Appendix

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

Appendix A
Derivation of recursive Bayesian estimation

The goal is to infer at each timestep t a distribution $bel(x_t) = P(x_t \mid u_{1:t}, z_{1:t})$ over the current state x_t from the sequence of sensor observations $z_{1:t}$ and the known sequence of actions $u_{1:t}$. $bel(x_t)$ is computed using *recursive Bayesian estimation* [65]. Combining the definition of $bel(x_t)$ and the Markov property of the dynamic Bayes network, we can derive the following recursive relationship [65]:

$$
\begin{aligned}
bel(x_t) &= P(x_t \mid u_{1:t}, z_{1:t}) \\
&\propto P(x_t, z_t \mid u_{1:t}, z_{1:t-1}) \\
&= P(z_t \mid x_t, u_{1:t}, z_{1:t-1})\ P(x_t \mid u_{1:t}, z_{1:t-1}) \\
&= P(z_t \mid x_t, u_t)\ P(x_t \mid u_{1:t-1}, z_{1:t-1}) \\
&= P(z_t \mid x_t, u_t) \int_{x_{t-1}} P(x_{t-1}, x_t \mid u_{1:t-1}, z_{1:t-1})\ \mathrm{d}x_{t-1} \\
&= P(z_t \mid x_t, u_t) \int_{x_{t-1}} P(x_{t-1} \mid u_{1:t-1}, z_{1:t-1}) \\
&\qquad \cdot P(x_t \mid x_{t-1}, u_{1:t-1}, z_{1:t-1})\ \mathrm{d}x_{t-1} \\
&= P(z_t \mid x_t, u_t) \int_{x_{t-1}} \underbrace{P(x_{t-1} \mid u_{1:t-1}, z_{1:t-1})}_{bel(x_{t-1})} \\
&\qquad \cdot P(x_t \mid x_{t-1})\ \mathrm{d}x_{t-1} \\
&= P(z_t \mid x_t, u_t) \int_{x_{t-1}} bel(x_{t-1})\ P(x_t \mid x_{t-1})\ \mathrm{d}x_{t-1} \\
&= P(z_t \mid x_t, u_t)\ \overline{bel}(x_t) \text{ (Measurement update), where}
\end{aligned}
$$

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

Based on the above recursive equations, recursive Bayesian estimation alternates between the following two steps:

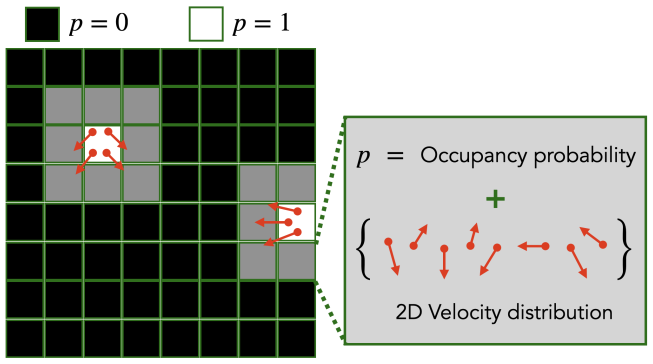
1) **Motion update step**: This step accounts for the dynamics of the environment. It first computes an intermediate quantity defined above: $\overline{bel}(x_t) = \int_{x_{t-1}} bel(x_{t-1})\ P(x_t \mid x_{t-1})\ \mathrm{d}x_{t-1}$. This is the result of "applying" a known or assumed motion model $P(x_t \mid x_{t-1})$ to the previous belief $bel(x_{t-1})$. In dynamic occupancy grids, this step accounts for the motion of scene points based on their current 2D velocities. The occupancies and velocities of the next timestep are computed based on the occupancies and velocities in the previous timestep. We use a constant velocity motion model with Gaussian noise in both velocity and position. When the motion model is applied to Fig. 7a, it is updated to Fig. 7b (illustration only). This correction step usually increases the uncertainty in occupancies and velocities.
2) **Measurement update step:** This step incorporates measurements from a sensor. It updates the prior belief $\overline{bel}(x_t)$ to $bel(x_t) \propto P(z_t \mid x_t, u_t)\ \overline{bel}(x_t)$ by weighting $\overline{bel}(x_t)$ by the likelihood of the observed measurements $P(z_t \mid x_t, u_t)$. This step usually reduces uncertainty in the state. In dynamic occupancy grids, occupancies are updated using the measurements from the light curtain. Since light curtains (or any depth sensor) only measure the locations of objects, this step does not update velocities. The velocity estimates are automatically refined in subsequent motion update steps. The measurement update reduced the probability of one of the positions of an object in Fig. 7b to Fig. 7c (illustration only).

Appendix B
Dynamic occupancy grids

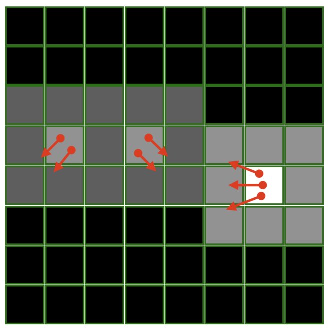
The dynamic occupancy grid, like conventional occupancy grids [29, 65], is an instance of Bayes filters. Occupancy grids [29, 65] are a standard tool in robotics for mapping the location of static objects in the environment. 2D occupancy grids that map objects from the top-down view are commonly used for mapping and SLAM in robot navigation. 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 Figure 7a). Each cell in the grid contains both (1) the occupany probability $p \in [0, 1]$, as well as (2) 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.

A. *Mathematical framework*

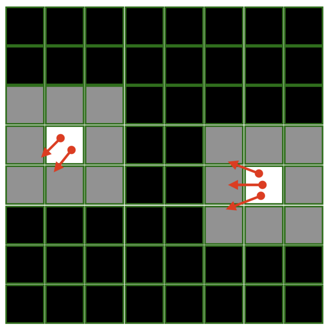
Our method is built upon dynamic occupancy grids introduced by Danescu. et. al. [23]. The authors describe particles as both representing a velocity distribution (i.e. weighted velocity hypotheses), as well as being "physical building blocks of the world". The former interpretation suggests that particles together represent the probability distribution of the velocity of a single physical scene point, whereas the latter suggests that each particle corresponds to its own scene point. Furthermore, the particles not only represent velocities, but their count represents the probability of occupancy. While the method is shown to be very promising, the precise role of particles and what they represent remains unclear. In this work, we re-derive dynamic occupancy grids using a more rigorous mathematical



(a) Dynamic occupancy grid before the motion update step.



(b) After the motion update step.



(c) After the measurement update step.

