both object positions and velocities.

B. Velocity estimation from point clouds

Prior works on estimating *scene flow* [68, 70, 49, 32, 63] compute correspondences between point clouds acquired at consecutive timesteps; velocities can then be extracted from these correspondences. Furthermore, self-supervised approaches [54, 75, 45, 48, 10, 35] can learn to estimate scene flow without requiring ground truth annotations. However, these methods are designed to compute flow between *complete* scans of the environment, such as those obtained from a LiDAR sensor, where correspondences exist for most points. In contrast, a single light curtain measurement is a *partial* point cloud – a subset of visible points that intersect the curtain. Depending on where they are placed, consecutive light curtains may not contain any correspondences at all. Therefore, scene flow methods are not suited for point clouds acquired by light curtains.

Another approach is to first segment the point cloud into a collection of separate objects [38, 27, 46, 64, 78, 41], track each object, and finally register each object's segmented point cloud across frames using either optimization-based [11, 59, 36, 79, 77, 51], probabilistic [37, 39, 34, 2], or learning-based [71, 72, 6, 20] methods. However, errors in point cloud segmentation can lead to incorrect velocity estimates. Instead, our method uses particle-based occupancy grids and avoids the need to perform either segmentation or explicit data association across frames.

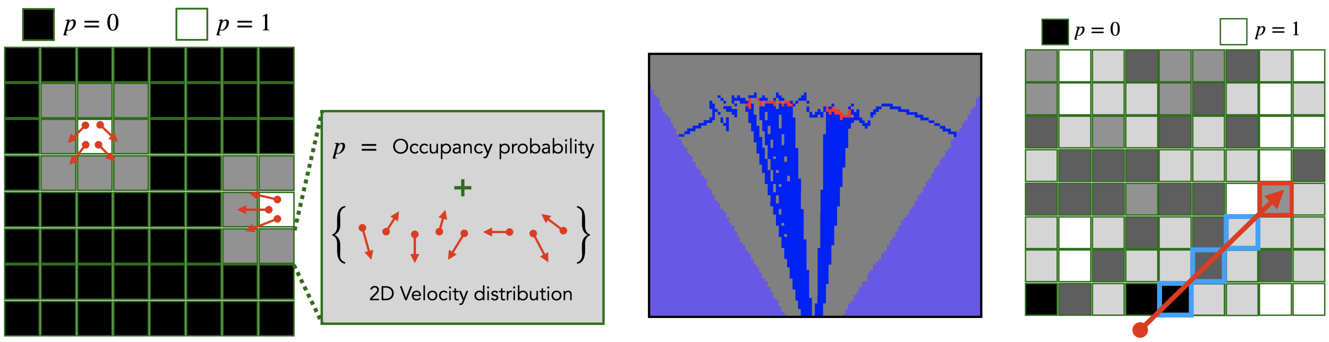
C. Self-tuning Bayes filters

Prior works have used *innovation* i.e. the difference between predicted and observed measurements of a Kalman filter, to "self-tune" model parameters without needing ground truth annotations. Earlier works use an autoregressive moving average innovation model (ARMA) [33, 55, 30, 25, 80]. More recent works use the *normalized innovation squared (NIS)* metric to optimize Kalman filter noise models using downhill simplex methods [57], Bayesian optimization [17, 18], and evolutionary algorithms [56, 12]. Our self-supervised metric is inspired by Kalman filter innovation, but is used to select a sensor control strategy at each timestep using multi-armed bandits rather than tuning noise models.

IV. Dynamic occupancy grids

We now describe how we apply *dynamic occupancy grids* [23] for velocity estimation with light curtains. A dynamic occupancy grid is a Bayes filter that combines two conventional representations in robotics: occupancy grids and particle filters. Occupancy grids [29, 65] are a standard tool for mapping the location of static objects in the environment from the 2D top-down view. Each cell in the grid contains an *occupancy probability* $p \in [0, 1]$, denoting the probability of the cell being occupied by an object. *Dynamic* occupancy grids [23] are an extension of classical occupancy grids (see Fig. 2a). Each cell in the grid contains both the occupancy probability p as well as a probability distribution over 2D velocities. The velocity distribution is represented by a set of weighted particles, where each particles stores a single 2D velocity. The set of weighted particles approximates the true velocity distribution.

While Danescu et. al. [23] showed that dynamic occupancy grids can accurately estimate occupancies and velocities, the precise role of particles and what they represent remained



(a) Dynamic occupancy grid: each cell contains occupancy and velocity distributions

(b) Raycasting to compute freespace

(c) Raymarching to compute depth probs.

Figure 2: **(a)** *Dynamic occupancy grid.* The 2D grid represents the top-down view. Like conventional occupancy grids [29, 65], each cell contains an occupancy probability $p \in [0, 1]$. In addition, each cell also contains a set of weighted *particles* where each particles stores a single 2D velocity. The set of particles together represents a probability distribution of that cell's velocity. **(b)** *Ray-casting* to light curtain detections to extract freespace information. Red cells contain detected points and are marked occupied. Blue cells are freespace; they either lie undetected on the light curtain or lie on rays cast from the sensor to the red cells. Gray denotes unknown occupancy. Purple cells are outside the light curtain's field of view. **(c)** *Ray-marching* to compute the depth probabilities of cells along a camera ray. The depth probability of the red cell is the product of the probability that the red cell is occupied and the probabilities of each blue cell being unoccupied.

unclear. Particles were described as representing a cell's velocity distribution; however, the movement of particles from one cell to another is somewhat inconsistent with this interpretation. Elsewhere, particles are described as being "physical building blocks of the world", i.e. parts of objects that can move; however, under this interpretation, it is unclear what distribution a set of particles is supposed to represent, since each particle represents a different part of an object. Furthermore, the particles were not only used to represent velocities, but their count inside a cell was proportional to the occupancy probability. In this work, we re-derive dynamic occupancy grids using a more rigorous mathematical analysis found in App. B, in which we explicitly state the assumptions made and provide a precise, mathematically rigorous interpretation of particles.

