Appendix

Active Safety Envelopes using Light Curtains with Probabilistic Guarantees

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

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

A. Transition distributions for sampling random curtains

We first describe, in Algorithm 1 the procedure to sample a random curtain from the extended constraint graph $\overline{\mathcal{G}}$ by successively generating it using a transition probability function $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$. The only constraint on $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ is that it must equal to 0 if $\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}$ do not satisfy both the velocity and acceleration constraints of Equations (1, 2).

Algorithm 1: Sampling a random curtain from $\overline{\mathcal{G}}$

```
/* Inputs                                                    */
```
$\overline{\mathcal{G}} \leftarrow$ extended constraint graph
$P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t) \leftarrow$ transition prob. distribution
$P((\mathbf{X}_1, \mathbf{X}_2)) \leftarrow$ initial probability distribution

```
/* Progressively generate curtain                            */
```
Curtain $\leftarrow \{\}$

```
/* Initialization                                            */
```
Sample $(\mathbf{X}_1, \mathbf{X}_2) \sim P((\mathbf{X}_1, \mathbf{X}_2))$
Curtain $\leftarrow \{\mathbf{X}_1, \mathbf{X}_2\}$

```
/* Iteration                                                 */
```
for $t = 2$ *to* $T - 1$ **do**
  $\mathbf{X}_{t+1} \sim P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$
  Curtain $\leftarrow$ Curtain $\cup \{\mathbf{X}_{t+1}\}$

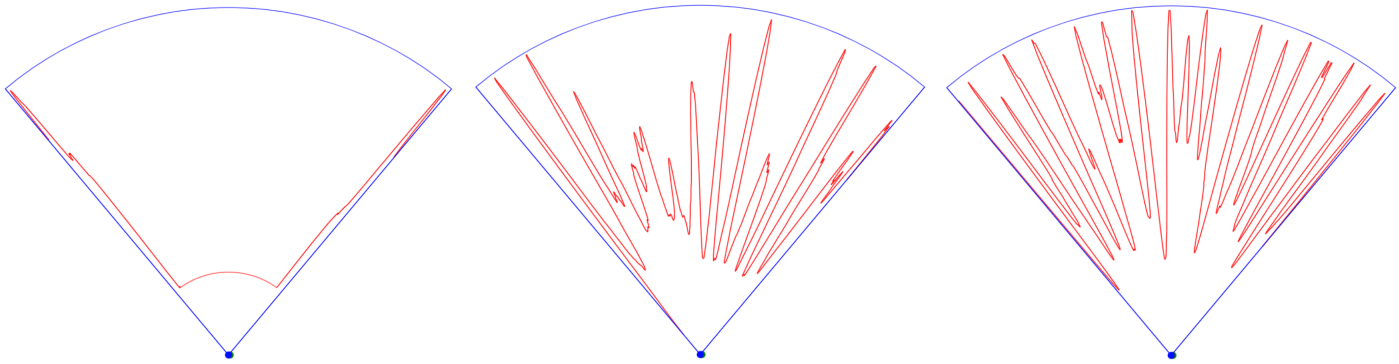
return Curtain

We initialize by sampling a location $\overline{\mathbf{N}}_2 = (\mathbf{X}_1, \mathbf{X}_2)$ according to an initial sampling distribution. At the $(t-1)$-th iteration, we will have sampled the $t-1$ nodes $(\overline{\mathbf{N}}_2, \ldots, \overline{\mathbf{N}}_t)$ corresponding to the first t control points $(\mathbf{X}_1, \ldots, \mathbf{X}_t)$ of the random curtain. At the t-th iteration, we sample $\mathbf{X}_{t+1}$ according to the transition probability distribution $\mathbf{X}_{t+1} \sim P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ and add $\mathbf{X}_{t+1}$ to the current set of control points. After all iterations are over, this algorithm generates a complete random curtain.