Figure 7: *Dynamic occupancy grid.* **Grid structure:** The 2D grid represents the top-down view and is made up of cells. Like conventional static occupancy grids [29, 65], each cell contains an occupancy probability $p \in [0, 1]$. Dark indicates low occupancy probability and bright indicates high occupancy probability. 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. **Grid updates:** The grid is a Bayes filter [65] that consists of two steps: the motion update step and the measurement update step. (a) The grid before performing any update. (b) *Motion update step*: the occupancies and velocities of the next timestep are computed based on the grid in the previous timestep, increasing uncertainty. (c) *Measurement update step:* the occupancies are updated using the measurements from the light curtain, decreasing uncertainty. This will refine the velocity estimates.

analysis. We explicitly state the assumptions made and provide a precise interpretation of particles. Our framework can be derived from three reasonably mild assumptions:

Assumption 1: There are no collisions

Each cell can be occupied by at most one physical scene point with a single velocity. Cells are sufficiently small that multiple objects with different velocities cannot exist ("collide") within a cell.

This assumption is required for a single velocity of a cell to be well-defined. Assumption 1 paves the way for a straightforward interpretation particles: all particles belonging to a cell represent a probability distribution over the single velocity of that cell. Assumption 1 allows us to define the state space of occupancies and velocities.

Representing the state space: Each cell is indexed by $i \in \mathcal{I}$ from an index set $\mathcal{I}$ of all cell in the grid. At timestep t, the state of the i-th cell is denoted by $x_t^i = (o_t^i, v_t^i)$. It contains two variables. The first is a binary occupancy variable $o_t^i \in \{0, 1\}$ which denotes whether the cell is occupied or not. The second is a 2D velocity variable $v_t^i \in \mathbb{R}^2$ representing the continuous velocity of the cell (*if* it is occupied) from the top-down view. The overall state of the dynamic occupancy grid x_t is a concantenation of the states of all cells in the grid, i.e. $x_t = \{x_t^i = (o_t^i, v_t^i) \mid i \in \mathcal{I}\}$. Note that the variable v_t^i is "conditional": it is only defined when the cell is occpied i.e. $o_t^i = 1$.

Assumption 2: Constant velocity motion model

Each scene point moves with a constant velocity, with added Gaussian noise.

Any motion can be approximated by a constant velocity motion model as long as the time interval is sufficiently small. Therefore, this assumption is reasonable in our setting since light curtains operate at very high speeds (45-60 Hz). Note that although we use the constant velocity motion model in this work following [23], the dynamic occupancy grid framework can still be used by swapping it with any other motion model of choice.

Let $\epsilon_t^i \sim \mathcal{N}(0, R_\epsilon)$ and $\delta_t^i \sim \mathcal{N}(0, R_\delta)$ be the Gaussian noise in velocity and position respectively for the i-th cell in the grid at time t. And let pos^i denote the 2D location of the center of the i-th cell. The constant velocity motion model can be expressed mathematically as

$$o_{t+1}^j, v_{t+1}^j = \begin{cases} 1,\ v_t^i + \epsilon_t^i & \text{if } \exists i \in \mathcal{I} \text{ such that } o_t^i = 1 \text{ and} \\ & \quad \text{pos}^j \approx \text{pos}^i + v_t^i\ \Delta t + \delta_t^i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In other words, a cell j will be occupied in the next timestep if and only if there exists another cell i in the previous timestep which moves to j under the constant velocity motion model. In that case, the velocity of cell j will be equal to that of cell i modulo the Gaussian noise. The equality is approximate taking into account the finite size of the cell.

Representing the belief distribution: A dynamic occupancy grid represent the current belief over velocities and occupancies of the environment. It is a probability distribution over states x_t described above. The state space is extremely large. Dynamic occupancy grids represent the belief compactly by making the following assumption:

Assumption 3: Cells are mutually independent

The probability distributions of all cells are mutually independent. This is a standard assumption made for occupancy grids for computational tractability.

Each cell contains two distributions:

- Occupancy distribution (ω_t^i): o_t^i is a Bernoulli random variable over $\{0, 1\}$ with probability $\omega_t^i \in [0, 1]$.
- Velocity distribution (V_t^i): v_t^i is a random variable over $\mathbb{R}^2$. It is represented by a set of M weighted particles $V_t^i = \{(v_t^{i,m}, p_t^{i,m}) \mid 1 \leq m \leq M\}$. The m-th particle has a velocity $v_t^{i,m}$ and weight $p_t^{i,m}$ that add up to 1 i.e. $\sum_{m=1}^{M} p_t^{i,m} = 1$. The larger the number of particles used, the better is the particle approximation to the true continuous velocity distribution.

The velocity distribution acts like a "conditional" distribution. The probability that cell i is occupied with velocity $v_t^{i,m}$ is the product $\omega_t^i\ p_t^{i,m}$ of the probability of being occupied (ω_t^i) and the probability of having the velocity $v_t^{i,m}$ given that it is occupied ($p_t^{i,m}$). The probability of being unoccupied is simply $1 - \omega_t^i$. Since cells are assumed to be mutually independent, the probability of the entire grid is the product of probabilities of individual cells in the grid.

B. Motion update step

We will now derive the motion update equations of the dynamic probability grid. These equations govern how particles will move across the grid and be reweighted. Consider a simplified grid shown in Fig. 8a. Assume that there are only three cells in the grid, and only cells 1 and 2 are occupied at time t. The occupancy and velocity distributions are illustrated in the figure, and particles $(v_t^{1,m_1}, p_t^{1,m_1})$ from cell 1 and $(v_t^{2,m_2}, p_t^{2,m_2})$ from cells 2 move to cell 3 in the next timestep $t+1$ (assuming that noise has been incorporated in v_t^{1,m_1}, v_t^{2,m_2}). What should be the occupancy and velocity distribution of cell 3?