Motion and measurement updates: In the motion update step, particles are resampled from each cell in the grid and moved to another cell based on their velocities and the motion model. We assume access to a depth sensor (e.g. light curtains, LiDAR, depth cameras) that measures depth but does not directly measure velocity. In the measurement update step, the sensor provides (noisy) observations of occupancy for a subset of un-occluded cells in the grid. These observations are used to update the occupancy probabilities; velocities are inferred indirectly in the motion update step that are consistent with observed occupancies. This method is able to estimate velocities from depth measurements alone without requiring explicit data association across frames.

Raycasting to extract freespace information: As explained in Section II-A, a light curtain only returns whether there is a 3D object surface at the location of the control points where the camera rays and the laser sheets intersect; no depth information is returned for other locations in the environment. Fig. 2b shows an observation grid from a light curtain placement where cells directly measured to be occupied are shown in red and free cells are shown in blue. From this figure, we see that all voxels in between the light curtain source and a detected point must be unoccupied. Since 3D points were detected in the occupied cells, light must have traveled along these rays without obstruction; we mark cells along these rays to be free (shown in blue in Fig. 2b). To take advantage of this information, we cast rays using an efficient voxel traversal algorithm [3, 42] from the sensor to occupied cells (shown in red). More details can be found in App. B-C. Thus by exploiting visibility constraints, we are able to extract more information from the light curtain.

V. Curtain placement strategies

Using dynamic occupancy grids and Bayesian filtering, we have a method to infer occupancies and velocities explicitly from light curtain measurements (details in App. B). The main challenge that we address in this paper is to compute the best curtain placement from the dynamic occupancy grid i.e. from the current estimates of occupancy and velocity. The measurements from the placed curtain will be input back to upgrade the grid, closing the loop.

In order to compute the best curtain placement, we must first predict the occupancy when the next light curtain will be placed. To do so, we forecast the current dynamic occupancy grid, using the currently estimated velocities, to the next timestep via the motion update step (Eqn. 1, Eqn. 4 in App. B-B). In this section, we propose various curtain placement strategies computed from the forecasted grid. In Sec. VI, we will propose a novel method to combine them and outperform each individual strategy.

A. Maximizing depth probability

Strategy 1: Depth Probability: Following Ancha et al. [5], Strategy 1 places curtains at the highest probability object

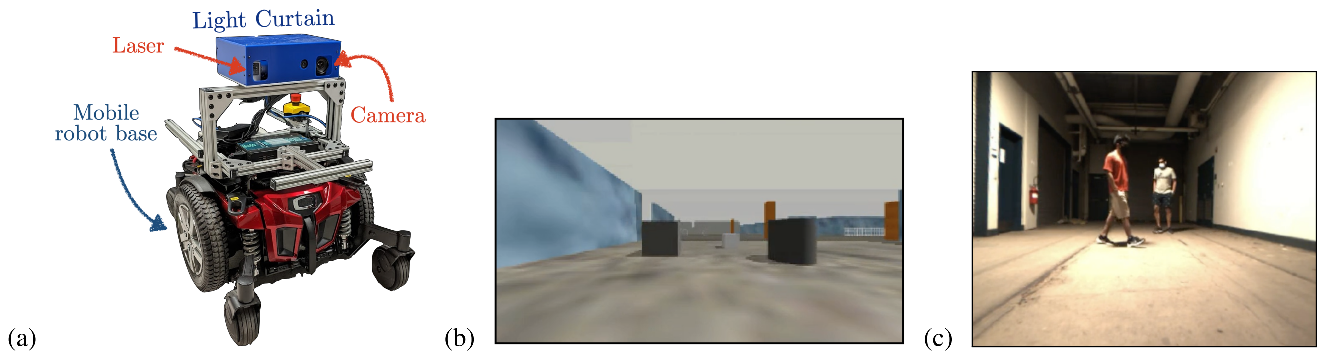


Figure 3: **(a)** *Mobile light curtain robot platform:* A light curtain device (in blue) is mounted on top of a mobile robot. We use this setup to perform real-world experiments. **(b)** *Simulated environment:* consists of differently shaped objects (cuboids and cylinders) moving in (1) linear oscillatory/harmonic motion along various directions, (2) curved sinusoidal motion, and (3) random Brownian motion. **(c)** *Real-world environment:* consists of two pedestrians walking in front of the sensor in multiple directions, at different speeds and in complex trajectories.

locations. This strategy is motivated by the fact that a light curtain only senses visible object surfaces when it intersects them. Therefore, this approach can be used to verify whether objects are indeed located at the forecasted object locations.

