TaCOS: Task-Specific Camera Optimization with Simulation*

Chengyang Yan, Donald G. Dansereau

Abstract—The performance of robots in their applications heavily depends on the quality of sensory input. However, designing sensor payloads and their parameters for specific robotic tasks is an expensive process that requires wellestablished sensor knowledge and extensive experiments with physical hardware. With cameras playing a pivotal role in robotic perception, we introduce a novel end-to-end optimization approach for co-designing a camera with specific robotic tasks by combining derivative-free and gradient-based optimizers. The proposed method leverages recent computer graphics techniques and physical camera characteristics to prototype the camera in software, simulate operational environments and tasks for robots, and optimize the camera design based on the desired tasks in a cost-effective way. We validate the accuracy of our camera simulation by comparing it with physical cameras, and demonstrate the design of cameras with stronger performance than common off-the-shelf alternatives. Our approach supports the optimization of both continuous and discrete camera parameters, manufacturing constraints, and can be generalized to a broad range of camera design scenarios including multiple cameras and unconventional cameras. This work advances the fully automated design of cameras for specific robotics tasks. Code and data are available on our project page at https://roboticimaging.org/Projects/TaCOS/.

I. INTRODUCTION

Modern robotic systems heavily rely on inputs from their perception systems, where cameras play a critical role. The quality of the camera captures directly impacts the performance of the robots. An analogy in nature is the significant impact of animals' visual perceptions on their everyday tasks. It is widely accepted that evolution designs distinct visual perception systems for different species to suit their habitats [1]. Therefore, a sophisticated approach is necessary for robots to design their cameras based on their applications.

The current method for designing cameras for robots is a cumbersome process, typically requiring professionals to devise designs based on their experiences, followed by extensive experiments with various design decisions. Furthermore, off-the-shelf cameras are often chosen for robots, designed in isolation from the perception tasks, while jointly designing the cameras with tasks has shown its advantages in literature.

Existing works model and optimize the imaging system for the tasks through software. Ray tracing renderers are typically used as they simulate the physical light transport and image formation process. Many existing works have employed a ray tracing renderer to design and evaluate



Fig. 1. Our method combines a derivative-free optimizer and gradientbased optimizer to co-design the camera with robotic perception tasks with a camera simulation built in UE5 with real-time ray tracing and a physicsbased noise model. Our approach supports the optimization of discrete and continuous camera parameters for manufacture constraints and the generalization to other camera design problems.

cameras for autonomous driving [2]–[6]. However, manual tuning of camera parameters for optimization is still required, and joint optimization of the cameras and the tasks is not addressed in these methods.

Other works propose to co-design the imaging system with perception tasks automatically using differentiable ray tracing or proxy neural networks to simulate the imaging systems [7]–[22]. Nevertheless, their design space is restricted by the image datasets which prevent them from generalizing to more complex camera design problems involving field-ofview (FOV), multi-cameras, or unconventional cameras.

Game engines, such as Unreal Engine (UE), have been repurposed for simulating cameras as they support ray tracing and allow the production of video sequences and interaction with virtual environments closely resembling the operations of robots. Klinghoffer et al. [23] propose a reinforcement learning (RL) method to design cameras and evaluate via a UE-based simulator. Although achieving high performance, this method optimizes a complex neural network that learns to design cameras, which is inefficient compared to a method that directly optimizes the camera parameters.

As illustrated in Fig. 1, we introduce an end-to-end camera design method that directly optimizes camera parameters for perception tasks with simulation. To design and evaluate the cameras, we establish a procedural generation algorithm generating virtual environments in UE5 for robotic operations with machine learning labels. We propose a camera simulator that allows the tuning of a variety of camera parameters, and we additionally address the images' signal-to-noise ratio

^{*}This work was supported by ARIA Research Pty Ltd and the Australian government CRC Projects Round 11 grant

The authors are with the Australian Centre For Robotics (ACFR), School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney. cyan8191@uni.sydney.edu.au, donald.dansereau@sydney.edu.au

(SNR) by including a physics-based noise model [24].