The above procedure to sample random curtain provides the flexibility to design any initial and transition probability distribution functions. Then, what are good candidate distributions? We use random curtains to detect the presence of objects in the scene whose location is unknown. Hence, the objective is to find the light curtian sampling distribution that maximizes the probability of detection of an object that might be placed at any arbitrary location in a scene. This objective will be achieved by random curtains that *cover a large area*. We now discuss a few sampling methods and qualitatively evaluate them in terms of the area covered by random curtains generated from them.

1. Uniform neighbor sampling: Perhaps the simplest transition probability distribution $P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$ for a given extended node $(\mathbf{X}_{t-1}, \mathbf{X}_t)$ is to select a neighboring control point $\mathbf{X}_{t+1}$ (that is connected by a valid edge originating from the node) with uniform probability. However, since the distribution does not take into account the physical locations of the current control point, it does not explicitly try to maximize coverage. To illustrate this, consider a random curtain that starts close to light curtain device. If it were to maximize coverage, the galvanometer would need to rotate so that the light curtain is placed farther from the device on subsequent camera rays. However, since its neighboring nodes are selected at random, the sampled locations on the next ray are equally likely to be nearer to the device than farther away from it. This can produce random curtains as shown in Fig. 7 (a).

2. Uniform linear setpoint sampling: a more principled way to sample neighbors is inspired by rapidly-exploring random trees (RRTs), which are designed to quickly explore and cover a given space. During tree expansion, an RRT first randomly samples a *setpoint* location, and selects the vertex that is closest to that location. We adopt a similar procedure. For any current node $(\mathbf{X}_{t-1}, \mathbf{X}_t)$, we first sample a setpoint distance $r \in [0, r_{\max}]$ uniformly at random on a line along the camera ray. The probability density of r is



(a) Uniform sampling of neighbors (b) Uniform linear sampling of setpoints (c) Uniform area sampling of setpoints

Fig. 7: Qualitative comparision of the coverage of random light curtains under different transition probability distributions. Sampled random curtains are shown in red. (a) *Uniform neighbor sampling*: for a given node, its neighbors on the next camera ray are sampled uniformly at random. This can produce random curtains that are at a constant distance away from the device. (b) *Uniform linear setpoint sampling*: for every camera ray, a setpoint distance $r \in [0, r_{\max}]$ is sampled uniformly at random. Then the neighbor closest to the setpoint is chosen. This has significantly higher coverage, but is biased towards sampling locations close to the device. (c) *Uniform area setpoint sampling*: for every camera ray, a setpoint distance $r \in [0, r_{\max}]$ is sampled with a probability proportional to r. This assigns a higher probability to a larger r, and corresponds to uniform *area* sampling. Then the neighbor closest to the setpoint is chosen. This method qualitatively exhibits the best coverage.

a constant and equal to $P(r) = 1/r_{\max}$. Then, we select a valid neighbor $\mathbf{X}_{t+1}$ that is closest to this setpoint location among all valid neighbors. Let us again consider the situation described in the previous approach, where the current node is located close to the light curtain device. When a setpoint is sampled uniformly along the next camera ray, there is high probability that it will correspond to a location that is farther away from the current node. Hence, the neighboring location $\mathbf{X}_{t+1}$ that is chosen on the next ray will likely lie away from the device as well. A random curtain sample generated from this distribution is shown in Fig. 7 (b). It tends to alternate between traversing near regions and far regions of the space in front of the curtain, covering a larger area than the previous sampling approach.