Let E_1 be the event that the particle from cell 1 enters cell 3. Similarly, let E_2 be the event that the particle from cell 2 enters cell 3. We have that $P(E_1) = \omega_t^1\ p_t^{1,m_1}$ and $P(E_2) = \omega_t^2\ p_t^{2,m_2}$. Cell 3 will be occupied when either E_1 or E_2 happens (from Eqn. 3). Its occupancy probability after the motion update is $\overline{\omega}_{t+1}^3 = P(E_1 \cup E_2)$. Now, from assumption 1 (the no collision assumption), objects of different velocities cannot occupy the same cell. Therefore events E_1 and E_2 must be disjoint. From the law of total probability, if E_1 and E_2 are disjoint, then $P(E_1 \cup E_2) = P(E_1) + P(E_2)$. Therefore, the occupancy probability of cell 3 after the motion update $\overline{\omega}_{t+1}^3 = \omega_t^1\ p_t^{1,m_1} + \omega_t^2\ p_t^{2,m_2}$. The conditional velocity distribution of cell 3 will comprise of v_t^{1,m_1} and v_t^{2,m_2}, with weights proportional to $P(E_1)$ and $P(E_2)$ respectively. This leads us to the general motion update of dynamic occupancy grids:

Motion update step for dynamic occupancy grids (4)

$$\overline{\omega}_{t+1}^j = \sum_{i \in \mathcal{I}} \omega_t^i \sum_{m=1}^{M_i} p_t^{i,m}\ \mathbb{I}[\text{pos}^j \approx \text{pos}^i + v_t^{i,m}\ \Delta t + \delta_t^{i,m}]$$

$$\overline{V}_{t+1}^j = \left\{ \left(v_t^{i,m} + \epsilon_t^{i,m}, \frac{\omega_t^i\ p_t^{i,m}}{\overline{\omega}_{t+1}^j}\right) \;\middle|\; i, m : \text{pos}^j \approx \text{pos}^i + v_t^{i,m}\ \Delta t + \delta_t^{i,m} \right\}$$

Note that the above derivation does not require assumption 3; the law of total probability applies even when the distributions of incoming cells are not independent! Therefore, the motion update is exact even if we treat ω_t^i and V_t^i as marginal distributions of potentially correlated cells. Assumption 3 is however required for the measurement update step (see App. B-C).

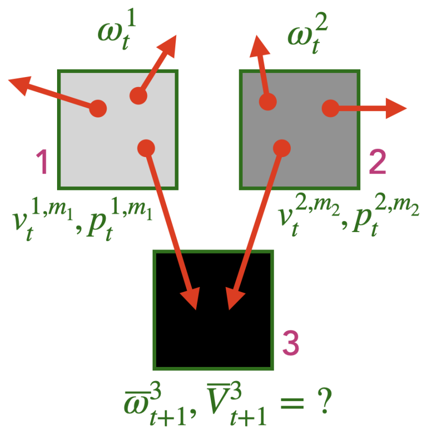

ω_t^1
ω_t^2
1
2
3
v_t^{1,m_1}, p_t^{1,m_1}
v_t^{2,m_2}, p_t^{2,m_2}
$\overline{\omega}_{t+1}^3, \overline{V}_{t+1}^3 = ?$

Figure 8: Applying the motion model to update occupancies and velocities of cells in the motion update step. ω's are occupancy probabilities of cells, v's and p's are velocities and weights of the individual particles respectively.

When adding the $\omega_t^i \; p_t^{i,m}$ terms from incoming particles to compute $\overline{\omega}_{t+1}^j$, the sum should not exceed 1 under the no collision assumption (assumption 1). However, in practice, this may be violated. In such cases, we truncate the occupancy probability to 1 following [23]. However, this happens rarely; we needed to perform truncation only 0.35% times on average.

C. Measurement update step

Sensors such as light curtains and LiDARs provide depth information, from which the current occupancy of cells can be inferred. We use a post-processing algorithm described in [42] to process sensor data and output a detection variable $z_t^i \in \{\texttt{OCCUPIED}, \texttt{FREE}, \texttt{UNKNOWN}\}$ for each cell i in the grid. z_t^i indicates the presence, absence or lack of knowledge about objects inside cell i.

For LiDAR scans, Hu et al. [42] marks each cell that contains LiDAR points as `OCCUPIED`. Then, it uses the fact that if a 3D point was detected, light must have traveled between the sensor and the detected point in a straight line without obstruction. Therefore, rays are cast starting from the sensor to the `OCCUPIED` cells using an efficient voxel traversal algorithm [3]. Cells lying along these rays are marked `FREE`. Any cells that remain unclassified are marked as `UNKNOWN`. This method exploits visibility constraints of light to extract the maximum possible information from a 3D scan. We use the same processing method customized for light curtains. When a light curtain is placed on a set of cells in the grid, the cells are classified as `OCCUPIED` or `FREE` based on whether points were detected inside the cell. Raycasting to occupied cells is also performed to discover additional `FREE` cells. Figure 2b visualizes an example of visibility classification. Cells detected as `OCCUPIED` are shown in red. Cells shown in blue are inferred as `FREE` because they either lie undetected on the curtain or lie on rays cast to red cells. `UNKNOWN` cells are shown in gray.

The measurement update step takes the visibility classification as input. Our observation model treats this classification as a noisy observation of true occupancy. We do not update the occupancy of `UNKNOWN` cells. For known cells, we assume a false positive rate $\alpha_{\text{fp}} \in [0,1]$ and a false negative rate $\alpha_{\text{fn}} \in [0,1]$. The observation model is:

$$\begin{aligned} P(z_t^i = \texttt{OCCUPIED} \mid o_t^i = 1) &= 1 - \alpha_{\text{fn}} \\ P(z_t^i = \texttt{FREE} \mid o_t^i = 1) &= \alpha_{\text{fn}} \\ P(z_t^i = \texttt{OCCUPIED} \mid o_t^i = 0) &= \alpha_{\text{fp}} \\ P(z_t^i = \texttt{FREE} \mid o_t^i = 0) &= 1 - \alpha_{\text{fp}} \end{aligned}$$

We first use the assumption that all cells are mutually independent (assumption 3) to write the belief of the overall grid $\overline{bel}(x_t) = \prod_{i\in\mathcal{I}} \overline{bel}(x_t^i)$ as a product of belief distributions of each cell in the grid. Since the likelihood function $P(z_t \mid x_t) = \prod_{i\in\mathcal{I}} P(z_t^i \mid o_t^i)$ is also independent for each cell, the updated posterior belief $bel(x_t \mid z_t) \propto \overline{bel}(x_t) \; P(z_t \mid x_t)$ can be computed independently for each cell. Given the prior occupancy distribution $\overline{\omega_t^i}$ and an observation z_t^i for cell i, its occupancy distribution after the measurement update can be computed using the Bayes rule:

Measurement update step for dynamic occupancy grids (5)

$$\omega_t^i = \frac{\overline{\omega_t^i} \; P(z_t^i \mid o_t^i = 1)}{\overline{\omega_t^i} \; P(z_t^i \mid o_t^i = 1) + (1 - \overline{\omega_t^i}) \; P(z_t^i \mid o_t^i = 0)}$$

Depth sensors only provide information about the occupancy of cells; they do not directly measure object velocities. Velocities are inferred indirectly by a combination of measurement- and motion- update steps. The measurement update incorporates information about occupancy and the motion update infers velocities that are consistent with occupancy across timesteps in a principled probabilistic manner. Therefore, our method estimates velocities from depth measurements without requiring explicit data association across frames!