Inspired by evolutionary processes we employ a genetic algorithm [25], a derivative-free optimizer, to optimize discrete, continuous, and categorical variables in our simulation framework. The approach supports manufacturing constraints and categorical variables, allowing selection of parts (optics, image sensors, etc.) from catalogues where customized manufacture is infeasible. Finally, we propose an alternative quantized continuous approach to discrete variables that allows consideration of the interdependencies between them.

We implement our approach on example tasks of obstacle avoidance, object detection, and feature detection. We demonstrate that the camera designed by our approach achieves compelling performance compared to high-quality robotic and machine vision cameras in our simulation, and we validate the accuracy of our simulation by comparing it with physical cameras. In summary, we make the following contributions:

- We introduce an end-to-end camera design method that combines derivative-free and gradient-based optimization to automatically co-design the camera with downstream perception tasks and allows optimization of continuous and discrete camera variables.
- We develop a camera simulation and evaluation technique that includes a physics-based noise model and procedurally generated virtual environments.
- We validate our simulator and evaluation method by establishing equivalence of both low-level image statistics and high-level task performance between synthetic and captured imagery.
- We demonstrate the design of cameras with stronger performance than common off-the-shelf alternatives.

This work is a key step in simplifying the process of designing cameras for robots, where mobility and the performance of tasks are significant and the manufacturability of cameras is constrained. Code and data will be released upon acceptance of the paper.

Limitations: The experiment in this work focuses on optimizing a single RGB camera with a CMOS image sensor. However, since our optimization method supports both discrete and continuous variables, and our camera simulator can handle additional camera design decisions, we expect our method can be immediately generalized to more complex camera design problems involving multiple cameras and unconventional cameras. Furthermore, our experiment solely considers camera parameters determined at the manufacturing stage, camera parameters that can be dynamically adjusted during robot operation such as exposure settings will be addressed in future work.

II. RELATED WORK

Designing cameras tailored for robotics or machine learning tasks using software simulation has gained popularity. Blasinski et al. [2] propose optimizing a camera design to detect vehicles for autonomous driving. They used synthetic data generated with ISET3D [26] and ISETCam [27]. The method optimizes cameras by experimentally analyzing the impact of different image postprocessing pipelines and autoexposure algorithms on downstream object detection tasks. This work continues in [3]–[6], extending the experiments to larger datasets and more diverse synthetic datasets, and an optimization framework for high dynamic range (HDR) imaging applications. Nevertheless, these methods often require manual tuning and testing of different camera parameters, whereas our method is an end-to-end approach that automatically optimizes the camera design.

Other works have also proposed end-to-end methods to optimize the imaging system based on tasks. Many use gradient-based optimization with differentiable camera simulators, including differentiable ray tracing algorithms and proxy neural networks for non-differentiable image formation processes. In these works, images captured by physical cameras are used as input scenes to their pipeline, and their simulators which simulate the image formation process convert the input images to images formed by their proposed cameras. Applications include extended depth of field (DOF) [7], [8], depth estimation [9]-[12], object detection [13]-[16], HDR imaging [17], [18], image classification [19]-[21], and motion deblurring [22]. However, these methods use physically captured image datasets so that their camera simulation is restricted to the domains of the datasets. Key camera design decisions such as the camera's FOV, resolution, use of multiple cameras, and the design of unconventional cameras (light field, etc.), are not addressed. Our method establishes a virtual environment to support the simulation and optimization of a much broader range of camera designs.

Recent works have explored the application of RL for end-to-end optimization of imaging systems. Klinghoffer et al. [23] advocate for RL-based training of a camera designer, encompassing various camera parameters using the CARLA Simulator [28]. Hou et al. [29] introduce another RL-based approach for pedestrian detection. Although RL demonstrates impressive results in camera design, it involves optimizing complex neural networks that learn to design cameras, demanding more training data and time since the neural network contains a larger number of parameters that need to be optimized. In contrast, our approach directly optimizes the camera's parameters, yielding competitive results with much less computation.