Since occupancy grids are probabilistic, this strategy places curtains at locations of highest "depth probability", which is the probability that a control point at a given cell would return a depth reading. The depth probability of a cell is the probability that the cell is occupied, and all occluding cells (lying on the ray starting from the sensor and ending at the target cell) are free (see Fig. 2c). We borrow the idea of "ray marching" from the literature on volumetric rendering [66, 53] to compute depth probabilities efficiently; see App. C for more details on the algorithm and computational complexity. For each camera ray, we place the curtain on the cell with the maximum depth probability.

B. Maximizing information gain

Another placement strategy that was found useful in previous work on 3D object detection [4] was to place curtains at the regions of highest "uncertainty". This is based on the principle of maximizing *information gain* for active sensing.

Recall the dynamic Bayes network in Fig. 1b. Given a forecasted prior belief $P(x_t) = \overline{bel}(x_t)$, the information gain framework prescribes that the action u_t should be taken that maximizes the information gain $\mathrm{IG}(x_t, z_t \mid u_t)$ between the state x_t and the observations z_t when using u_t. Information gain, which is a well-studied quantity in information theory, is the expected reduction in entropy (i.e. uncertainty) before and after sensing: $\mathrm{H}(P(x_t)) - \mathbb{E}_{z_t|u_t}\big[\mathrm{H}(P(x_t \mid z_t, u_t))\big]$.

While information gain for conventional occupancy grids is straightforward to derive [4], it is not so for the case of dynamic occupancy grids. This is because the underlying state space of dynamic occupancy grids is a 'mixture' of discrete and continuous spaces – a cell can either be unoccupied or occupied with a continuous velocity. Unfortunately, the entropy of such mixed discrete-continuous spaces is not well-defined [31]. We overcome this problem using a more general definition of information gain based on the "Radon–Nikodym" derivative [31] that doesn't require explicitly calculating the entropy. In App. D, we show that the formula for information gain for dynamic occupancy grids (under certain assumptions) turns out to equal the occupancy uncertainty, described next.

Strategy 2: Occupancy Uncertainty: Let ω_t^i be the occupancy probability estimated for the i-th cell at the t-th timestep. Then, the information gain is the sum of binary cross entropies $\mathrm{H}_{\text{occ}}(\omega_t^i) = -\omega_t^i \ \log_2 \omega_t^i - (1-\omega_t^i) \ \log_2(1-\omega_t^i)$ of the cells that the curtain lies on. Intuitively, since measurements from a depth sensor only provide information about occupancy and not velocity, the overall information gain is equal to the total occupancy uncertainty. A similar information gain computation was used in Ancha et al. [4] for static occupancy grids; in App. D we prove that the formula for information gain is the same as total occupancy uncertainty even for the more complex case of mixed discrete-continuous distributions. Strategy 2 places a curtain that maximizes the occupancy uncertainty.

Strategy 3: Velocity Uncertainty: Each cell also contains a velocity distribution $V_t^i = \{(v_t^{i,m}, p_t^{i,m}) \mid 1 \leq m \leq M\}$ represented by a set of M weighted particles with velocities $v_t^{i,m}$ and weights $p_t^{i,m}$ that sum to 1. In this strategy, we maximize the sum of velocity entropies. The discrete set of particles is used to approximate what is inherently a continuous velocity distribution. Therefore, we must compute the *differential* entropy of the continuous velocity distribution by first estimating its probability density function. We fit a multivariate Gaussian distribution to the set of weighted particles with mean $\mu = \sum_{m=1}^M p_t^{i,m} v_t^{i,m}$ and covariance matrix $\Sigma = \sum_{m=1}^M p_t^{i,m} (v_t^{i,m} - \mu)(v_t^{i,m} - \mu)^T$. Then, we compute the differential entropy of the fitted Gaussian: $\mathrm{H}_{\text{vel}}(V_t^i) = \frac{1}{2} \log\det(2\pi e \Sigma)$. One could alternatively use other families of continuous distributions, such as kernel density estimators. Finally, we place a curtain that maximizes the sum of velocity entropies H_{vel} of the cells the curtain lies on.

Strategy 4: Combined Uncertainty: In this strategy, we maximize a weighted combination of occupancy and velocity entropies: $\mathrm{H}_{\text{cmb}}(\omega_t^i, V_t^i) = \mathrm{H}_{\text{occ}}(\omega_t^i) + \omega_t^i \ \mathrm{H}_{\text{vel}}(V_t^i)$. The velocity uncertainty is weighted by the occupancy probability. This captures the notion that if the occupancy probability is very low, then the overall uncertainty should also be low even if the velocity uncertainty is high, because the velocity uncertainty is not relevant if the cell is unoccupied. This is a heuristic curtain placement policy that performs well in practice.

VI. Self-supervised multi-armed bandits

Can we combine the various curtain placement strategies developed in Sec. V to improve performance? In this section, we develop a multi-armed bandit method to do so enabled by a novel *self-supervised* reward function.

A. Multi-armed bandit framework

A multi-armed bandit [61] is an online learning framework consisting of a set of actions or "arms", where each action is associated with an unknown reward function. The agent only observe samples from the reward distribution when it takes that action. The goal is to maximize the cumulative reward over time. The agent maintains a running average of the rewards for each action, called *Q-values*. We use ϵ-greedy multi-armed bandits [61], that trades-off exploration with exploitation. With probability ϵ, the bandit performs exploration and chooses an action at random. With probability $1-\epsilon$, it performs exploitation and chooses the action that has the highest Q-value. We use multi-armed bandits to intelligently switch between the four curtain placement strategies at test time.