3. Uniform area setpoint sampling: We use the setpoint sampling approach described above, but revisit how the setpoint itself is sampled. Previously, the setpoint $r \in [0, r_{\max}]$ was sampled uniformly at random on the line along the ray, with $P(r) = 1/r_{\max}$. Now, we propose an alternate sampling distribution and provide some theoretical justification. Since we want to uniformly cover the area in front of the light curtain device, consider the following experiment. Let us sample a point (x, y) uniformly from the area within a circle of radius $r_{\max}$ (or equivalently, within any sector of that circle). Then the cumulative distribution function of $r = \sqrt{x^2 + y^2}$ is $P(r < r') = \pi r'^2 / \pi r_{\max}^2$ (area of the smaller circle divided by the area of entire circle), which implies that the probability density function of r is equal to $P(r) = 2r/r_{\max}^2$. This suggests that we must assign a higher probability to a larger r (proportional to r), since a larger area exists away from the center of the circle than near the center. Hence, we sample the setpoint r from $P(r) = 2r/r_{\max}^2$, by first sampling $s \sim \text{Uniform}(0, r_{\max}^2)$ and then setting $r = \sqrt{s}$. Finally, we select the valid neighbor $\mathbf{X}_{t+1}$ on the next ray that is closest to this setpoint. Since this method is motivated by sampling areas rather than sampling along a line, we call this approach "area setpoint sampling". An example curtain sampled using this approach is visualized in Fig. 7 (c), which generally exhibits the best coverage among all methods. We use this method to sample random light curtains for all experiments in this paper.

B. Dynamic programming for computing detection probability

A detailed alogrithm of our dynamic programming approach is given in Algorithm 2. To compute the quantity $P(\mathbf{D})$, we first decompose the overall problem into smaller subproblems by defining the *sub-curtain detection probability* $P_{\text{det}}(\mathbf{X}_{t-1}, \mathbf{X}_t) = P(\bigvee_{t'=t}^{T} \mathbf{D}_t \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$. This is the probability that a random curtain *starting* on the extended node $(\mathbf{X}_{t-1}, \mathbf{X}_t)$ and ending on the last camera ray, detects the object O between rays R_t and R_T. Note that the overall detection probability can be written in terms of the sub-curtain detection probabilities of the second ray as $P(\mathbf{D}) = \sum_{\mathcal{S}_2} P(\mathbf{X}_1, \mathbf{X}_2) \cdot P_{\text{det}}(\mathbf{X}_1, \mathbf{X}_2)$. Then we iterate over camera rays from R_T to R_2. The node detection probabilities on the last ray will simply be either 1 or 0, based on whether the object is detected at the node or not. After having computed node detection probabilities for all rays between $t+1$ and T, the probabilities for nodes at ray t can be computed using a recursive formula. Finally, after obtaining the probabilities for nodes on the initial rays, the overall detection probabilities can be computed as described previously.

Algorithm 2: Dynamic programming to compute detection probabilities $\overline{\mathcal{G}}$

```
/* Inputs                                                    */
```
$\overline{\mathcal{G}} \leftarrow$ extended constraint graph
$P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t) \leftarrow$ transition prob. distribution
$P((\mathbf{X}_1, \mathbf{X}_2)) \leftarrow$ initial probability distribution
$\{O_1, \ldots, O_T\} \leftarrow$ ground truth object locations
$\tau \leftarrow$ detection intensity threshold

$\mathcal{S}_t \leftarrow$ the set of nodes on the t-th camera ray in the constraint graph.

```
/* Detection at each location                                */
```
forall $\mathbf{X}_t \in \mathcal{S}_t, \quad 1 \leq t \leq T$ **do**
  $\mathbf{I}_t(\mathbf{X}_t \mid O_t) \leftarrow$ intensity using rendering
  $\mathbf{D}_t(\mathbf{X}_t \mid O_t) \leftarrow [\mathbf{I}_t(\mathbf{X}_t \mid O_t) > \tau]$

```
/* Initializing last ray                                     */
```
for $(\mathbf{X}_{T-1}, \mathbf{X}_T) \in \mathcal{S}_T$ **do**
  $P_{\text{det}}(\mathbf{X}_{T-1}, \mathbf{X}_T) \leftarrow \mathbf{D}_t(\mathbf{X}_T \mid O_T)$

```
/* Dynamic programming loop                                  */
```
for $t = T - 1$ *to* 2 **do**
  for $(\mathbf{X}_{T-1}, \mathbf{X}_T) \in \mathcal{S}_t$ **do**
    if $\mathbf{D}_t(\mathbf{X}_t, O_t) = 1$ **then**
      $P_{\text{det}}(\mathbf{X}_{T-1}, \mathbf{X}_T) \leftarrow 1$
    else
      $P_{\text{det}}(\mathbf{X}_{T-1}, \mathbf{X}_T) \leftarrow$
      $\sum_{\mathbf{X}_{T+1}} P_{\text{det}}(\mathbf{X}_T, \mathbf{X}_{T+1}) \cdot P(\mathbf{X}_{t+1} \mid \mathbf{X}_{t-1}, \mathbf{X}_t)$

```
/* Initial ray                                               */
```
$P(\mathbf{D}) \leftarrow 0$
for $(\mathbf{X}_1, \mathbf{X}_2)$ **do**
  $P(\mathbf{D}) \leftarrow P(\mathbf{D}) + P(\mathbf{X}_1, \mathbf{X}_2) \cdot P_{\text{det}}(\mathbf{X}_1, \mathbf{X}_2)$