Appendix C
Computing depth probabilities using raymarching

The depth probability of a cell in the grid is the probability that the depth of the scene along the cell's direction is the cell's location. In other words, it is the probability that a visible surface exists in the cell i.e. the cell is occupied and all other occluding cells are empty. Once we compute the depth probability of each cell in the grid, we can place a curtain that lies on the cells with the highest depth probability.

How do we compute the depth probability in a probabilistic occupancy grid? We borrow the idea of "ray marching" from the literature on volumetric rendering [66, 53]. In order to reconstruct the implicit depth surface from a probabilistic volume, ray marching travels along a ray originating from the sensor and computes the probability of visibility and occlusion at each point. Tulsiani et al. [66] performs this for discretized 3D grids (similar to our case) whereas NeRFs [53] perform this in a continuous space using neural radiance fields. Consider an example of raymarching in Fig. 2c. Let the sequence of cells on a ray be indexed as $1, 2, \ldots, n, \ldots N$. Recall from App. B that ω_t^i is the occupancy probability of the i-th cell at timestep t. The depth probability of the n-th cell $P_t^{\text{D}}(n) = \omega_t^n \; \prod_{i=1}^{n-1}(1-\omega_t^i)$ is product of the probabilities that the n-th cell is occupied (ω_t^n) and the probabilities that each i-th cell on the ray before the n-th cell is unoccupied $(1-\omega_t^i)$ so that light can reach the n-th cell unoccluded. Let us define the "visibility" probability $P_t^{\text{V}}(n) = \prod_{i=1}^{n}(1-\omega_t^i)$ that all cells are visible upto the n-th cell. Then, we have the following recursive equations:

$$\begin{aligned} P_t^{\text{V}}(i) &= P_t^{\text{V}}(i-1) \; (1-\omega_t^i) \\ P_t^{\text{D}}(i) &= P_t^{\text{V}}(i-1) \; \omega_t^i \end{aligned}$$

These recursive equations can be used to compute the depth probability of each cell along a ray efficiently in time $O(N)$ linear in the number of cells on that ray. This strategy is implemented as follows. For each camera ray, we perform the ray-marching procedure and compute the depth probability of each cell along that ray. Then, for each camera ray, we place the curtain at the cell with the maximum depth probability.

Appendix D
Maximizing information gain

Consider the dynamic Bayes network shown 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 taking an action u_t. Information gain is a well-studied quantity in information theory and is usually defined as:

$$\mathrm{IG}(x_t, z_t \mid u_t) = \underbrace{\mathrm{H}(P(x_t))}_{\text{entropy of } x_t} - \underbrace{\mathbb{E}_{z_t|u_t}\big[\mathrm{H}(P(x_t \mid z_t, u_t))\big]}_{\text{conditional entropy of } x_t \,\mid\, z_t \text{ under } u_t} \quad (6)$$

The information gain is the expected reduction in entropy (i.e. uncertainty) in x_t before and after taking the action u_t. Ancha et al. [4] showed that under certain assumptions, the information gain of conventional occupancy grids on placing light curtains is equal to the sum of binary entropies of the occupancy probabilities of the cells that the curtain lies on.

While information gain for conventional occupancy grids is straightforward to derive, 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. Consider the state x_t^i of the i-th cell in the grid. The space of the state x_t^i is:

$$x_t^i \in \underbrace{\{\text{unoccupied}\}}_{\text{discrete space}} \cup \underbrace{\{\text{occupied with } v_t^i \mid v_t^i \in \mathbb{R}^2\}}_{\text{continuous space}}$$

The cell can either be unoccupied, or be occupied with a continuous velocity. Unfortunately, the entropy of such mixed discrete-continuous spaces is not well-defined [31]. Therefore the "2H-estimator" in Eqn. 6 (named so because it contains two entropy terms) cannot be used to calculate information gain since the individual terms on the right hand side are not well-defined.

Fortunately, information gain (unlike entropy) is well-defined for most distributions, including discrete-continuous mixtures [31]. This is possible by using a more general definition of information gain given by the "Radon–Nikodym" derivative [31]:

$$\mathrm{IG}(x_t, z_t \mid u_t) = \int_{x_t, z_t} \underbrace{\log \frac{dP_{x,z}}{dP_x P_z}}_{\text{Radon–Nikodym derivative}} dP_{x,z} \quad (7)$$

The Radon–Nikodym is well-defined for discrete-continuous mixtures [31]. When the individual entropy terms of Eqn. 6 (the 2H-estimator) are well-defined, the more general definition of Eqn. 7 reduces to Eqn. 6. In other words, the two definitions are consistent.

We will now derive the information gain of dynamic occupancy grids using the more general Radon–Nikodym definition. Here, we derive the information gain of a single i-th cell. Let ω be the occupancy probability of the cell. Let the continuous velocity distribution of the cell be denoted by $P(v)$. Assume that we place a curtain on this cell, and we obtain an observation $z_t^i \in \{0, 1\}$ to be a noisy measurement of the cell's occupancy. This is assuming that we are using a depth sensor that can only partially observe occupancy but cannot directly observe velocities. Let α_{fp} and α_{fn} be the false-positive and false-negative rates of the sensor respectively. Then,

