DISeR: Designing Imaging Systems with Reinforcement Learning

Tzofi Klinghoffer* Kushagra Tiwary* Nikhil Behari Bhavya Agrawalla Ramesh Raskar
Massachusetts Institute of Technology

Abstract

Imaging systems consist of cameras to encode visual information about the world and perception models to interpret this encoding. Cameras contain (1) illumination sources, (2) optical elements, and (3) sensors, while perception models use (4) algorithms. Directly searching over all combinations of these four building blocks to design an imaging system is challenging due to the size of the search space. Moreover, cameras and perception models are often designed independently, leading to sub-optimal task performance. In this paper, we formulate these four building blocks of imaging systems as a context-free grammar (CFG), which can be automatically searched over with a learned camera designer to jointly optimize the imaging system with task-specific perception models. By transforming the CFG to a state-action space, we then show how the camera designer can be implemented with reinforcement learning to intelligently search over the combinatorial space of possible imaging system configurations. We demonstrate our approach on two tasks, depth estimation and camera rig design for autonomous vehicles, showing that our method yields rigs that outperform industry-wide standards. We believe that our proposed approach is an important step towards automating imaging system design. Our project page is https://tzofi.github.io/diser.

1. Introduction

Cameras are ubiquitous across industries. In autonomous vehicles, camera rigs provide information on the ego-vehicle's surroundings so it can navigate; in biology, microscopy allows new viruses to be studied and vaccines to be developed; and in AR/VR systems, advanced headsets provide immersive reconstructions of the user's surroundings. In each of these applications, camera configurations must be carefully designed to capture relevant information for downstream tasks, often done with perception models (PMs). PMs are typically implemented as neural networks and use the output of cameras to predict information such as where other vehicles are on the road, what type of molecule is present in a biological sample, or where the user is located within a virtual environment. Yet, despite their interdependence, cameras and PMs are often designed independently.

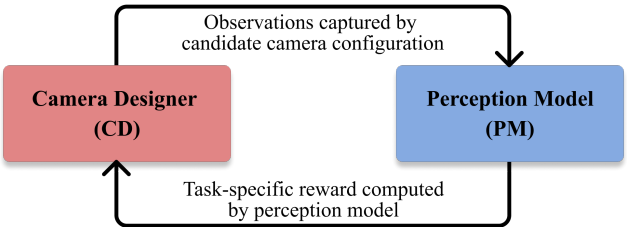

Figure 1: Overview: The camera designer selects imaging hardware candidates, which are used to capture observations in simulation. The perception model is then updated and computes the reward for the camera designer using the captured observations. In our paper, we implement the camera designer with reinforcement learning and the perception model with a neural network.

Designing camera systems is non-trivial due to the vast number of engineering decisions to be made. For example, consider designing a camera rig on an autonomous vehicle. Suppose the ego-vehicle is limited to up to 5 lidar sensors, 5 radars, and 5 RGB sensors, with 1,000 possible spatio-temporal resolutions. If there are 1,000 discrete candidate camera positions on the ego-vehicle, the search space expands to 10^8 different configurations. In practice, the search space can become many orders larger with more possibilities for each imaging system building block. Furthermore, because the search space is non-differentiable, there exists a need to develop efficient methods to effectively traverse the search space for an optimal imaging configuration.

In our paper, we propose using reinforcement learning (RL) to automate search over imaging systems. We first define a language for imaging system design using context-free grammar (CFG), which allows imaging systems to be represented as strings. The CFG serves as a search space for which search algorithms can then be used to automate imaging system design. We refer to such an algorithm as a camera designer (CD) and implement it with RL. RL allows us to search over imaging systems without relying on differentiable simulators and can scale to the combinatorially large search space of the CFG. Inspired by how animal eyes and brains are tightly integrated [28], our approach jointly trains

* Equal contribution.

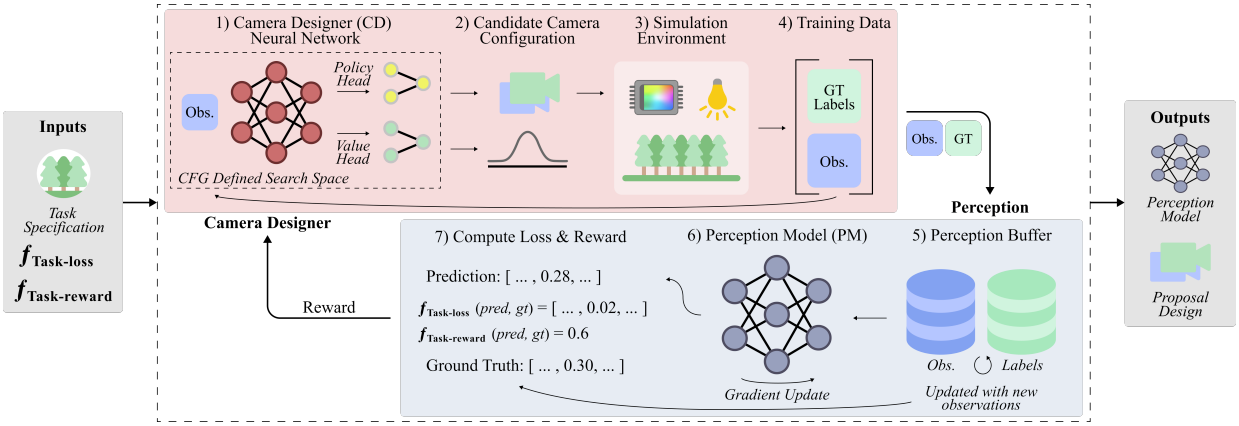

Figure 2: Approach: Our approach allows a camera configuration and perception model (PM) to be co-designed for task-specific imaging applications. At every step of the optimization, the camera designer (CD), implemented with reinforcement learning, proposes candidate camera configurations (1-2), which are used to capture observations and labels in a simulated environment (3-4). The observations and labels are added to the perception buffer (5) and used to compute the loss and reward, while the N most recent observations in the perception buffer are used to train the PM. The reward is propagated to the CD agent which proposes additional changes to the candidate camera configuration. After the episode terminates, the CD agent is trained using proximal policy optimization (PPO) [39] until convergence.