return $P(\mathbf{D})$

C. Computational complexity of the extended constraint graph

In this section, we discuss the computational complexity associated with the extended constraint graph. Let K be the number of discretized control points per camera ray in the constraint graph, and let T be the number of camera rays.

Constraint graph size: in the original constraint graph $\mathcal{G}$ of Ancha et al. [1], since a node $\mathbf{N}_t = \mathbf{X}_t$ contains only one control point, there can be $O(K)$ nodes per camera ray and $O(K^2)$ edges between consecutive camera rays. This means that there are $O(TK)$ nodes and $O(TK^2)$ edges in the graph.

However, in the extended constraint graph $\overline{\mathcal{G}}$, each node $\overline{\mathbf{N}}_t = (\mathbf{X}_{t-1}, \mathbf{X}_t)$ contains a pair of control points. Hence, there can be up to $O(K^2)$ nodes per camera ray and $O(K^4)$ edges between consecutive camera rays! This implies that the total nodes and edges in the graph can be up to $O(TK^2)$ and $O(TK^4)$ respectively.

Dynamic programming: dynamic programming involves visiting each node and each edge in the graph once. Therefore, the worst-case computation time of dynamic programming in the extended constraint graph, namely $O(TK^4)$, might seem prohibitively large at first. However, the additional acceleration constraints in $\overline{\mathcal{G}}$ can significantly limit the increase in the number of nodes and edges. Additionally, we perform graph pruning as a post-processing step, to remove all nodes in the graph that do not have any edges. Since the topology of the constraint graph is fixed, the graph creation and pruning steps can be done offline and only once. These optimizations enable our dynamic programming procedure to be very efficient, as shown in Sec. VII-A. That being said, any slow down in dynamic programming is generally acceptable because it is only used for offline probabilistic analysis.

Random curtain generation: random curtains are placed by our online method. Fortunately, generating random curtains from the constraint graph is very fast. It involves a single forward pass (random walk) through the graph, visiting exactly one node per ray. It also involves parsing each visited node's transition probability distribution vector, whose length is equal to the number of edges of that node. Since both $\mathcal{G}$ and $\overline{\mathcal{G}}$ can have at most K edges per node, the runtime of generating a random curtain is $O(TK)$ (for both $\mathcal{G}$ and $\overline{\mathcal{G}}$). In practice, a large number of random curtains can be precomputed offline.

D. Network architectures and training details

In this section, we describe in detail the network architectures used by our main method, as well as various baseline models.

*1) **2D-CNN**:* The 2D-CNN architecture we use to forecast safety envelopes is shown in Fig. 8. It takes as input the previous k light curtain outputs. These consists of the intensities of the light curtain per camera ray $\mathbf{I}_{1:T}$, as well as the control points of the curtain that was placed i.e. $\mathbf{X}_{1:T}$. Each light curtain output $(\mathbf{X}_{1:T}, \mathbf{I}_{1:T})$ is converted into a *polar occupancy map*. A polar occupancy map is a $T \times L$ image, where the t-th column of the image corresponds to the camera ray R_t. Each ray is binned into L uniformly spaced locations; although L could be set to the number of control points per camera ray in the light curtain constraint graph, it is not required. Each column of the occupancy map has atmost one non-zero cell value. Given $\mathbf{X}_t, \mathbf{I}_t$, the cell on the t-th column that lies closest to $\mathbf{X}_t$ is assigned the value $\mathbf{I}_t$. We generate k such top-down polar occupancy maps encoding intensities. We generate k more such polar occupancy maps, but just assigning binary values to encode the control points of the light curtain. Finally, another polar occupancy map is generated using the forecast of the safety envelope from the handcrafted baseline policy. The $2k+1$ maps are fed as input to the 2D-CNN. We use $k = 5$ in all our experiments. The input is transformed through a sequence of 2D convolutions; the convolutional layers are arranged in a manner similar to