$$\begin{aligned}
&\mathrm{IG}(x_t^i \mid z_t^i) \\
&= \int_{x,z} dP_{x,z} \log \frac{dP_{x,z}}{dP_x P_z} \quad \text{(Radon-Nikodym formulation)} \\
&= \underbrace{(1-\omega)(1-\alpha_{\text{fp}}) \log \frac{(1-\omega)(1-\alpha_{\text{fp}})}{(1-\omega)P(z_t^i = 0)}}_{\text{unoccupied and undetected}} + \\
&\quad \underbrace{(1-\omega)\ \alpha_{\text{fp}} \log \frac{(1-\omega)\ \alpha_{\text{fp}}}{(1-\omega)P(z_t^i = 1)}}_{\text{unoccupied and detected}} + \\
&\quad \int_v \underbrace{\omega\ P(v)dv\ (1-\alpha_{\text{fn}}) \log \frac{\omega\ P(v)dv\ (1-\alpha_{\text{fn}})}{\omega\ P(v)dv\ P(z_t^i = 1)}}_{\text{velocity v and detected}} + \\
&\quad \int_v \underbrace{\omega\ P(v)dv\ \alpha_{\text{fn}} \log \frac{\omega\ P(v)dv\ \alpha_{\text{fn}}}{\omega\ P(v)dv\ P(z_t^i = 0)}}_{\text{velocity v and undetected}} \\
&= (1-\omega)(1-\alpha_{\text{fp}}) \log \frac{(1-\alpha_{\text{fp}})}{P(z_t^i = 0)} + (1-\omega)\ \alpha_{\text{fp}} \log \frac{\alpha_{\text{fp}}}{P(z_t^i = 1)} \\
&\quad + \omega\ (1-\alpha_{\text{fn}}) \log \frac{(1-\alpha_{\text{fn}})}{P(z_t^i = 1)} + \omega\ \alpha_{\text{fn}} \log \frac{\alpha_{\text{fn}}}{P(z_t^i = 0)} \\
&= -\big[(1-\omega)(1-\alpha_{\text{fp}}) + \omega\ \alpha_{\text{fn}}\big] \log P(z_t^i = 0) \\
&\quad - \big[(1-\omega)\ \alpha_{\text{fp}} + \omega(1-\alpha_{\text{fn}})\big] \log P(z_t^i = 1) \\
&\quad - \omega\ \mathrm{H}(\alpha_{\text{fn}}) - (1-\omega)\ \mathrm{H}(\alpha_{\text{fp}}) \\
&= -P(z_t^i = 0) \log P(z_t^i = 0) - P(z_t^i = 1) \log P(z_t^i = 1) \\
&\quad - \omega\ \mathrm{H}(\alpha_{\text{fn}}) - (1-\omega)\ \mathrm{H}(\alpha_{\text{fp}}) \\
&= \mathrm{H}(z) - \omega\ \mathrm{H}(\alpha_{\text{fn}}) - (1-\omega)\ \mathrm{H}(\alpha_{\text{fp}}) \\
&= \mathrm{H}(\omega) \quad \text{(assuming that } \alpha_{\text{fp}} = \alpha_{\text{fn}} = 0)
\end{aligned}$$

Assumption 1: We assume that the sensor is accurate (i.e. the false positive rate α_{fp} and the false negative rate α_{fn} are both close to zero). Then, the information gain of a single cell due to placing a light curtain on that cell is equal to its binary occupancy entropy $\mathrm{H}_{\text{occ}}(\omega) = -\omega\ \log_2 \omega - (1-\omega)\ \log_2(1-\omega)$.

Assumption 2: We assume that all cells are independently distributed. Since the information gain of independently distributed random variables is the sum of information gain of individual variables [22], the total information gain is the sum of binary cross entropies $\mathrm{H}_{\text{occ}}(\omega_t^i)$ of the cells that the curtain lies on. This is similar to the information gain in Ancha et al. [4]. However, we have been able to prove this mathematically in the more complex case of mixed discrete-continuous distributions.

This theoretical result is also intuitive – since the depth sensor measurements only provide information about occupancy and not velocity, it is not surprising that the information gain is equal to the total occupancy uncertainty.

Appendix E
Advantages of light curtains over conventional depth sensors

LiDARs have long range and high accuracy under strong ambient light. However, compared to light curtains, they have poor vertical resolution ($\leq$128 rows), low frame rate (5-20Hz), and are very expensive (>$20K). Table III compares LiDARs and light curtains.

	LiDAR	Light Curtains
Resolution	128 rows	1280 rows
Cost	~$20,000	~$1,000
Frame rate	10-20 Hz	45-60 Hz

Table III: Comparison between a modern Ouster OS1 [43] LiDAR, and Programmable Light Curtains [9].

Passive RGBD sensors (stereo sensors that do not project light) have high spatial and temporal resolution and are inexpensive. However, their accuracy is poor in non-textured regions due to inaccuracies in stereo feature matching.

Active RGBD sensors (like the Kinect sensor that projects light) inherit the benefits of stereo sensors but also work in texture-less regions. However, they have virtually no range outdoors.

Light curtains combine the best of these sensors. They have a long range (nearly 35-50m) both outdoors and indoors, high spatial resolution (1280 rows) and temporal resolution (45-60 Hz), work for textured and texture-less regions, and are inexpensive (<1$K). These advantages have been demonstrated in previous works on programmable light curtains [9, 58].

Appendix F
Using an extra grid for thread-safety and efficiency

Our parallelized pipeline contains three threads: (1) light curtain sensing, (2) Bayes filtering using dynamic occupancy grids, and (3) computing curtain placements. How many grids are required to run these threads in parallel, especially threads 2 and 3?

The motion update step (Eqns. 1, 4) moves particles across the grid according to the motion model. The particles cannot be moved in place inside the same grid since it may cause the same particle to be erroneously moved more than once. Therefore, the motion update step requires two grids: a "source/current" grid and a "destination/next" grid. Particles from the current grid are copied, moved and placed in the next grid. After the motion update is complete, the roles of the current and next grids are swapped. The next grid is now assigned to be the new "current" grid since it is now the most up-to-date, incorporating the latest measurements. In the next motion update step, particles move from this grid to the older current grid (now taking the role of the "next" grid).

The curtain computation thread also performs a motion update when it needs to forecast the current grid to a future timestep (when the next curtain is expected to be imaged). It uses the current grid of the Bayes filtering thread as the source, but requires a third, "forecasting" grid as a destination grid.

Although three grids are sufficient to implement parallelization, the pipeline can be made more efficient. Specifically, consider the situation where two motion updates take place simultaneously from "current" to "next" grids in the Bayes filtering thread and "current" to "forecasting" grid in the curtain computation thread. Once the motion update in the Bayes filtering thread is complete, it cannot immediately perform the next motion update step. It must wait for the curtain computation thread to finish forecasting using the "current" grid before the "current" and "next" grids can be swapped and the next motion update dirties the "current" grid. If we use an additional "extra" grid, the Bayes filtering thread can use this as the destination grid for its next motion update step without needing to wait on the curtain computation thread to finish the latter's forecasting step.

Our parallelized pipeline tightly integrates the three inter-dependent processes in a closed loop. We use a total of four grids to simultaneously guarantee the following two properties: (1) grids in use are never mistakenly overwritten, and (2) no thread ever needs to wait on another to finish processing.

Appendix G
Evaluation Metrics

As mentioned in Sec. VIII-B, *forecasted occupancy* [50, 1] simultaneously captures both the accuracy of occupancy estimates as well as velocity estimates. This metric is especially pertinent for obstacle avoidance where future occupancy of obstacles is needed to plan paths that avoid collisions.