the CD and PM, using the accuracy of the PM to inform how the CD is updated in training (Fig. 1). Because searching over the entire CFG is infeasible with available simulators, we take the first step of *validating* that RL can be used to search over subsets of the CFG, including number of cameras, pose, field of view (FoV), and light intensity. First, we apply our method to depth estimation, demonstrating the viability of jointly learning imaging and perception. Next, we tackle the practical problem of designing a camera rig for AVs and show that our approach can create rigs that lead to higher perception accuracy than industry-standard rig designs. While AV camera rigs are one of many potential applications of our method, to the best of our knowledge, we are among the first to propose a way to optimize AV camera rigs. Our paper makes the following contributions:

- **Imaging CFG**: We introduce a context-free grammar (CFG) for imaging system design, which enumerates possible combinations of illumination, optics, sensors, and algorithms. The CFG can be used as a search space and theoretical framework for imaging system design.
- **Co-Design**: We demonstrate how task-specific camera configurations can be co-designed with the perception model by transforming the CFG into a state-action space and using reinforcement learning (Fig. 2). Our approach can converge despite the reward function being jointly trained with the policy and value functions.
- **Experimental Validation**: We demonstrate our method for co-design by applying it to (1) the task of depth estimation using stereo cues, and (2) optimizing camera rigs for autonomous vehicle perception, showing in both cases that camera configuration and perception model can be learned together.

2. Related Work

2.1. Joint Optimization of Optics & Algorithms

Our work is most closely related to end-to-end optimization of cameras, which is an area of research focused on jointly optimizing components of cameras together with an algorithm, typically a neural network. Instead of relying on heuristics, the goal of end-to-end optimization is to produce images that optimize the pertinent information required for the task. Existing work primarily focuses on optimizing the parameters of the optical element, sensor, and image signal processor of a single camera. Applications of end-to-end optimization include extended depth of field and superresolution imaging [41], high dynamic range (HDR) imaging [34, 42], demosaicking [9], depth estimation [2, 11, 22, 23], classification [10] and object detection [12, 36, 37]. Tseng *et al.* [43] employ gradient descent on a non-differentiable simulator by training a proxy neural network, whereas we directly operate on the non-differentiable simulator with RL. For a more comprehensive review of end-to-end optimization, we refer readers to [26]. In contrast to end-to-end optimization methods, we focus on optimizing over the much larger space of possible imaging system designs, rather than the parameters of an individual camera. Our search space contains varying illumination sources, optics, sensors, and algorithms, each with many parameters. Rather than using stochastic gradient descent for optimization, we use reinforcement learning, allowing our approach to be used with non-differentiable simulators.

2.2. Reinforcement Learning

Deep reinforcement learning (RL) has become widely used in recent years as a way to do sequential decision mak-

ing for a wide array of problems, such as protein folding [24], learning faster matrix multiplication [17], and automated machine learning [3]. Many RL techniques focus on the *exploration-exploitation* trade-off, where an agent must learn to balance exploring new states with exploiting previously visited states that lead to high reward. RL is also used for many combinatorial optimization problems [33]. In our work, we take inspiration from automated chip placement [35], which, like our approach, is formulated to allow an RL agent to place a new component at every step and select the placement of that component. Like many other problems RL has been applied to, imaging contains a high dimensional search space. In our work, we use proximal policy optimization (PPO) [39], which has been used for combinatorial search in past work [45].

Context-free grammars (CFGs) have been used to design machine learning (ML) pipelines, which are combinations of data-flows, ML operators, and optimizers [15][32][25]. Typically, ML pipeline design is done via a search over strings in the CFG using tree search algorithms, such as Monte Carlo tree search or upper confidence trees [27] [44]. CFGs have also been adopted for robot design [46], molecule generation [20], and material design [19]. We use CFG to functionally represent imaging systems as combinations of illumination, sensors, optics and algorithms such that the output string describes a camera configuration and perception model that can be used to solve a desired task.

3. Automated Imaging System Design

3.1. Language for Imaging

We define the configuration space of imaging systems using context-free grammar (CFG) as it allows for a flexible configuration space that can be searched. A typical context-free grammar, G, is represented as a tuple, $G = (V, \Sigma, P, R)$, where V corresponds to non-terminal symbols in the grammar, Σ corresponds to terminal symbols, P corresponds to the production rules, and R is the start symbol. The goal of our proposed CFG is to allow the construction of strings to represent arbitrarily complex imaging systems, which usually consist of illumination sources, optical elements, sensors to convert light into digital signals, and algorithms that decode the scene. For example, consider the task of depth estimation that can be done in numerous ways. One solution is depth from stereo, which involves placing two cameras, c_1, c_2, in the scene at points, p_1, p_2, with some baseline. Each camera has an optical element, $o_1 = (f,d)$, with a focal length, f, and aperture, d, and a sensor, $s_1 = ((h,w),t)$, with spatial and temporal resolutions, (h,w) and t, respectively. Thus the cameras can be expressed as $c_1 = (o_1, s_1)$ and $c_2 = (o_2, s_2)$. An algorithm can decode the outputs of the two cameras to produce depth, and can be implemented with correspondence-matching [6],

$$\mathrm{R} \rightarrow \mathrm{X}\mathcal{S}\mathrm{XA} \tag{1}$$
$$\mathrm{X} \rightarrow \mathcal{I}\mathrm{X}|\mathrm{O}\mathcal{S}\mathrm{X}|\mathrm{A}_2\mathrm{X}|\epsilon \tag{2}$$
$$\mathrm{O} \rightarrow \mathcal{O}\mathrm{O}|\epsilon \tag{3}$$
$$\mathrm{A}_1 \rightarrow \mathcal{A}_1\mathrm{A}_1|\mathcal{A}_1 \tag{4}$$
$$\mathrm{A}_2 \rightarrow \mathcal{A}_2\mathrm{O}\mathcal{S}|\mathcal{A}_2\mathcal{I}|\mathcal{A}_2\mathcal{S}|\epsilon \tag{5}$$
$$\mathcal{S} := \{s_p s_{hw} s_t s_\lambda s_q\}_{p\in\mathbb{R}^6, h,w,t,q\in\mathbb{Z}} \tag{6}$$
$$\mathcal{O} := \{o_f o_d\}_{f\in\mathbb{R}, D\in\mathbb{Z}} \tag{7}$$
$$\mathcal{I} := \{i_p i_i\}_{p\in\mathbb{R}^6, i\in\mathbb{Z}} \tag{8}$$
$$\mathcal{A}_1 := \{a_{nn}, a_{fourier}, ...\} \tag{9}$$
$$\mathcal{A}_2 := \{\text{autofocus}, ...\} \tag{10}$$