the the U-Net [23] architecture. This involves skip connections between downsampled and upsampled layers with the same spatial size. The output of the U-Net is a 2D image. The U-Net is a fully convolutional architecture, and the spatial size of the output is equal to the spatial size of the input. Column-wise soft-max is then applied to transform the output into T categorical probability distributions, one per column. We sample a cell from the t-th distribution, and the location of that cell in the top-down view is interpreted as the t-th control point. This produces a forecasted safety envelope.

*2) **1D-CNN**:* We use a 1D-CNN as a baseline network architecture. The 1D-CNN takes as input the previous k light curtain placements $\mathbf{X}_{1:T}$, and treats it as a 3-channel 1-D image (the three channels being the x-coordinate, the z-coordinate, and the range $\sqrt{x^2+z^2}$). It also takes the previous k intensity outputs $\mathbf{I}_{1:T}$, and treats them as 1-dimensional vectors. It also takes as input the forecasted safety envelope from the hand-crafted baseline. The overall input to the 1D-CNN is a $4k+1$ channel 1D-image. It applies a series of 1D fully-convolutional operations, with ReLU activations. The output is a 1-D vector of length T. These are treated as ranges on each light curtain camera ray, and are converted to the control points $\mathbf{X}_{1:T}$ of the forecasted safety envelope.

*3) **1D-GNN**:* We use a graph neural network as a baseline to perform safety envelope forecasting. The GNN takes as input the output of the previous two light curtain placements. The GNN contains $2T$ nodes, T nodes corresponding to each curtain. The graph contains two types of edges: vertical edges between corresponding nodes of the two curtains (T in number), and horizontal edges between nodes corresponding to adjacent rays of the same curtain ($2T-1$ in number). Each node gets exactly one feature: the intensity value of its corresponding curtain and camera ray. Each horizontal and vertical edge gets 3 input features: the differences in the $x, z, \sqrt{x^2+z^2}$ coordinates of the control points of the rays corresponding to the nodes the edge is connected to. Then, a series of graph convolutions are applied. The features after the final graph convolution, on the nodes corresponding to the most recent light curtain placement are treated as range values on each camera ray R_t. The t-th range value is converted to a control point $\mathbf{X}_t$ for camera ray R_t, and the GNN generates a forecast $\mathbf{X}_{1:T}$ of the safety envelope.

We find that providing the output of the hand-crafted baseline policy as input to the neural networks improves performance (compare the last two rows of Table I). We attribute this improvement to two reasons:

1) It helps *avoid local minima during training*: when training the neural networks without the handcrafted input, we observe that the networks quickly settle into local minima where the loss is unable to decrease significantly. This suggests that the input helps with training.
2) It *provides useful information to the network*: To determine if it is also useful after training is complete, we replace the handcrafted input with a constant value and find that this significantly deteriorates performances. This indicates that the 2D CNN continues to rely on the handcrafted inputs at test time.

E. Parallelized pipelining and runtime analysis

In this section, we describe the runtime of our approach. Our overall pipeline has three components: (1) *forecasting* the safety envelope, (2) *imaging* the forecasted and random light curtains, and (3) *processing* the light curtain images. Since these processes can be run independently, we implement them as parallel threads that run simultaneously. This is shown in Figure 9.