B. Self-supervised rewards

The bandit framework requires a reward function to evaluate actions. Our eventual goal is to accurately estimate occupancy and velocity. How can we design a function that rewards improvements in occupancy and velocity estimates, but can also be computed at test-time using only light curtain placements and measurements? This is challenging because light curtains cannot directly measure velocities; they can only measure the occupancies of a small set of locations where they are placed.

Let us revisit the dynamic Bayes network from Sec. II-B, shown in Fig. 1b. Belief distributions are represented by dynamic occupancy grids. At timestep $t-1$, the grid representing the belief $bel(x_{t-1})$ was forecasted by applying the motion model to obtain the prior belief $\overline{bel}(x_t)$ at timestep t (Eqn. 1, Eqn. 4 in App. B). Then, in the measurement update step, the current light curtain measurement z_t obtained by placing a curtain at locations u_t is used to update the grid to $bel(x_t)$ (Eqn. 2, Eqn. 5 in App. B). Therefore, *we attribute the accuracy of $bel(x_t)$ to action u_t*.

The *forecasted occupancy* at time $t+1$ is computed by using the current velocity estimates to forecast the current occupancy by an interval Δt using the motion update step (Eqn. 1, Eqn. 4 in App. B). The *forecasted occupancy* will be accurate if both the current velocities and current occupancies are accurate. Therefore, the accuracy of forecasted occupancy acts as an appropriate reward function that captures both occupancy and velocity accuracies.

How do we evaluate forecasted occupancy computed using $bel(x_t)$, without requiring ground truth, in a self-supervised way? This is possible by reusing intermediate quantities output during recursive Bayesian updates.

First, note that the forecasted occupancy of $bel(x_t)$ is $\overline{bel}(x_{t+1})$ computed by the next motion update step. *Our main insight is that before applying the next measurement update step, $\overline{bel}(x_{t+1})$ can be evaluated using the partial occupancy observed by the next light curtain measurements z_{t+1}*. We use the F_1-score between the forecasted occupancy grid and the partially observed occupancy grid *as a self-supervised reward for the previous light curtain placement u_t* (See Fig. 1b). Specifically, we compute the self-supervised reward $R_t = F_1(\overline{bel}(x_{t+1}), z_{t+1})$, where $\overline{bel}(x_{t+1})$ is computed using Eqn. 1 (more specifically, Eqn. 4 in App. B), and z_{t+1} is the partial occupancy observed at time $t+1$. See App. G for details on the F_1-score.

An advantage of our self-supervised reward is that it does not require any extra computation. This is because (1) occupancy forecasting of $bel(x_t)$ is performed anyway as part of the motion update step, and (2) the partial occupancy information from z_{t+1} is computed anyway in the next measurement update step. By reusing quantities already computed during recursive Bayes filtering, our self-supervised reward does not require any extra forecasting steps nor any extra light curtain placements.

At each timestep, we use the ϵ-greedy strategy to select one among the four curtain placement strategies $a \in \{a_1, a_2, a_3, a_4\}$. Then we compute the curtain placement u_t according to strategy a. When the accuracy of the forecasted occupancy R_t is obtained in the next timestep, we update the Q-value of a as $Q(a) \coloneqq Q(a) + \alpha\,[R_t - Q(a)]$. We use the *non-stationary reward* formulation [61] of multi-armed bandits with smoothing parameter α to account for the possibility that different strategies $\{a_1, a_2, a_3, a_4\}$ may be superior at different times. See App. I for more details.

VII. Parallelized pipeline

Fig. 4 shows our pipeline that has three processes: (1) light curtain sensing, (2) Bayes filtering using dynamic occupancy grids, and (3) computing curtain placement. The processes are run in parallel threads with shared memory, at their own independent speeds.

1. Light curtain imaging: This thread continuously places curtains at locations determined by one of the four strategies described in Sec. V. However, when waiting for the next curtain placement to be computed, it places *random* curtains [5] (that are generated offline) to sense random locations in the scene. This ensures that the device is always kept busy and runs at approximately 45 Hz.

2. Bayes filtering: This thread inputs light curtain measurements and updates the dynamic occupancy grid. It alternates between motion and measurement update steps (Eqns. 1, 2, Eqns. 4, 5 in App. B). The motion update step requires two grids, each representing the current and next timesteps. Particles are sampled from the *current* grid, perturbed according to the motion model, and inserted into the *next* grid. The roles of the