III. METHOD

In this work, we introduce a procedural generation method in UE5 for photorealistic renders, followed by a physicsbased noise model. The quality of the resulting images is then evaluated in robotic tasks and utilized to optimize the camera design. The proposed method is illustrated in Fig. 2.

A. Simulation Environment

To create diverse environments for evaluation, we implemented the procedural generation of random virtual environments and their associated semantic labels, with support for application-specific objects. Our virtual environment is generated with UE5. We utilize real-time hardware ray tracing,



Fig. 2. Proposed methodology. We establish a virtual environment in UE5 with a procedural generation method and obtain renders from the environment using a ray-traced simulated camera. We then add physics-based sensor-specific image noise to the renders and input them to the robotic perception tasks. In our optimization, we jointly optimize the camera parameters Φ_{camera} on a fitness function F with a derivative-free optimizer, as well as the weights of an object detector Φ_{OD} on an object detection loss function l_{OD} with a gradient-based optimizer.

combined with the software ray tracing global illumination and reflection method supported by UE5. This combination provides realistic shadowing, ambient occlusion, illumination, reflections, etc [30].

An auto-agent, simulating the robot, navigates the virtual environment automatically. Its trajectory is randomly generated with the environment using UE Navigation System [30]. The auto-agent carries a UE5 camera, capturing scene renders as it moves, and it interacts with the environments so that every object the agent collides with is recorded.

B. Camera Simulation

Our camera simulation comprises two components, as depicted in Fig. 2, including a scene capture component and an image noise synthesis component.

Scene Capture. We employ a UE camera to capture scene irradiance from the virtual environment with ray tracing. The scene capture component allows the configuration of parameters associated with cameras' placement (location, orientation), optics (focal length and aperture), the image sensor (width, height, and pixel count), exposure settings (shutter speed and ISO), and multi-camera designs (number of cameras and their poses), as well as the configuration of algorithms in the image processing pipeline (white balancing, tone mapping, colour correction, gamma correction, and compression).

We experimentally validate our method using parameters readily supported by UE. Additional parameters like geometric distortion and defocus blur could be added by augmenting the renderer. The noise synthesis model described below serves as an example of such an augmentation.

Noise Synthesis. Image noise is a fundamental limiting factor for many robotic vision tasks that is tightly coupled to camera design parameters such as pixel size and exposure settings. As the UE5 simulation lacks a realistic noise model, we incorporate a post-render image augmentation that introduces noise. We employ thermal and signal-dependent Pois-

son noise following the affine noise model (Heteroscedastic Gaussian) [24]. The affine noise model expresses the pixel variance (σ^2) as

$$\sigma^2 = \sigma_p^2 \bar{I} + \sigma_t^2, \tag{1}$$

where σ_t^2 is the variance of constant thermal noise, σ_p^2 is a scaling factor for intensity-varying photon noise, and \overline{I} is the mean intensity observed at a pixel. The measured intensity by the camera is the integration of the input intensity arriving at the pixel over the camera's exposure time, scaled by the camera's gain:

$$I = EGI_{\text{input}},\tag{2}$$

where E and G are the exposure time and gain, respectively.

Calibrating the noise model follows established methods [31]–[33]. In this work, we use a greyscale test target with colorbars containing uniformly distributed grey levels from fully white to fully black. With captured images of the test target, we determine mean intensities and variances for each pixel, using these values to fit the affine noise model defined in Eq. 1.