Figure 3: Context-free grammar (CFG) for imaging: Production rules (1-5) and alphabets (6-10) for our proposed CFG for designing imaging systems. R is the starting symbol from which a design starts. All imaging systems must have at least one sensor, $\mathcal{S}$, and one algorithm, $\mathcal{A}$. The grammar allows arbitrary physically plausible combinations of illumination ($\mathcal{I}$) optics ($\mathcal{O}$), sensors ($\mathcal{S}$), and algorithms ($\mathcal{A}$), each defined in their respective alphabet above. A_1 refers to algorithms that process the output of hardware, while A_2 refers to algorithms that control hardware.

(a_{st}), or deep stereo [31], (a_{ds}). The full system can be described as a string, s_1 = "$c_1 c_2 a_{st}$" or s_2 = "$c_1 c_2 a_{ds}$". Another way to estimate depth is with active illumination or time-of-flight (ToF) imaging. We can represent lidar as an algorithm, a_{control}, that illuminates the scene at the same point with a laser, l_1, and ToF sensor, s_{ToF}. We can describe this system as $s_{\text{lidar}} = a_{\text{control}} l_1 s_{ToF} a_{ToF}$. These examples illustrate how CFG can represent imaging systems with different illumination, optics, sensors, and algorithms as strings. The goal of the proposed CFG is not to describe how the individual components of an imaging system are made, e.g. their electronics, but rather to describe the function of each component. Next, we define the grammar's alphabet and production rules.

Grammar. Our proposed CFG can be stated as $G = (V, \Sigma, P, R)$. We define the variables as $V = \{\mathrm{X}, \mathrm{O}, \mathrm{A}_1, \mathrm{A}_2\}$, each defined in the following sections, and the terminals, Σ, which we refer to as alphabets, as $\Sigma = \{\mathcal{I}, \mathcal{O}, \mathcal{S}, \mathcal{A}_1, \mathcal{A}_2\}$, where $\{\mathcal{I}\}$ is illumination, $\{\mathcal{O}\}$ is optics, $\{\mathcal{S}\}$ is sensors, and $\{\mathcal{A}_1\}$ and $\{\mathcal{A}_2\}$ are algorithms. Each alphabet contains possible components and parameters, defined in lower case, e.g. a_{nn}. Each component within an alphabet is parameterized by its functionality, e.g. focal length, rather than an off-the-shelf component. We describe each alphabet below and in Fig. 3.

Illumination. The illumination alphabet, $\mathcal{I}$, functionally represents different types of possible illuminations. In imaging, illumination can be represented with many parameters, such as duration (d), intensity (i), color, wavelength (λ), polarization η, pose (position & orientation), (p) and modulation in space and time [5]. In the scope of this work,

we consider pose and intensity. These can later be extended to other forms of illumination.

Optics. We define the optics alphabet, $\mathcal{O}$, to capture the most important (but not exhaustive) optical properties in an imaging system: focal length (f) and aperture (D). The optics alphabet can be extended to include more complex techniques such as phase masks or diffractive optical elements (DOE). The non-terminal O indicates that optical elements can be stacked to create a multi-lens system.

Sensors. The sensor alphabet, $\{\mathcal{S}\}$, functionally describes different types of sensors, such as RGB and SPAD. We parameterize a sensor by its pose s_p, spatial (or angular) resolution s_{hw}, temporal resolution s_t, bit quantization s_q and wavelength s_λ. For example, a SPAD sensor has higher temporal resolution (picosecond scale) and generally lower spatial resolution (on the order of 1,000 to 100,000 pixels), while a typical RGB sensor (CMOS) has a higher spatial resolution (hundreds of megapixels), but a lower temporal resolution (30 fps). Similarly, quantization (for example) can be varied between 1, 8 or 12 bits. The pose is the position (x,y,z) and the orientation (pitch, yaw, and roll) of the sensor in 3D space, $s_p \in \mathbb{R}^6$.

Algorithms. Algorithms are needed to decode raw images and control other alphabets. We denote the alphabet for algorithms with two sets: $\{\mathcal{A}_1, \mathcal{A}_2\}$. $\mathcal{A}_2$ is the set of algorithms that affect subsequent illumination, optics, and sensors (e.g. autofocus, controlling where to shine illumination), whereas $\mathcal{A}_1$ are algorithms that decode the incoming data from the sensors for a given task. These algorithms include standard imaging operators, such as the Fourier transform, backprojection, Radon transform, Gerchberg-Saxton algorithm, photometric stereo, and more. Additionally, $\mathcal{A}_1$ includes neural networks, which can perform detection, classification, etc. Due to the production rule, $\mathrm{A} \rightarrow \mathcal{A}_1\mathrm{A}|\mathcal{A}_1$, $\mathcal{A}_1$ can be repeated and stacked together. For example, an algorithm can be designed that takes the Fourier transform of the input data and feeds it through a multilayer perceptron (MLP).

Production Rules. We define a set of production rules, shown in Fig. 3, that can produce strings representing possible imaging system configurations. In our formulation, every imaging system includes at least one sensor and algorithm. The X accounts for imaging systems with different illumination, optics and sensors. In all cases, the string must end with at least one algorithm that outputs the desired task. Additionally, each $\mathcal{A}_2$ also requires an illumination, optics component, or sensor that it controls. The production rules account for multiple sensors and illuminations that illuminate and sense different parts of the scene.