The imaging and processing threads run continuously at all times. If a forecasted curtain is available to be imaged, it is given priority and is scheduled for the next round of imaging. But if there are no forecasted curtain waiting to be imaged, random curtains are placed and processed. This scheduling leads to an overall latency of 75ms (13.33 Hz). Due to the parallelized implementation, we are able to place two random curtains during each cycle of our pipeline.

Figure 9 (*right*) shows a breakdown of the timing of the forecasting method method. It consists of the feed-forward pass of the 2D CNN, as well as other high-level processing tasks.

F. Results for the simulated environment without using random curtains

In this section, we include additional results corresponding to the "Ours w/o Random curtains" row of Table I. Table III contains results of other policies (handcrafted baseline, 1D-CNN baseline, 1D-GNN baseline) and ablation conditions (Ours w/o Forecasting, Ours w/o Baseline input) when random curtains are not used. The top half of Table III contains results without using random curtains. The bottom half contains results for the same policies using random curtains (this is essentially a copy of Table I, to aid with comparisions). We find that the conclusions of Table I still hold when random curtains are not used: our method still outperforms the baselines and removing any component of our method (not forecasting to the next timestep or removing the output of the hand-crafted policy as input) reduces performance.

G. Hardware specification of light curtains

In this section, we provide some details about the hardware specification of light curtains, as well as comparing it with the specifications of a Velodyne HDL-64E LiDAR.

The distance between the camera and the laser (the baseline of the device) is 20 cm. The maximum angular velocity of the galvanometer is 2.5×10^4 rad/sec and the maximum angular acceleration of the galvanometer is 1.5×10^7 rad/sec^2. The operating range of the light curtain device is up to 20 meters (daytime outdoors) and 50 or more meters (indoor or night time).

The following table compares the light curtain device with a Velodyne HDL-64E:

The LiDAR is limited to fixed scan patterns. Light curtains are designed to be programmable as long as the curtain profiles

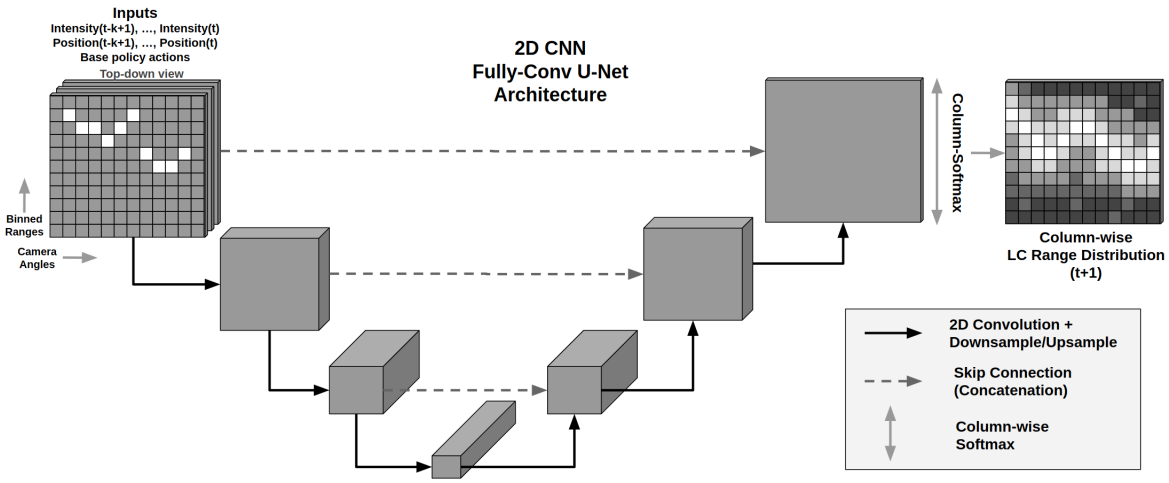


Fig. 8: The network architecture of the 2D CNN model used for safety envelope forecasting. It takes as input the previous k light curtain outputs, and converts them into top-down polar occupancy maps. Each column of the image is assigned to a camera ray, and each row is treated as a binned location. It also takes the prediction of the hand-crafted baseline as additional input. The input is transformed through a series of 2D convolution layers, arranged in a manner similar to the U-Net [23] architecture. This involves skip connections between downsampled and upsampled layers with the same spatial size. The output of the U-Net is a 2D image. This is a fully-convolutional architecture, and the spatial dimensions of the input and output are equal. Column-wise soft-max is then applied to the output to transform it to a probability distribution per column. A value $\mathbf{X}_t$ is sampled per column to produce the profile of the forecasted safety envelope.