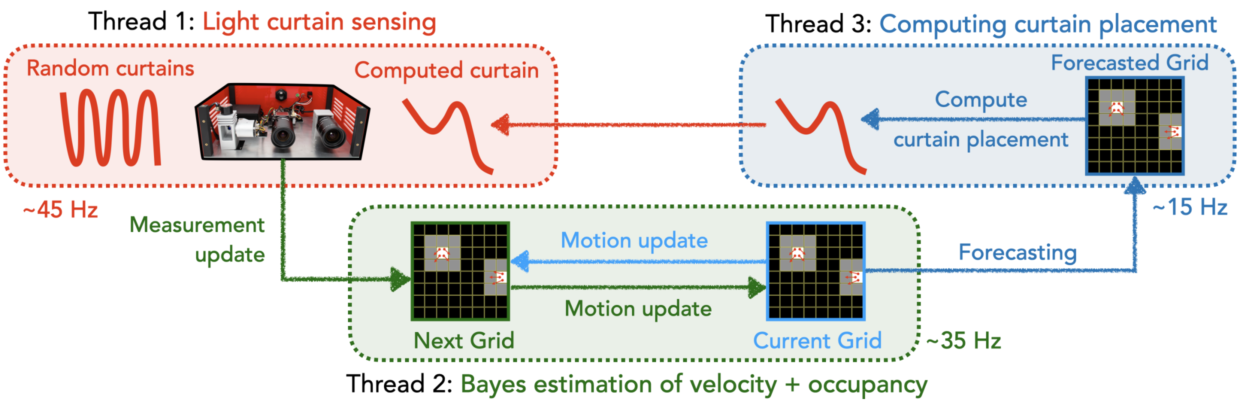


Figure 4: *Implementation of our method as a parallelized pipeline.* Our methods contains three components: (1) light curtain sensing, (2) Bayes estimation of dynamic occupancy grids, and (3) computing curtain placement. Each process can be run in parallel in a separate thread at its own independent speed. The three processes are tightly coupled in a closed loop using three grids as shared memory. Our implementation ensures that information flows between the threads safely and continuously.

two grids are swapped at every successive motion update to avoid copying data. This thread runs at approximately 35 Hz.

3. Computing curtain placement: This uses the most recent dynamic occupancy grid to compute the next curtain placement (Sec. V). It first *forecasts* the grid, using the same motion update step (Eqn. 1, Eqn. 4 in App. B), to the next timestep when the next curtain is expected to be imaged. The forecasted occupancy is used to compute the curtain placement. In App. F, we describe how an extra grid is used to ensure thread-safety and that no thread ever needs to wait on another to finish processing. Finally, the control points of the computed curtain are sent to the light curtain device. The three inter-dependent processes are tightly coupled and continuously interact with each other.

VIII. Experiments

A. Environments

Simulation environment: We use a simulated environment consisting of various blocks moving in a variety of motions (see Fig. 3b). The environment contains cylinders and cuboids, moving in (1) linear, harmonic (oscillatory) motion along different directions, (2) curved sinusoidal motion, and (3) random Brownian motion. We use an efficient light curtain simulator described in App. J.

Real-world environment: Our real-world environment consists of a mobile robot with a mounted light curtain device (Fig. 3a) navigating in the presence of two pedestrians walking in multiple directions, at different speeds and in complicated trajectories (see Fig. 3c).

B. Evaluation metrics

Since we wish to evaluate the accuracy of both occupancy and velocity estimates, we use the *forecasted occupancy* [50, 1] as our evaluation metric. As noted in Sec. VI-B, the forecasted occupancy will be accurate if both current velocities and current occupancies are accurate. The future occupancy at time $t+\Delta t$ is computed by the motion update step (Eqn. 1, Eqn. 4 in App. B) that uses the current velocity to forecast the current occupancy by an interval Δt. This metric is particularly relevant for obstacle avoidance where estimates of future obstacle locations must be accurately computed to plan safe, collision-free paths.

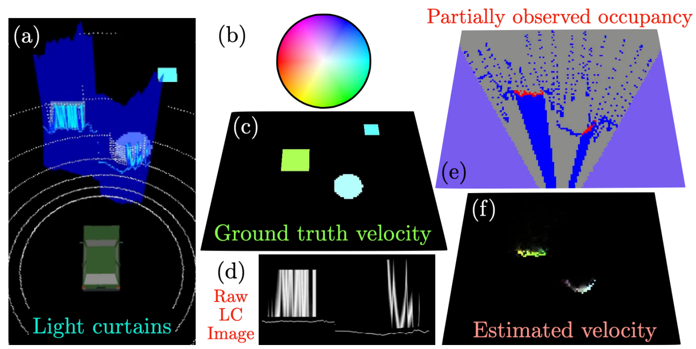


Figure 5: *Velocity estimation in simulation.* (a) The environment with moving blocks and curtain placement shown in blue. (b) Color-coding for visualizing velocity from the top-down view. (c) Ground truth occupancy and velocity. (d) Raw light curtain images; high intensities are good because it means that object surfaces were found and intersected by the light curtain. (e) Partial occupancy observations from light curtain measurements that are input to the dynamic occupancy grid. (f) Velocity and occupancy estimated by the dynamic occupancy grid. We advise the reader to view video examples on the project website.

Ideally, the accuracy of forecasted occupancy can be computed by comparing it against ground truth occupancy at $t + \Delta t$. This is possible in simulated environments where ground truth occupancy is available for all grid cells. In real-world environments, true occupancy can only be measured for a subset of cells by the light curtain; in this case, we use the "self-supervised" version of the metric described in Sec. VI-B. We follow prior works [40, 52, 62] that treat the evaluation of occupancy as a classification problem and compute several metrics: (1) classification accuracy [40, 52], (2) precision, (3)