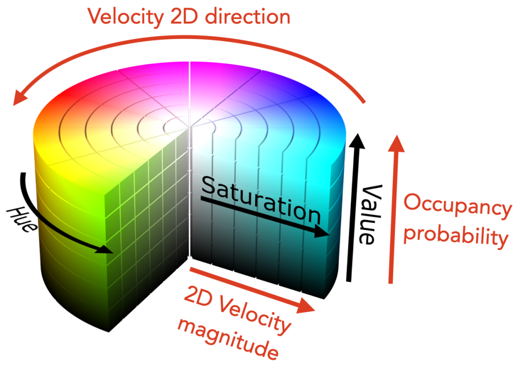
We will use the notation for dynamic occupancy grids introduced in Sec. B-A. The current dynamic occupancy grid at timestep t is represented by $G_t = \{\omega_t^i, V_t^i \mid i \in \mathcal{I}\}$, where ω_t^i is the Bernoulli occupancy probability of the i-th cell, and $V_t^i = \{(v_t^{i,m}, p_t^{i,m}) \mid 1 \leq m \leq M\}$ is the set of M weighted particles that represents the velocity distribution of the i-th cell.

To evaluate G_t, we first apply the motion update step (Eqns. 1, 4) to forecast it by a time Δt and obtain the dynamic occupancy grid $G_{t+\Delta t}$ at time $t + \Delta t$. Then, the forecasted occupancy probabilities $\{\omega_{t+\Delta t}^i \mid i \in \mathcal{I}\}$ are evaluated against the ground truth occupancies $\{o_{t+\Delta t}^i \mid i \in \mathcal{I}\}$. We follow prior works [40, 52, 62] that treat the evaluation of occupancy as a classification problem and compute binary occupancies $\widetilde{o}_{t+\Delta t}^i = \mathbb{I}(\omega_{t+\Delta t}^i \geq 0.5)$ thresholded at 0.5 probability.

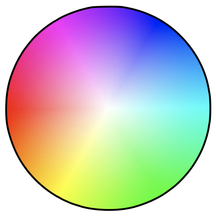
We use the following metrics to evaluate the quality of predicted occupancy. We ignore cells that are occluded (in the ground truth) since (1) they cannot be observed by optical sensors, and (2) they are not the closest object to the robot making them less relevant for obstacle avoidance. Let $\mathcal{I}_{\text{LOS}}$ be the subset of cells that are in the sensor's line-of-sight (LOS). Note that $\widetilde{o}_{t+\Delta t}^i, o_{t+\Delta t}^i \in \{0, 1\}$.

1) *Classification accuracy:* The fraction of cells whose occupancy is correctly predicted.

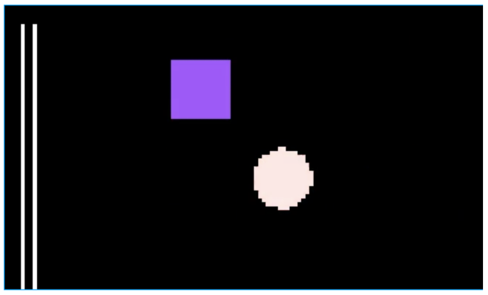
$$\texttt{Accuracy} = \frac{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}\{\widetilde{o}_{t+\Delta t}^i = o_{t+\Delta t}^i\}}{|\mathcal{I}_{\text{LOS}}|}$$



(a) HSV colorwheel used to visualize velocities.



(b) Colorwheel from the top-down view



(c) GT velocities of the simulated environment.

Figure 9: (a) The HSV (hue-saturation-value) colorwheel [73] used to visualize 2D velocities and occupancies. Value corresponds to the occupancy probability. The hue corresponds to the direction of the velocity. Saturation corresponds the magnitude of velocity. (b) The HSV colorwheel in the top-down view, denoting the direction of velocity. (c) Ground truth velocities and occupancies in the simulated environment. The grid shows a stationary wall to the left and two objects. The bluish-purple square is moving upwards in 2D (i.e. farther away from the sensor in 3D) whereas the other objects in white are stationary.

2) *Precision:* The fraction of cells predicted to be occupied that were actually occupied.

$$\texttt{Precision} = \frac{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}\{\widetilde{o}^i_{t+\Delta t} = 1\}\ \mathbb{I}\{o^i_{t+\Delta t} = 1\}}{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}\{\widetilde{o}^i_{t+\Delta t} = 1\}}$$

3) *Recall:* The fraction of occupied cells that were also predicted to be occupied.

$$\texttt{Recall} = \frac{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}\{\widetilde{o}^i_{t+\Delta t} = 1\}\ \mathbb{I}\{o^i_{t+\Delta t} = 1\}}{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}\{o^i_{t+\Delta t} = 1\}}$$

4) *F₁-Score:* A combination (harmonic mean) of precision and recall that is a commonly used for binary classification [74]. The F_1 score is robust to class imbalance; unlike precision and recall, it cannot be trivially improved by predicting mostly negative labels and mostly positive labels.

$$\texttt{F}_1\texttt{-Score} = \frac{2 \cdot \texttt{Precision} \cdot \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}}$$

5) *IoU:* The intersection-over-union between cells that are occupied (in ground truth) and cells that are predicted to be occupied.

$$\texttt{IoU} = \frac{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}(\widetilde{o}^i_{t+\Delta t} = 1)\ \mathbb{I}(o^i_{t+\Delta t} = 1)}{\sum_{i \in \mathcal{I}_{\text{LOS}}} \mathbb{I}(\widetilde{o}^i_{t+\Delta t} = 1 \text{ or } o^i_{t+\Delta t} = 1)}$$

For all metrics, a higher numerical score is better.

Appendix H
Visualizing Velocities and Occupancies

Fig. 9 shows how we visualize 2D velocities and occupancies (both ground truth and estimated velocities and occupancies). The visualization of an example ground truth grid is shown in Fig. 9c. We use the three-dimensional HSV colorwheel shown in Fig. 9a to jointly visualize velocities (two-dimensional) and occupancies (one-dimensional). The 'value' encodes the occupancy probability; dark means low occupancy probability and bright means high occupancy probability. The 'hue' encodes the direction of velocity. We show a top-down view of the HSV colorwheel in Fig. 9b for clarity. For example, the bluish-purple hue of the cuboid in Fig. 9c means that the cuboid is moving upwards in 2D i.e. away from the sensor in 3D. 'Saturation' encodes the magnitude of velocity. This means that white is stationary (e.g. the walls of the environment shown as parallel white lines) whereas colorful regions corresponds to high speed.