3.2. Imaging Design with Reinforcement Learning

The proposed context-free grammar (CFG) defines ways of combining illumination, optics, sensors, and algorithms to form an imaging system. The goal of our work is to automate imaging system design by searching over the CFG. Because the output of the cameras in the imaging system must be well suited for a specific, downstream task, we co-design them with the task-specific perception model (PM). We next propose using a learned camera designer (CD) to automatically search over the CFG. We implement the CD with reinforcement learning (RL) because (1) the combination of continuous variables in our CFG causes an explosion in the search space, which, as a result, makes search with methods such as Monte Carlo tree search (MCTS) [7] or alpha-beta search [38] intractable, and (2) many advanced imaging simulators are not differentiable [18, 16, 21], and thus gradient descent cannot be directly applied. Our problem is well suited for sequential decision making because the task performance achieved with each choice of camera configurations directly affects subsequent design choices.

Overview: Our approach is illustrated in Fig. 2. The input is a task-specific loss and reward function. When optimization starts, the imaging system contains no hardware. At each step, the CD selects whether to add a component into the system and the component's parameters (Fig. 2a-b). A simulator can then be used to collect observations from the candidate camera configuration (Fig. 2c). These observations are used by the perception model to compute the reward and loss (Fig. 24-7). The reward is used to train the CD and the loss is used to train the perception model. This loop repeats until a camera configuration and perception model have been created that maximize task accuracy.

RL Formulation: We transform the CFG into a state-action space which the RL agent, henceforth referred to as the CD, can search over. We use proximal policy optimization (PPO) to train the CD and model the RL problem with the following states, actions, and rewards:

- states, S: the possible states of the world, which, in our case, are the possible enumerations of illumination, optics, and sensors, and possible observations that can be captured from each enumeration.
- actions, A: the actions an agent can take at any step, which, our case, consist of choosing illumination, optics, sensors, algorithms, and all parameters.
- reward, R: the reward for taking an action in a state, which, in our case, is computed by passing observations from the candidate camera configuration into the PM to compute accuracy for a target task.

Simulation & Environment: Unlike standard RL problems where the agent acts based on observations from a fixed sensor, the observations provided to the CD can change, meaning the CD has to learn how to act with varying input (e.g. varying numbers of images, sensor parameters, etc). The simulator should thus be able to render

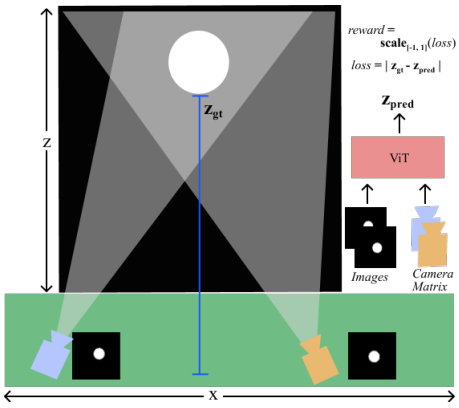

Figure 4: Depth from Stereo Setup: The goal of this experiment is to estimate the depth of a sphere using stereo cues. The camera designer (CD) places up to C cameras within the green box. Camera poses and images are input to the perception model (PM) which outputs a predicted depth. We render environments that are devoid of monocular cues to force (1) the CD to learn to obtain multi-view cues and (2) the PM to learn to exploit these cues.

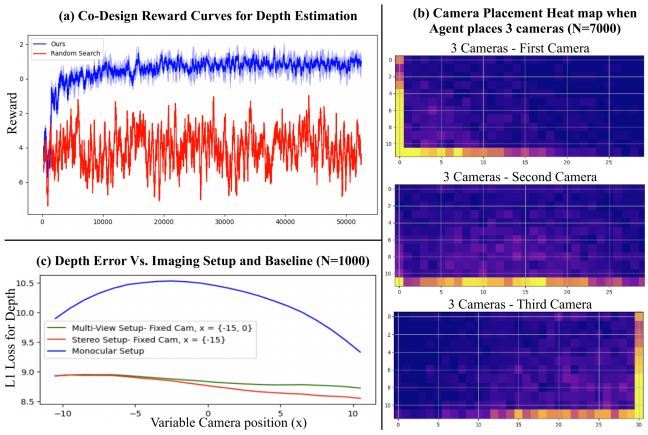

Figure 5: Joint Camera and Perception Design for Stereo Depth. We train the CD and PM from scratch to estimate depth of a sphere. (a) Our reward function consistently improves, even though it constantly changes due to the PM concurrently training with the CD. (b) The CD learns to maximize the baseline between different cameras over the course of 1000 experiments when placing 3 cameras. (c) The loss decreases with more placed cameras and larger distances between the cameras, which shows that the PM learns to exploit multi-view cues.

data from all potential imaging systems that can be derived from the CFG. Because simulators that encompass the entire CFG are not available, we search over subsets of the CFG to validate our method. While we use a simulator, a dataset can also be used with offline RL approaches [29].

Perception Model: In our experiments, we set the algorithm, $\mathcal{A}$, to be a trainable neural network (NN). The NN's role is to produce a task prediction given arbitrary observations from candidate camera configurations. The NN must be able to map a varying input (number of observations, modality, etc.) to a fixed output. For example, the CD may increase the number of sensors in the system beyond one, leading to multiple observations. We propose using transformers to mitigate this problem since they map a dynamic number of observations to a fixed-size feature embedding by converting inputs into sequences of patches [40]. To reduce noise in the gradients when jointly training the PM with the CD, we propose a perception buffer (Fig. 2.5), which stores the previous N observations from candidate camera configurations, allowing the PM to be trained over all data in the buffer at each step.

4. Experiments and Results

Overview: We apply our method to two problems, both of which exercise a subset of our proposed CFG to validate DISeR. First, we show how DISeR can jointly learn a camera configuration and perception model to solve depth estimation. Second, we apply DISeR to a practical engineering problem of designing camera rigs for AVs. The same formulation is used in both problems: at each step of optimization, the CD chooses whether to add a camera to the imaging system by predicting an action, p, in $[0, 1]$, referred to as camera placement probability, along with camera parameters. When p is greater than a threshold of 0.5, a camera is added with the predicted parameters. The camera parameters for each problem are shared in the sections below. In both problems, we compare our approach against random search, which we note is often very difficult to beat [4] [48].