	Simulated environment					*Real environment*				
	Classification accuracy ↑	Precision ↑	Recall ↑	F_1-score ↑	IOU ↑	Classification accuracy ↑	Precision ↑	Recall ↑	F_1-score ↑	IOU ↑
Simulated LiDAR	0.9584	0.2498	0.1073	0.1360	0.0787	*NA*				
Random curtains only	0.9662	0.2698	0.0567	0.0850	0.0468	0.9852	0.6306	0.2357	0.2405	0.2136
Max. depth probability	0.9610	0.2448	0.1146	0.1388	0.0792	0.9832	0.5943	0.2727	0.3047	0.2353
Max. occupancy uncertainty	0.9609	0.2717	0.1266	0.1493	0.0857	0.9811	0.5733	0.3041	0.3319	0.2515
Max. velocity uncertainty	**0.9648**	0.2728	0.0581	0.0838	0.0458	**0.9864**	0.6221	0.2615	0.2545	0.2232
Max. occupancy + velocity uncertainty	0.9629	**0.3026**	0.1251	0.1544	0.0895	0.9822	0.5899	0.3175	0.3421	0.2727
Multi-armed bandits (**Ours**)	0.9623	0.2814	**0.1402**	**0.1690**	**0.0976**	0.9854	**0.6467**	**0.3647**	**0.3703**	**0.3053**

Table I: Accuracy of occupancy and velocity estimation measured using *forecasted occupancy* in (a) simulated, and (b) real environments.

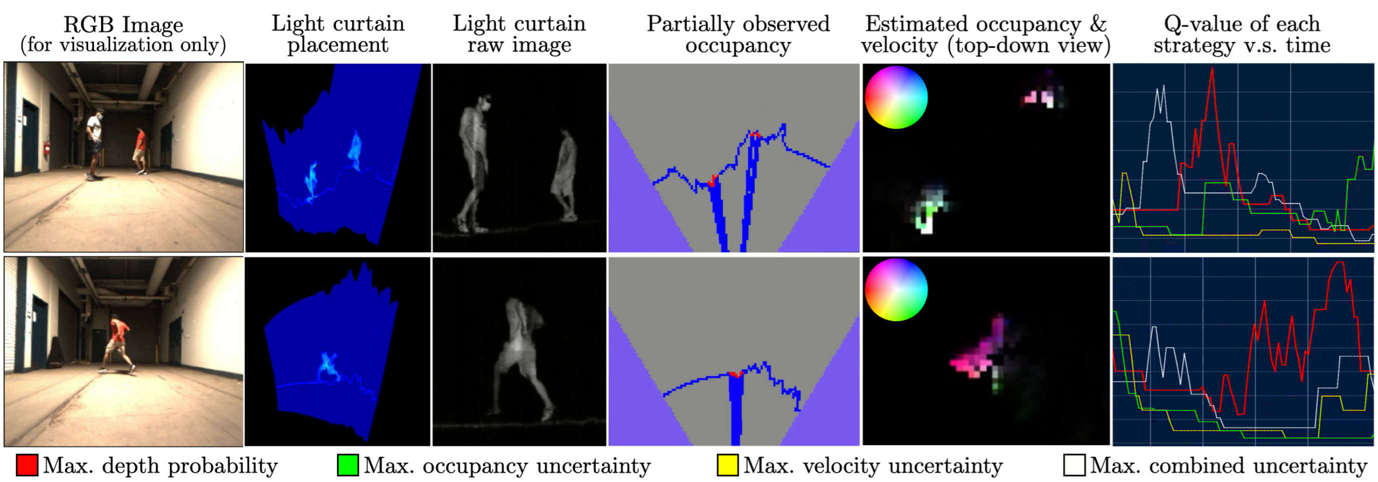


Figure 6: *Velocity estimation in the real world using multi-armed bandits (MAB)*. Please refer to titles and Fig. 5 for descriptions of each column. Our method only uses light curtains; RGB images are for visualization only. The rightmost column shows the Q-values of each strategy. Higher Q-value is better; the action with the highest Q-value is chosen during exploitation. *Top row:* shows two pedestrians walking at relaxed speeds. The directions of motion are correctly inferred for each person: the pedestrian walking to the right is shown in greenish-blue and the person walking to the left is colored in red. The current action selected maximizes occupancy uncertainty. The bottom row shows a more challenging environment where where a lone pedestrian performs fast motion: running and jumping. The direction of velocity is correctly inferred as moving top-left (left and away from the sensor) i.e. reddish pink. The color saturation is high indicating the larger magnitude of velocity. The current action selected maximizes depth probability. We advise the reader to view the video examples on the project website.

recall, (4) F_1-score and (5) the IoU [62] between the predicted and ground truth occupancy masks. For more details on these metrics, please see App. G.

C. Quantitative analysis

Table I shows the performance of various light curtain placement strategies in simulated and real-world environments, evaluated using multiple forecasted occupancy metrics (see Sec. VIII-B, App. G). Since a large proportion of cells are unoccupied, the classification accuracy of all methods is very similar. Furthermore, precision and recall metrics can be deceived by mostly predicting negative and positive labels respectively. However, the F_1-score and IoU metrics are discriminative and robust; they are high only when both precision and recall are high. Therefore, we focus on these two metrics (shown in blue). In both sets of experiments, multi-armed bandits that combine the four curtain placement policies using our self-supervised reward outperform all other methods. This shows that intelligently switching between multiple placement strategies is more beneficial than using any one single strategy at all times.