Appendix I
Non-stationary Rewards

Vanilla multi-armed bandits assume that the reward distribution for each action is stationary. The Q-value of an action after it was been performed n times is computed as $Q_n = \frac{1}{n}\sum_{i=1}^{n} R_i$, where R_i is the reward obtained in the i-th trial. This is equivalent to the following recursive update rule: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$, where the Q-value is incremented by the error scaled by a decaying factor $\frac{1}{n}$.

However, in our case, a single placement strategy may not be superior to the rest at all times and in all situations. The reward distribution for each strategy (action) may change with time. Hence, we assume that our rewards are non-stationary. For non-stationary rewards, we wish to give more weight to recent rewards than to older rewards. Therefore, the decaying parameter is replaced by a constant step-size parameter α: $Q_{n+1} = Q_n + \alpha\ [R_n - Q_n]$. This weights newer rewards exponentially more than older rewards according to the expression: $Q_n = (1-\alpha)^{n-1}\ R_1 + \sum_{i=2}^{n} \alpha\ (1-\alpha)^{n-i}\ R_i$.

Appendix J
Efficient Light Curtain Simulation

Fig. 10 shows the working principle behind the illumination module of a programmable light curtain. It consists of a fixed laser source that emits a light beam, and a rotating galvo-mirror that reflects and redirects the light in any desired direction. The laser beam is collimated to a thin rectangular sheet; however, in reality it is a prismatic slab containing a small divergence. The

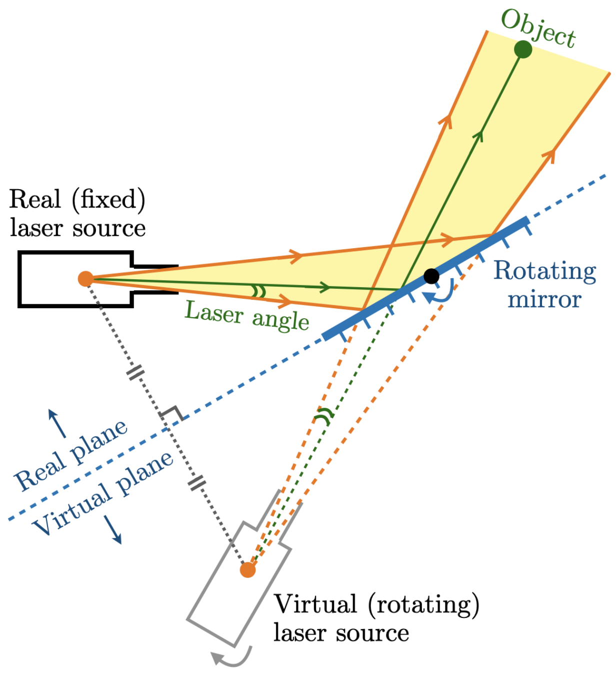


Figure 10: *Efficient light curtain simulation using a virtual laser.* A light curtain consists of a fixed laser source that emits a light beam, and a rotating galvo-mirror that reflects and redirects the light in any desired direction. The laser beam is collimated to a thin rectangular sheet; however, in reality it is a prismatic slab containing a small divergence. The pixel intensity of an object (shown by a green circle ●) imaged by the light curtain depends on the radiant intensity of the laser ray that is incident on the object (shown by the green ray →). A ray at the center of the beam has the highest intensity whereas a ray at the boundary of the beam (shown by orange rays →) has the lowest intensity. In order to simulate light curtain pixel intensities for a given object, we must compute its incident ray i.e. compute its “laser angle” shown by ∢. This computation can be expensive since (1) it involves tracing rays between the source and the object through a reflection at the mirror, and (2) it must be performed for each pixel. Our insight is to construct a “virtual” laser source by reflecting the real source about the mirror plane. Due to the laws of reflection, the reflected beam is equivalent to originating from the virtual source behind the mirror. This allows the laser angle to be computed efficiently by projecting the object point in the virtual source's frame. Furthermore, the virtual source needs to be reflected only once for each mirror configuration; all pixel points in the currently active camera column can be efficiently projected into the same virtual laser source.

pixel intensity of an object (shown by a green circle ●) imaged by the light curtain depends on the radiant intensity of the laser ray that is incident on the object (shown by the green ray →). A ray at the center of the beam has the highest intensity whereas a ray at the boundary of the beam (shown by orange rays →) has the lowest intensity. In order to simulate light curtain pixel intensities for a given object, we must compute its incident ray i.e. compute its “laser angle” shown by ∢.

Computing the laser ray incident to the object is expensive because (1) it involves tracing rays between the source and the object through a reflection at the mirror, and (2) it must be performed for each pixel. Our insight is to construct a “virtual” laser source by reflecting the real source about the mirror plane. Due to the laws of reflection, the reflected beam is equivalent to originating from the virtual source behind the mirror. This allows an object point's laser angle to be computed efficiently by projecting it in the virtual source's frame. Furthermore, the virtual source needs to be reflected only once for each mirror configuration; all pixel points in the currently active camera column can be efficiently projected into the same virtual source.

Appendix K
Full-stack Navigation

We integrate our system into a full-stack navigation pipeline based on the Autonomous Exploration Development Environment [15]. We mount the light curtain device on a mobile robot (see Fig. 3a). Our tightly integrated pipeline performs localization, mapping, planning, control and obstacle avoidance. We use ORB-SLAM3 [13] for localization and mapping that takes depth from light curtains as input while the planning and control capabilities are provided by Cao et al. [15]. The pipeline described in Sec. VII combines light curtain placement strategies using self-supervised multi-armed bandits with recursive Bayes estimation of dynamic occupancy grids. The output of this pipeline i.e. position and velocity estimates are used by the autonomy stack to perform dense mapping in an indoor environment and obstacle avoidance. Furthermore, the localization from ORB-SLAM3 [13] is fed back into our pipeline for ego-motion subtraction in the motion update step (Eqns. 1, 4). We show two demonstrations of our fully integrated autonomy stack:

A. Real-time dense mapping

Light curtains sense objects that intersect its surface at a high resolution. This ability can be leveraged to perform dense mapping and reconstruction of an environment. Please see a video of dense real-time reconstruction of an indoor hallway environment using our system on the project website. Fig. 11 shows a sideways and top-down projection of the same. The robot was operated in the indoor hallway environment, and 3D points detected by light curtains were input to ORB-SLAM3 [13], an RGB-D based localization and mapping system that estimates the robot's pose. The pose estimates are fed back into our pipeline to perform ego-motion subtraction in the motion update step. The robot trajectory is shown as a white line. Fig. 11a shows that the floor, walls and other objects are reconstructed densely and accurately. Fig. 11b contains a top-down orthographic view which shows the accuracy of our system's localization – the robot's trajectory was correctly determined to be an (approximately) closed loop around the building floor. This experiment serves to demonstrate our full-stack navigation pipeline using light curtains.