4.1. Stereo Depth Estimation

4.1.1 Experimental Setup

Environment The goal of the first experiment is to estimate the depth of a sphere using stereo cues. The CD is allowed to place a maximum of C cameras in the scene (though it can also place fewer cameras). In theory, the CD could place a single camera and learn monocular cues (e.g. shading/lighting, texture, linear perspective). However, we simulate an environment where monocular cues are unavailable, making monocular depth estimation ill-posed.

Our environment consists of a randomly placed white sphere with a random radius, as shown in Fig. 4. We use PyRedner [30] to render images. The sphere position and radius are randomly sampled per episode from $(r, x, z) = \{r \in [3,9], x \in [-10, 10], z \in [1, 60]\}$. The depth is the z distance from the sphere to the average position of the placed cameras. The scene is illuminated such that shading cues and the position of the light source are absent as cues. The only feedback that the PM and CD receive is a loss between the predicted and ground truth depth. The goal of rendering such an environment is to determine whether the CD can adapt to the context and realize that only a multi-

view system can estimate depth. In parallel, the PM learns to exploit multi-view stereo cues. We show the supervised results of this experiment for validation in the supplement.

Action Space: The action space for depth estimation is $(p, x, z, \theta) = \{p \in [0,1], x \in [-15, 15], z \in [69, 80], \theta \in [-60°, 60°]\}$, where p is camera placement probability, (x, z) is location (see Fig. 4) and θ is yaw. FoV is $45°$.

Experiment Details: We use a modified version of the vision transformer (ViT) architecture [13] [1] that accepts an arbitrary number of images of fixed resolution and their corresponding camera parameters as input, and outputs a scalar depth. The spatial resolution is fixed to $(128, 128)$. The maximum number of cameras the CD can place is set to $C = 5$. The CD's PPO backbone and the perception model share the same network architecture and are initialized randomly. The reward is computed before updating the perception model and is re-scaled to $[-1, 1]$. Additional information about the training is provided in the supplement.

4.1.2 Results and Discussion

We evaluate the joint training (Fig. 5a), the learned policy (Fig. 5b), and the perception model (Fig. 5c) in isolation. Fig. 5a illustrates how our system maximizes reward when co-designing the PM with the camera design. The reward function is dictated by the output of the PM, but the PM is concurrently training with the camera design, which results in inconsistent rewards during training for the same states. In spite of this fact, our model is able to consistently increase the reward, even at the beginning of training when the PM is untrained and with random initialization. Our results show that the CD and PM are able to learn intuitions that hold true in conventional multi-view stereo.

Strategy #1 – Maximize Coverage: When given the option to place up to 5 cameras, the CD places 1 camera 7.6% of the time and 2, 3, 4, and 5 cameras 27.7%, 36.6%, 22.7%, 5.4% times, respectively. Fig. 5b shows the heatmaps of where the CD decides to place each camera, specifically when the CD chose to place exactly three cameras. The heatmaps denote the number of times the CD placed the camera at a particular location over the course of 7000 experiments, where each experiment denotes the placement of a new random size sphere at a random location. From the heatmaps, we see that the CD strategically placed the cameras at locations that maximize the baselines between different cameras. Camera 1 was predominantly placed in the left side of the allowed region, camera 2 at the center bottom, and camera 3 at the right. From these results, we see that the CD optimizes to place more cameras spaced far apart. However, placing more cameras doesn't necessarily mean that the CD is obtaining multiple views of the object (e.g. some cameras may be pointed in the opposite direction of the object). Therefore, we account for this case by defining the metric of *coverage*, which defines the number of cameras that have at least one pixel viewing the object. The CD policy learns a configuration which maximizes coverage of the allowed region. We find that performance improves as coverage increases from 0 to 3, with the L1 loss being 14.0, 9.2, 7.2, and 5.7 as the coverage increases. Coverage is discussed in detail in the supplementary.

Strategy #2 – Multi-View Cues and Maximal Baseline: Fig. 5c shows that the PM learns to exploit stereo cues when presented with multiple images. The experiment shown here compares the PM performance on a one-camera, two-camera, and three-camera system when estimating the depth of a sphere (averaged over 1000 different spheres of varying size and depth). All three systems have a camera that can be moved along the x axis, the two- and three-camera system have a fixed camera at $x = -15$, and the three-camera system has an additional fixed camera at $x = 0$. The blue curve illustrates the L1 loss between the ground truth and one-camera system predictions. The red and green curves illustrate the performance of the two-camera and three-camera system respectively. The three-camera system performs slightly better than the two-camera system, and both perform significantly better than the one-camera system. The multi-view systems also see a decrease in loss (and variance) as the baseline between the cameras increases (i.e. as the movable camera moves along the $+x$ axis). These curves indicate that the PM has learned similar wisdom to that of conventional stereo – multiple views with a large baseline enable better depth estimation [5]. While gradient descent could also be used to learn to maximize baseline given a differentiable simulator, we use RL, which can be used with non-differentiable simulators to search over both number of cameras and their baseline.

Searching Illumination: We also repeated the above experiment with an expanded action space that includes angle and intensity of a single spot light at a fixed position. To estimate the depth of the sphere, the CD must learn to sweep the light over the scene with a sufficiently high intensity until the sphere is illuminated. At each step, angle and intensity can be changed within the bounds of $[-60°, 60°]$ and $[0, 1]$, respectively, where 0 leaves the scene dark and 1 illuminates it. We found that the CD learns to increase the intensity so the sphere can be illuminated and change the angle such that the number of illuminated pixels on the sphere consistently increases over the episode.