Between the other four strategies, maximizing occupancy

	Frequency of selection	Avg. Q-value function (IOU)
Max. depth probability	22.9%	0.261
Max. occupancy uncertainty	31.1%	0.276
Max. velocity uncertainty	13.4%	0.202
Max. occupancy + velocity uncertainty	32.5%	0.292

Table II: Quantitative analysis of the multi-armed bandit method. The first column shows the percentage of times each action (i.e. curtain placement policy) was chosen. The second column shows the average Q-value of each action computed by the multi-armed bandit. Higher Q-value is better; the action with the highest value is selected during exploitation.

uncertainty and maximizing a linear combination of occupancy and velocity uncertainty perform comparably. Maximizing velocity uncertainty tends to perform the worst. Fortunately, multi-armed bandits learn to downweight this under-performing strategy (see Table II, rightmost column in Fig. 6). We also compare against other baselines: using only random curtains (without placing any computed curtains), and with a simulated LiDAR. Unsurprisingly, using random curtains performs the worst. All non-random curtain policies except maximizing velocity uncertainty are able to outperform LiDAR. This is because light curtains are faster (∼45 Hz) and can be placed intelligently to maximize the accuracy of occupancy and velocity estimates.

Table II shows an analysis specific to the multi-armed bandit method. Please see the caption for details. We find that the best performing policies in Table I have the highest Q-values and are selected most frequently. The following trend holds: the better the performance of an individual policy when used in isolation (shown in Table I), the higher is its average Q-value and its frequency of being chosen. However, a combination of all policies (MAB) is better than any single one.

D. Qualitative analysis

Visualizing velocities and occupancies: We use the HSV colorwheel [73] shown in Fig. 5 and 6, to jointly visualize velocities and occupancies. The color 'value' (from HSV) encodes the occupancy probability; dark is low occupancy probability and bright is high occupancy probability. The 'hue' encodes the direction of velocity from the top-down view. 'Saturation' encodes the magnitude of velocity: white is stationary whereas colorful corresponds to high speed. See App. H for more details.

Examples. Fig. 5 shows an example of velocity estimation in the simulated environment and Fig. 6 shows qualitative results on the real-world environment using our multi-armed bandits (MAB) curtain placement method. Please see captions for explanation. We advise the reader to view the video examples on the project website. In Fig. 5, we see that the estimated velocities appear to be consistent with the ground truth, as shown by the corresponding colors that indicate the estimated and ground-truth velocity directions.

Full-stack navigation. We integrate our system into a full-stack navigation pipeline [15] that performs planning, control and obstacle avoidance. We mount the light curtain device on a mobile robot (see Fig. 3a). We use ORB-SLAM3 [13] for localization and mapping that takes depth from light curtains as input. Using position and velocity estimates, the robot is able to perform dense mapping in an indoor environment and avoids static and dynamic obstacles. Please see App. K for more details.

IX. Conclusion

In this work, we develop a method using programmable light curtains, an actively controllable resource-efficient sensor, to estimate the positions and velocities of objects in complex, dynamic scenes. We use a probabilistic framework based on particle filters and occupancy grids to estimate velocities from partial light curtain measurements. We design curtain placement policies that verify predicted object locations and maximize information gain. Importantly, we combine the strengths of these policies using a novel multi-armed bandits framework that switches between the placement strategies to improve performance. This is enabled by our novel self-supervised reward function that evaluates current velocity estimates using future light curtain placements with only minimal computational overhead. We integrate our method into a full-stack navigation system that performs localization, mapping and obstacle avoidance using light curtains. We hope our work paves the way for combining multiple sensor control strategies using self-supervised feedback for perception and navigation in complex and dynamic environments.

Acknowledgments

We thank Pulkit Grover for discussions on information-theoretic measures of mixed discrete-continuous random variables. This material is based upon work supported by the National Science Foundation under Grants No. IIS-1849154, IIS-1900821, the United States Air Force and DARPA under Contract No. FA8750-18-C-0092, and a grant from the Manufacturing Futures Institute at Carnegie Mellon University.

References

[1] Waymo Occupancy and Flow Prediction Challenge. 2022. [Online; accessed 22-December-2022].

[2] Gabriel Agamennoni, Simone Fontana, Roland Y Siegwart, and Domenico G Sorrenti. Point clouds registration with probabilistic data association. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4092–4098. IEEE, 2016.

[3] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[4] 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.

[5] Siddharth Ancha, Gaurav Pathak, Srinivasa Narasimhan, and David Held. Active Safety Envelopes using Light Curtains with Probabilistic Guarantees. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/rss.2021.xvii.045.

[6] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using PointNet. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7163–7172, 2019.

[7] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[8] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2): 177–196, 2018.

[9] 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.

[10] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. SLIM: Self-supervised LiDAR scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13126–13136, 2021.

[11] P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

[12] Levi Cai, Burak Boyacıoğlu, Sarah E Webster, Lora Van Uffelen, and Kristi Morgansen. Towards autotuning of Kalman filters for underwater gliders based on consistency metrics. In *OCEANS 2019 MTS/IEEE Seattle*, pages 1–6. IEEE, 2019.

[13] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[14] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems Conference (RSS), Virtual*, 2021.

[15] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous exploration development environment and the planning algorithms. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8921–8928. IEEE, 2022. URL https://www.cmu-exploration.com/development-environment.

[16] Dorian Chan, Srinivasa Narasimhan, and Matthew O'Toole. Holocurtains: Programming Light Curtains via Binary Holography. *Computer Vision and Pattern Recognition*, 2022.

[17] Zhaozhong Chen, Christoffer Heckman, Simon Julier, and Nisar Ahmed. Weak in the NEES?: Auto-tuning Kalman filters with Bayesian optimization. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1072–1079. IEEE, 2018.

[18] Zhaozhong Chen, Nisar Ahmed, Simon Julier, and Christoffer Heckman. Kalman filter tuning with Bayesian optimization. *arXiv preprint arXiv:1912.08601*, 2019.