	Random curtain	Huber loss ↓	RMSE Linear ↓	RMSE Log ↓	RMSE Log Scale-Inv. ↓	Absolute Relative Diff. ↓	Squared Relative Diff. ↓	Thresh (1.25) ↑	Thresh (1.25^2) ↑	Thresh (1.25^3) ↑
Handcrafted baseline	✗	0.1989	2.5811	0.2040	0.0904	0.2162	2.0308	0.6321	0.7321	0.7657
1D-CNN	✗	0.1522	2.3856	0.2176	0.1076	0.1750	0.9482	0.5842	0.7197	**0.7868**
1D-GNN	✗	0.1584	2.2114	0.1835	**0.0839**	0.1772	1.1999	0.6546	0.7381	0.7710
Ours w/o Forecasting	✗	0.1691	2.6047	0.2288	0.1158	0.1927	1.2555	0.6109	0.7114	0.7654
Ours w/o Baseline input	✗	0.1556	2.5987	0.2273	0.1135	0.1797	1.1063	0.6021	0.7094	0.7683
Ours	✗	**0.1220**	**2.0332**	**0.1724**	0.0888	**0.1411**	**0.9070**	**0.6752**	**0.7450**	0.7852
Handcrafted baseline	✓	0.1145	1.9279	0.1522	0.0721	0.1345	1.0731	0.6847	0.7765	0.8022
Random curtain only	✓	0.1484	2.2708	0.1953	0.0852	0.1698	1.2280	0.6066	0.7392	0.7860
1D-CNN	✓	0.0896	1.7124	0.1372	0.0731	0.1101	0.7076	0.7159	0.7900	0.8138
1D-GNN	✓	0.1074	1.6763	0.1377	0.0669	0.1256	0.8916	0.7081	0.7827	0.8037
Ours w/o Forecasting	✓	0.0960	1.7495	0.1428	0.0741	0.1163	0.6815	0.7010	0.7742	0.8024
Ours w/o Baseline input	✓	0.0949	1.8569	0.1600	0.0910	0.1148	0.7315	0.7082	0.7740	0.7967
Ours	✓	**0.0567**	**1.4574**	**0.1146**	**0.0655**	**0.0760**	**0.3662**	**0.7419**	**0.8035**	**0.8211**

TABLE III: Performance of safety envelope estimation on the SYNTHIA [34] urban driving dataset under various metrics, with and without using random curtains. Policies in the top half of the table were trained and evaluated without random curtains, while policies in the bottom half were trained and evaluated with random curtain placement.

	Light curtain	Velodyne HDL-64E LiDAR
Horizontal resolution	0.08°	0.08° - 0.35°
Vertical resolution	0.07°	0.4°
Rotation speed	60 Hz	5 Hz – 30 Hz
Cost	Less than $1000	Approx. $80,000

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

satisfy the velocity and acceleration limits. Note that the resolution of the light curtain is the same as the 2D camera used which can be significantly higher than any LIDAR. Our current prototype uses a camera with a resolution of 640×512.