4.2. Camera Rigs for Autonomous Vehicles

Next, we describe how our method can be used to optimize an AV camera rig for the perception task of bird's eye view (BEV) segmentation by jointly training the CD and PM. We validate our approach with three sets of experiments, described in the Experiment Details section be-

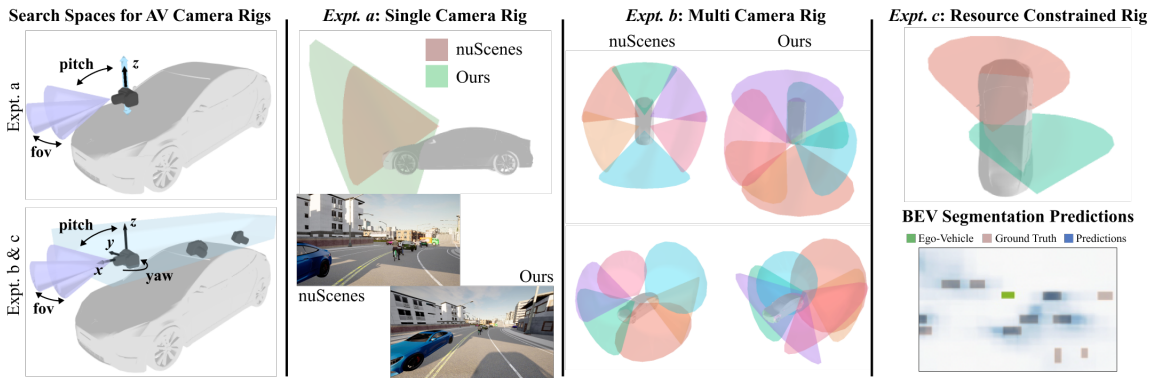

Figure 6: Autonomous Vehicle (AV) Camera Rig Task & Results: We demonstrate that our approach can be used to create AV camera rigs that are optimized for BEV segmentation. (Left) Our search space is shown – in expt. a, we optimize the height, pitch, and FoV of a single camera rig, while in expt. b and c, we optimize # cameras, x, y, z, pitch, yaw, and FoV. Results for each experiment are shown and we compare the optimized camera rig to the camera rig used in nuScenes [8]. In expt. c, the camera designer learns to place fewer cameras in only the direction where cars are placed. We also show the BEV segmentation predictions of our jointly trained perception model.

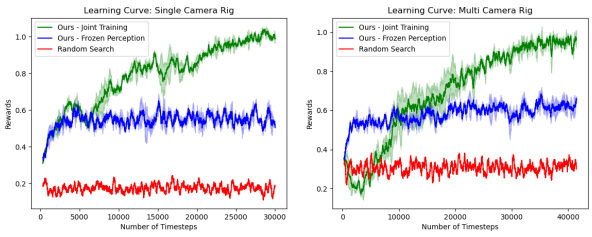

Figure 7: Results for AV Camera Rig Co-Design: Shown are the reward curves for the CD optimizing camera rigs for BEV segmentation. Reward is intersection over union (IoU). To demonstrate the effectiveness of co-designing the camera configuration with the perception model (PM), we show results when the PM is pre-trained and frozen (blue) vs. pre-trained and fine-tuned (green). Compared to random search (red), where actions are uniformly sampled from a random distribution at each step, our approach significantly outperforms, and discovers camera rigs that increase BEV segmentation IoU.

low. We find that the rigs created with our approach lead to higher BEV segmentation accuracy in our environment compared to the industry-standard nuScenes [8] rig. Our camera rig search space and results are visualized in Fig. 6.

4.2.1 Experimental Setup

Environment: We use the CARLA Simulator [14] to render observations from candidate camera rigs selected by the CD during training. For every camera on the candidate rig, the environment returns images, extrinsics, intrinsics, and 3D bounding box labels of vehicles in the scene. The 3D bounding boxes are used to compute the reward (for training the CD) and loss (for fine-tuning the PM). We use the same CARLA environment to create 25,000 samples rendered from randomly generated camera rigs to pre-train the PM for the task of BEV segmentation.

	IoU (Expt. a)	IoU (Expt. b)
Random Rig	0.254	0.084
nuScenes Rig	0.267	0.355
Our Rig	**0.341**	**0.427**

Table 1: We compare the BEV segmentation IoU for models trained and tested with a random rig, nuScenes rig, and our approach's rig. CARLA train and test scenes are the same for each. Our rig achieves higher performance than industry standards.

Action Space: The action space for AV camera rig design is $(p,x,y,z,\theta,\beta,\lambda) = \{p \in [0,1], x \in \eta_x, y \in \eta_y, z \in [z_{max}, z_{max} + 0.5m], \theta \in [-180°, 180°), \beta \in [-20°, 20°], \lambda \in [50°, 120°]\}$, where p is the camera placement probability, (x,y,z) is location, θ is yaw, β is pitch, and λ is FoV. η_x and η_y are the extents of the ego-vehicle in x and y, respectively, and z_{max} is the height of the ego-vehicle, meaning cameras can be placed anywhere within 0.5 meters (m) above the ego-vehicle. This action space conforms with rooftop rigs used in industry and the size and height of the roof match the Renault Zoe from nuScenes.

Experiment Details: We use a recent BEV segmentation model, Cross View Transformers (CVT) [47], as the PM. It is first pre-trained on a dataset containing randomly placed cameras to allow it to more easily generalize to all candidate camera rigs that the CD may select. We then use the pre-trained CVT model to initialize the PM and CD's PPO backbone. Finally, we train the CD to optimize camera rigs. The PM uses the observations from each candidate rig and 3D bounding box labels to compute a reward (IoU) and loss (binary segmentation loss). The reward is used to update the candidate camera configuration, while the loss is used to update the PM. We conduct three sets of experiments, one with a single camera rig (expt. a), one with a multi-camera rig

(expt. b), and one with a custom scenario and penalty for placing many cameras (expt. c). Each experiment is conducted with a frozen and a jointly trained PM. Fig. 7 shows that joint training leads to higher rewards (IoU).

- **Expt. a:** The CD exercises a limited action space, including only (p,z,β,λ) for a single camera on the front of the ego-vehicle. We use the same formulation as described above, but, at each step, if the CD places a camera, the previous camera is *replaced*, rather than the new one being added on the rig. After six steps, the episode terminates.
- **Expt. b:** The CD exercises the full action space, including $(p,x,y,z,\theta,\beta,\lambda)$. Cameras are placed within a bounding box on top of the ego-vehicle, as shown in Fig. 6. For comparison with the nuScenes rig, which has six cameras, we set the episode length to six, so at most six cameras can be placed.
- **Expt. c:** This experiment includes two modifications to Expt. b. First, a penalty is enforced each time a camera is added to the rig to disincentivize the CD from placing unnecessary cameras. Second, the distribution of vehicles during training is changed to only be in front of the ego-vehicle to demonstrate that the CD can customize its rig design to specific scenarios.