[19] 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.

[20] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2514–2523, 2020.

[21] 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.

[22] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

[23] Radu Danescu, Florin Oniga, and Sergiu Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1331–1342, 2011.

[24] 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.

[25] Zi-Li Deng, Yuan Gao, Chun-Bo Li, and Gang Hao. Self-tuning decoupled information fusion Wiener state component filters and their convergence. *Automatica*, 44 (3):685–695, 2008.

[26] 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.

[27] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3D LIDAR point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[28] 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.

[29] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[30] Patrick Fung and Mike Grimble. Dynamic ship position-

ing using a self-tuning Kalman filter. *IEEE Transactions on Automatic Control*, 28(3):339–350, 1983.

[31] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for discrete-continuous mixtures. *Advances in neural information processing systems*, 30, 2017.

[32] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3254–3263, 2019.

[33] Per Hagander and Björn Wittenmark. A self-tuning filter for fixed-lag smoothing. *IEEE Transactions on Information Theory*, 23(3):377–384, 1977.

[34] Dirk Hähnel and Wolfram Burgard. Probabilistic matching for 3D scan registration. In *Proc. of the VDI-Conference Robotik*, volume 2002. Citeseer, 2002.

[35] Pan He, Patrick Emami, Sanjay Ranka, and Anand Rangarajan. Self-Supervised Robust Scene Flow Estimation via the Alignment of Probability Density Functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):861–869, Jun. 2022. doi: 10.1609/aaai.v36i1.19968.

[36] David Held, Jesse Levinson, and Sebastian Thrun. Precision tracking with sparse 3D and dense color 2D data. In *2013 IEEE International Conference on Robotics and Automation*, pages 1138–1145. IEEE, 2013.

[37] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3D Shape, Color, and Motion for Robust Anytime Tracking. In *Robotics: science and systems*, volume 1. Citeseer, 2014.

[38] David Held, Devin Guillory, Brice Rebsamen, Sebastian Thrun, and Silvio Savarese. A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues. In *Robotics: Science and Systems*, volume 12, 2016.

[39] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Robust real-time tracking combining 3D shape, color, and motion. *The International Journal of Robotics Research*, 35(1-3):30–49, 2016.

[40] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[41] Peiyun Hu, David Held, and Deva Ramanan. Learning to optimally segment point clouds. *IEEE Robotics and Automation Letters*, 5(2):875–882, 2020.

[42] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3D object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020.

[43] Ouster Inc. Ouster OS1 Hardware Specification Sheet. 2021. URL https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p0-os1.pdf.

[44] 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.

[45] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4114–4123, 2021.

[46] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3D laser data. In *2008 IEEE international conference on robotics and automation*, pages 4043–4048. IEEE, 2008.

[47] 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.

[48] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15577–15586, 2021.

[49] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3D: Learning scene flow in 3D point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 529–537, 2019.

[50] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):5639–5646, 2022.

[51] Ameesh Makadia, Alexander Patterson, and Kostas Daniilidis. Fully automatic registration of 3D point clouds. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 1297–1304. IEEE, 2006.

[52] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[53] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[54] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11177–11185, 2020.

[55] T Moir and M Grimble. Optimal self-tuning filtering, prediction, and smoothing for discrete multivariable processes. *IEEE Transactions on Automatic control*, 29 (2):128–137, 1984.

[56] Yaakov Oshman and Ilan Shaviv. Optimal tuning of a Kalman filter using genetic algorithms. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page

4558, 2000.

[57] Thomas D Powell. Automated tuning of an extended Kalman filter using the downhill simplex algorithm. *Journal of Guidance, Control, and Dynamics*, 25(5):901–908, 2002.

[58] Yaadhav Raaj, Siddharth Ancha, Robert Tamburo, David Held, and Srinivasa Narasimhan. Exploiting and Refining Depth Distributions with Triangulation Light Curtains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[59] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[60] 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.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[62] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *Proceedings of the IEEE international conference on computer vision*, pages 2088–2096, 2017.

[63] Zachary Teed and Jia Deng. RAFT-3D: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021.

[64] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3D object recognition via classification of arbitrary object tracks. In *2011 IEEE International Conference on Robotics and Automation*, pages 4034–4041. IEEE, 2011.

[65] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[66] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[67] 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.

[68] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729. IEEE, 1999.

[69] 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.

[70] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2589–2597, 2018.

[71] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3523–3532, 2019.

[72] Yue Wang and Justin M. Solomon. PRNet: Self-supervised learning for partial-to-partial registration. *Advances in neural information processing systems*, 32, 2019.

[73] Wikipedia contributors. HSL and HSV — Wikipedia, The Free Encyclopedia, 2022. [Online; accessed 30-November-2022].

[74] Wikipedia contributors. F-score — Wikipedia, The Free Encyclopedia, 2022. [Online; accessed 05-December-2022].

[75] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020.

[76] 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.

[77] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[78] Yiming Zhao, Xiao Zhang, and Xinming Huang. A divide-and-merge point cloud clustering algorithm for LiDAR panoptic segmentation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7029–7035. IEEE, 2022.

[79] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European conference on computer vision*, pages 766–782. Springer, 2016.

[80] Deng Zi-Li and Li Chun-Bo. Self-tuning information fusion Kalman predictor weighted by diagonal matrices and its convergence analysis. *Acta Automatica Sinica*, 33 (2):156–163, 2007.