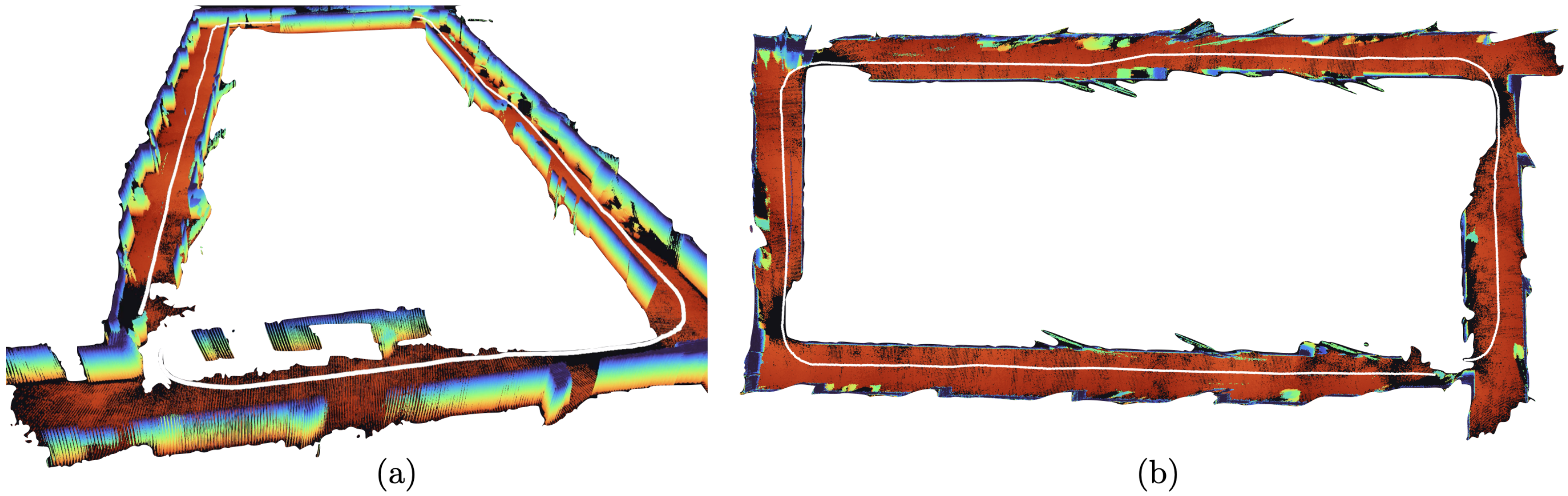


Figure 11: *Dense indoor reconstruction and mapping using our integrated system.* The light curtain was mounted on a mobile robot (Fig. 3a) and operated in an indoor hallway environment. Detected depth points from light curtains were input to ORB-SLAM3 [13], an RGB-D based localization and mapping system that estimates the robot's pose. The pose estimates are fed back into our pipeline to perform ego-motion subtraction in the motion update step. The robot trajectory is shown as a white line. (a) *Sideways perspective view:* showing dense reconstruction of walls, the floor and other objects. (b) *Top-down orthographic view:* showing the accuracy of localization (white line) and loop-closure. Please find the full video on the project website. This experiment serves to demonstrate our full-stack navigation pipeline using light curtains.

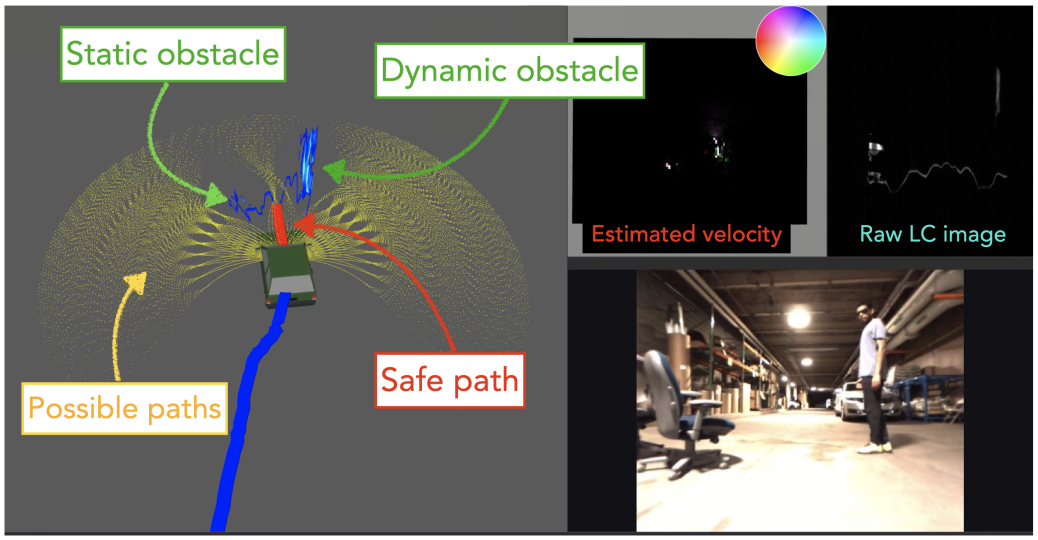


Figure 12: *Real-time obstacle avoidance using our integrated system.* The robot is represented by a green vehicle. The yellow curves show a library of dynamically feasible paths of the robot [14]. Points detected by the light curtain on the static (chair) and dynamic (person) objects are show in blue. The feasible paths that are expected to collide with objects are removed, and a safe path (shown in red) is chosen by the planner. Please find the full video on the project website.

B. *Real-time obstacle avoidance*

Light curtains are a fast sensor ($\sim$45 Hz) whose speed is leveraged by our system to produce position and velocity estimates at a high frequency ($\sim$35 Hz). We use these estimates for real-time obstacle avoidance, shown in Fig. 12. The robot is represented by a green vehicle. The yellow curves show a library of dynamically feasible paths of the robot [14]. Points detected by the light curtain are shown in blue. There are two objects in the scene: a static chair, and a moving person. Using the position estimates of obstacles, feasible paths that are expected to collide with objects are rejected, and a safe path (shown in red) is chosen by a local planner [14]. Please find the full video of real-time obstacle avoidance on the project website.