We collect data on a Tesla Model 3 (TM3) since Renault Zoe (RZ) is not available in CARLA (placing cameras within the bounds of the RZ roof). Since TM3 is slightly smaller than RZ, this does not significantly affect what the cameras see. Our approach is flexible and the action space can be changed or other constraints added per requirements.

Evaluation Protocol: After training, we use the following protocol to evaluate the quality of the CD-optimized camera rigs. First, we test the CD over 100 episodes, saving the candidate camera rig and sum of rewards at the end of each episode. We then select the top 20 rigs based on their sum of rewards. We fix these rigs and evaluate them over more episodes (20), again recording their sum of rewards. We sort the top twenty rigs by their sum of rewards and select the rig with the top reward, which we call the selected rig.

We test the efficacy of the selected rig by comparing its BEV performance to that of the nuScenes [8] rig. To compare BEV performance, we collect 25,000 training images and 5,000 test images in CARLA using both our selected rig and the nuScenes rig. We do this by deploying both rigs on a Tesla Model 3. Next, we train one BEV segmentation model for each rig, using the collected training data. Finally, we test both BEV segmentation models on the corresponding test dataset captured from that rig. By collecting train and test data in the exact same CARLA scenes, we ensure a fair comparison. The test IoU then serves as a final measure of the selected rig's utility for BEV segmentation.

4.2.2 Results and Discussion

As shown in Fig. 7, the CD significantly outperforms random search, and we observe that the rewards consistently increase over time across experiments. While using the pre-trained, frozen PM allows the CD to create camera configurations that increase BEV segmentation accuracy, jointly training the PM and CD together yields the best results. We note that the pre-trained CVT model (before RL) has 9% and 11% IoU for expts a and b, respectively, due to the challenging nature of fitting across many rigs. This IoU is improved during joint training. Using the above evaluation protocol for our CD, we find rigs created with the CD, for both experiments a and b, outperform the nuScenes rig on the task of BEV segmentation in our CARLA environment, as shown in Table 1. The top rigs for each experiment, example images from our rig vs. nuScenes, and a PM prediction are shown in Fig. 6. In expt. b, the CD can place up to the number of cameras in nuScenes (six). We find that the created rigs conform with AV conventions in several ways, such as distributing views around the ego-vehicle and using varying FoVs. While AV rigs, such as nuScenes, are well-engineered, our method suggests it may be possible to further improve them for specific tasks and environments.

In expt. b, the CD learns to place the maximum number of cameras (six) on the rig since there is no penalty for placing additional cameras. However, in many cases, AV companies may want to reduce rig cost and inference time by using fewer cameras. Different camera rigs may also be better suited to different AV scenarios. We test whether the CD can take both of these considerations into account in expt. c by only placing cars in front of the ego-vehicle and enforcing a penalty to the reward each time an additional camera is added to the rig. As a result, we find that the CD places fewer cameras and places them facing forward, as shown in Fig. 7. This result demonstrates that our approach can be used to build resource limited imaging systems that are well suited for specific test scenarios.

Strategy #1 - Camera Placement: Across experiments, we observe that the CD consistently learned two behaviors that lead to increased performance: (1) maximize camera height to 0.5 m above the ego-vehicle, and (2) reduce camera pitch to $-20°$. Maximizing camera height reduces the number of occlusions, thus leading to more ground truth pixels, and potentially incentivizing this behavior. However, we note the average number of 3D bounding box labels across both test sets is the same, suggesting occlusions do not incentivize higher camera placement and BEV segmentation performance is naturally improved with higher camera positions. The negative pitch could mean the CD has learned to prioritize detecting nearby cars, perhaps because the perception model has higher confidence of those predictions. We also observe that all vehicles in the scene are still visi-

ble with a -20° pitch, and only the sky is cropped, thus the CD reduces the number of uninformative pixels, while maximizing the number of pixels on the road. Finally, we find that two front-facing cameras are placed at the rear of the vehicle in expt. b. We ablate this placement by re-training the PM with both cameras moved to the front. IoU is 5% better when the cameras are in the back, perhaps since, by placing the front-facing cameras at the back, both the front and sides of the ego-vehicle are visible in captured images.

Strategy #2 - FoV vs Object Resolution: In expt. a, we found the FoV was always maximized by the CD, which makes sense because it allows the CD to obtain higher reward when more vehicles in the scene are visible. In expt. b, the FoV of the target rig varies between 85° and 120°, suggesting when all of the scene is visible (as is the case in expt. b due to the CD learning to distribute the camera yaws in all directions), FoV is less important or that the CD may have learned a tradeoff between FoV and object resolution.

Limitations: We demonstrate the CD on a limited number of scenarios within CARLA and focus only on the task of BEV segmentation of vehicles. In the future, our approach can be applied to more scenarios, tasks, and object classes. In addition, our experiments are done in simulation only. That said, our method has a direct path to real-world use. It can be used by AV companies to design rigs using their own simulator and requirements; those rigs can then be deployed on test cars to collect data. As AV simulators improve, we expect any gap in rig utility in sim vs real to fall.

5. Conclusion

Our paper proposes a novel method to co-design camera configurations with perception models (PMs) for task-specific applications. We define a context-free grammar (CFG) that serves as a search space and theoretical framework for imaging system design. We then propose a camera designer (CD) that uses reinforcement learning to co-learn a camera configuration and PM for the proposed task by transforming the CFG into a state-action space. The PM is jointly trained with the CD and predicts the task output, which is used to compute the PM loss and reward for the CD to propose better candidate camera configurations. We demonstrate our method for co-design by applying it to (1) depth estimation using stereo cues, and (2) optimizing camera rigs for autonomous vehicle perception. We show in both cases that CD and PM can be learned together. Our co-design framework shows that camera configurations and perception models are closely linked and task-specific optimal designs that outperform human designs can be searched for computationally.

Acknowledgements: We would like to thank Siddharth Somasundaram for his diligent proofreading of the paper. KT was supported by the SMART Contract IARPA Grant #2021-20111000004. We would also like to thank Systems & Technology Research (STR).