H. Results for the real-world environment under high latency

The results for the real-world enviroment with walking pedestrians (see Table II of Section VII) were generated using the parallelized and efficient pipeline described in Appendix VIII-E. We now present some older results for the same environment that did not use the efficient implementation. Random curtains were not imaged and processed in parallel with the forecasting method. Instead, these operations were performed sequentially: we alternated between the forecasting step and placing a single random curtain. This increases the latency of the pipeline. A comparison between our method and

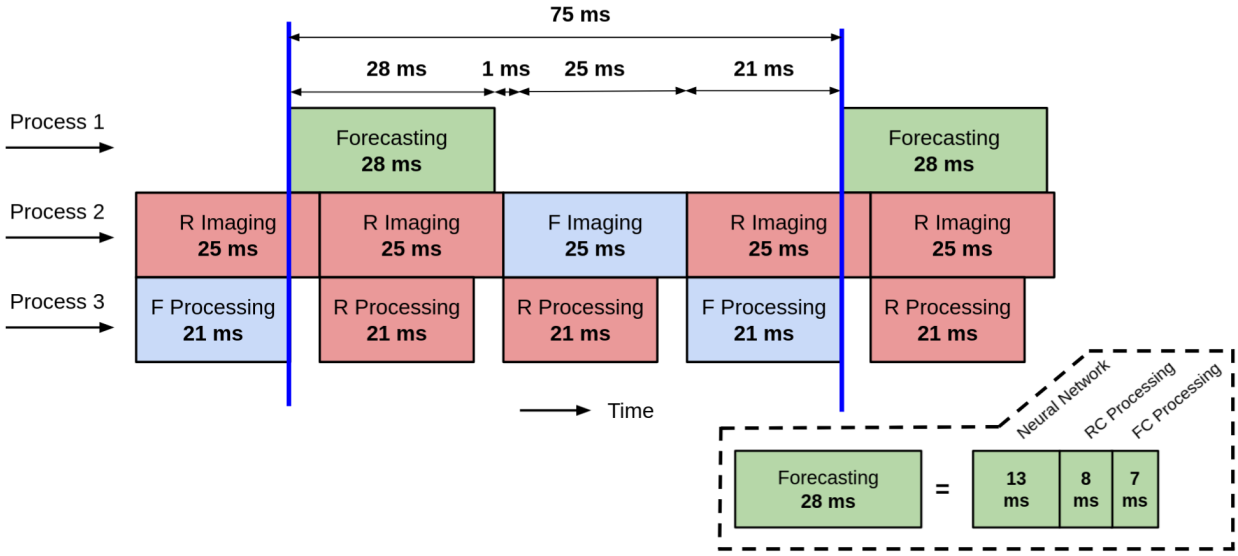


Fig. 9: Pipeline showing the runtime of the efficient, parallelized implementation of our method. The pipeline contains three processes running in parallel: (1) the method that forecasts the safety envelope, (2) imaging of the light curtains performed by the physical light curtain device, and (3) low-level processing of images. Here, "R" or "RC" stands for "random curtain" and "F" or "FC" stands for "forecasting curtain". *Bottom right:* the forecasting method further consists of running the feed-forward pass of the 2D CNN and high-level processing of the random and forecasting curtains. The overall latency of our pipeline is 75ms (13.33 Hz). We are able to place two light curtains in each cycle of the pipeline.

the handcrafted baseline when both use this slower implementation is shown in Table V. Our method is able to outperform the handcrafted baseline under various implementations with varying latencies.

	Huber loss	RMSE Linear	RMSE Log	RMSE Log Scale-Inv.	Absolute Relative Diff.	Squared Relative Diff.	Thresh (1.25)	Thresh (1.25^2)	Thresh (1.25^3)
	↓	↓	↓	↓	↓	↓	↑	↑	↑
Handcrafted baseline	0.07045	0.7501	0.1282	0.0886	0.1070	0.1072	0.8907	**0.9975**	**1.0000**
Ours	**0.0189**	**0.3556**	**0.0667**	**0.0443**	**0.0449**	**0.0271**	**0.9890**	0.9953	0.9976

TABLE V: Performance of safety envelope estimation in the real-world pedestrian environment under a high latency i.e. slower implementation.

References

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

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

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

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

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

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

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

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

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

[10] Joachim Denzler and Christopher M Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(2): 145–157, 2002. URL https://ieeexplore.ieee.org/abstract/document/982896.

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

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

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

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

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

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

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

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

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

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

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

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

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

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. URL https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28.

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

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

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

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

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

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

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

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

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

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

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