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826, 2021. 6

[2] Seung-Hwan Baek and Felix Heide. Polka lines: Learning structured illumination and reconstruction for active stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5757–5767, 2021. 2

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*. 3

[4] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. 5

[5] Ayush Bhandari, Achuta Kadambi, and Ramesh Raskar. *Computational Imaging*. MIT Press, 2022. 3, 6

[6] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000. 3

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012. 4

[8] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 7, 8

[9] Ayan Chakrabarti. Learning sensor multiplexing design through back-propagation. *Advances in Neural Information Processing Systems*, 29, 2016. 2

[10] Julie Chang, Vincent Sitzmann, Xiong Dun, Wolfgang Heidrich, and Gordon Wetzstein. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific reports*, 8(1):1–10, 2018. 2

[11] Julie Chang and Gordon Wetzstein. Deep optics for monocular depth estimation and 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10193–10202, 2019. 2

[12] Philipp Del Hougne, Mohammadreza F Imani, Aaron V Diebold, Roarke Horstmeyer, and David R Smith. Learned integrated sensing pipeline: Reconfigurable metasurface transceivers as trainable physical layer in an artificial neural network. *Advanced Science*, 7(3):1901913, 2020. 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner,

Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 7

[15] Iddo Drori, Yamuna Krishnamurthy, Raoni Lourenço, Rémi Rampin, Kyunghyun Cho, Cláudio T. Silva, and Juliana Freire. Automatic machine learning by pipeline synthesis using model-based reinforcement learning and a grammar. *CoRR*, abs/1905.10345, 2019. 3

[16] Epic Games. Unreal engine. 4

[17] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. 3

[18] Joseph M Geary. *Introduction to lens design: with practical ZEMAX examples*. Willmann-Bell Richmond, VA, USA:, 2002. 4

[19] Minghao Guo, Wan Shou, Liane Makatura, Timothy Erps, Michael Foshey, and Wojciech Matusik. Polygrammar: grammar for digital polymer representation and generation. *Advanced Science*, 9(23):2101864, 2022. 3

[20] Minghao Guo, Veronika Thost, Beichen Li, Payel Das, Jie Chen, and Wojciech Matusik. Data-efficient graph grammar learning for molecular generation. *arXiv preprint arXiv:2203.08031*, 2022. 3

[21] John K Haas. A history of the unity game engine. 2014. 4

[22] Harel Haim, Shay Elmalem, Raja Giryes, Alex M Bronstein, and Emanuel Marom. Depth estimation from a single image using deep learned phase coded mask. *IEEE Transactions on Computational Imaging*, 4(3):298–310, 2018. 2

[23] Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9):4676–4689, 2018. 2

[24] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 3

[25] Michael Katz, Parikshit Ram, Shirin Sohrabi, and Octavian Udrea. Exploring context-free languages via planning: The case for automating machine learning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30(1):403–411, Jun. 2020. 3

[26] Tzofi Klinghoffer, Siddharth Somasundaram, Kushagra Tiwary, and Ramesh Raskar. Physics vs. learned priors: Rethinking camera and algorithm design for task-specific imaging. In *2022 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2022. 2

[27] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 3

[28] Michael F Land and Dan-Eric Nilsson. *Animal eyes*. OUP Oxford, 2012. 1

[29] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 5

[30] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 5

[31] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5695–5703, 2016. 3

[32] Radu Marinescu, Akihiro Kishimoto, Parikshit Ram, Ambrish Rawat, Martin Wistuba, Paulito P. Palmes, and Adi Botea. Searching for machine learning pipelines using a context-free grammar. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8902–8911, May 2021. 3

[33] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021. 3

[34] Christopher A Metzler, Hayato Ikoma, Yifan Peng, and Gordon Wetzstein. Deep optics for single-shot high-dynamic-range imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1375–1385, 2020. 2

[35] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021. 3

[36] Emmanuel Onzon, Fahim Mannan, and Felix Heide. Neural auto-exposure for high-dynamic range object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7710–7720, 2021. 2

[37] Nicolas Robidoux, Luis E Garcia Capel, Dong-eun Seo, Avinash Sharma, Federico Ariza, and Felix Heide. End-to-end high dynamic range camera pipeline optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6297–6307, 2021. 2

[38] Jonathan Schaeffer and Aske Plaat. New advances in alpha-beta searching. In *Proceedings of the 1996 ACM 24th annual conference on Computer science*, pages 124–130, 1996. 4

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 3

[40] Javier Selva, Anders S Johansen, Sergio Escalera, Kamal Nasrollahi, Thomas B Moeslund, and Albert Clapés. Video transformers: A survey. *arXiv preprint arXiv:2201.05991*, 2022. 5

[41] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein. End-to-end optimization of optics and image processing for achromatic extended

depth of field and super-resolution imaging. *ACM Trans. Graph. (SIGGRAPH)*, 2018. 2

[42] Qilin Sun, Ethan Tseng, Qiang Fu, Wolfgang Heidrich, and Felix Heide. Learning rank-1 diffractive optics for single-shot high dynamic range imaging. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1386–1396, 2020. 2

[43] Ethan Tseng, Ali Mosleh, Fahim Mannan, Karl St-Arnaud, Avinash Sharma, Yifan Peng, Alexander Braun, Derek Nowrouzezahrai, Jean-Francois Lalonde, and Felix Heide. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Transactions on Graphics (TOG)*, 40(2):1–19, 2021. 2

[44] Hernan Ceferino Vazquez, Jorge Sánchez, and Rafael Carrascosa. GramML: Exploring context-free grammars with model-free reinforcement learning. In *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2022. 3

[45] Tianyu Zhang, Amin Banitalebi-Dehkordi, and Yong Zhang. Deep reinforcement learning for exact combinatorial optimization: Learning to branch. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3105–3111. IEEE, 2022. 3

[46] Allan Zhao, Tao Du, Jie Xu, Josie Hughes, Juan Salazar, Pingchuan Ma, Wei Wang, Daniela Rus, and Wojciech Matusik. Automatic co-design of aerial robots using a graph grammar. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11260–11267. IEEE, 2022. 3

[47] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13760–13769, 2022. 7

[48] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 5