The noise model can be generalized to different camera exposure and gain settings. Considering the observed pixel intensity by the camera, changes due to exposure time setting are reflected in the intensity value, as explained by Eq. 2, therefore, we generalize the noise model to other exposure and gain settings by multiplying the ratio of the new gain (*G*) and the calibrated gain (*G*₀) used in the noise calibration stage. Since the noise model describes the variance of the noise, the gain ratio is in the second order. Substituting the new gain with the ratio into Eq. 2, replacing \overline{I} in Eq. 1 with Eq. 2, and rewriting σ_t^2 with the new gain using the ratio, the noise model becomes

$$\sigma^{2} = \frac{G}{G_{0}}\sigma_{p}^{2}\bar{I} + \frac{G^{2}}{G_{0}^{2}}\sigma_{r}^{2}.$$
(3)

We generalize our noise model to different image sensors by considering the ratio of their pixel sizes. For the same illumination condition, larger pixel sizes capture more photons, resulting in a higher input intensity level (I_{input}). This is readily reflected by adjusting the gain in Eq. 3 inversely proportional to the pixel area. While we employ these observations to generalise noise characterisations, it is also possible to directly characterize multiple sensors and directly use these characterizations.

C. Optimization

Optimizers. We combine derivative-free and gradientbased optimizers to optimize the camera design and the downstream perception task respectively. Specifically, we use the genetic algorithm [25] to optimize the camera design and use the Adam [34] optimizer to optimize the object detection model. We also expect other derivative-free optimizers would apply in our proposed approach.

Camera Parameters. Our optimizer can handle the optimization of all parameters captured in the camera simulation, e.g. those outlined in Sec. III-B. The genetic algorithm is designed to accommodate both continuous and discrete parameters, enhancing the generalizability of our method. For instance, parameters related to optics and image sensors can be optimized as continuous variables if there are no manufacturing constraints on new optics and sensors. Alternatively, they can be selected from existing lens/sensor catalogs, allowing manufacturing and availability to be considered.

Discrete Variable Optimization. The discrete variable optimization in our approach offers two schemes: *fully discrete* and *quantized continuous*. In the fully discrete scheme, we optimize the parameter *x*, representing an index of the parameter in its available values, by constraining the mutation stage of the genetic algorithm to ensure that only available values of this variable are used.

In the quantized continuous scheme, we adopt the "quantized continuous variables" method introduced in [16]. Here, in each iteration, the discrete parameter x can freely change as a continuous variable from its current best value obtained in the previous iteration. However, it is then replaced with the closest value from its available range:

$$x^* = \arg\min_{k} ||x - x_k||_2^2,$$
 (4)

where x^* is the parameter retained from its available range, *x* is the variable obtained from the optimization process, and *x_k* represents the *k*-th parameter in its available values.

A limitation of the fully discrete scheme appears when the variable x connects to other variables. For example, if x is the available image sensors, then this single variable encompasses multiple sensor-related parameters such as width, height, and pixel size. Selecting x categorically does not benefit from the interrelationships between these parameters. The quantized continuous scheme overcomes this by allowing all included parameters in x to be optimized freely and then replaced by a set of values corresponding to the closest categorical x. **Fitness Functions.** To optimize the camera design, we establish a fitness function based on several robotic tasks. Initially, we ensure that the FOV is sufficient to capture obstacles in the robots' motion path. Therefore, we propose an obstacle detection term for the fitness function, given by the ratio of the number of obstacles the camera sees (o_{seen}) to the total number of obstacles in the robot's path (o_{total}):

$$F_{obstacle} = \frac{o_{seen}}{o_{total}}.$$
(5)

Object detection is another essential robotic task. In this work, we jointly train an object detection network with the optimization of the camera. The object detection network is trained using its loss function (l_{OD}), established based on the specific model in use. Simultaneously, the parameters of the camera are optimized using the average precision (AP) as a term in the fitness function, accounting for both object classification accuracy (precision) and object detection accuracy (intersection-over-union or IoU). We compute the AP with an IoU threshold of 0.5:

$$F_{OD} = AP@0.5IoU.$$
(6)

Simultaneous Localization and Mapping (SLAM) is common in robotics, and we address this task by optimizing the number of features found in the camera's frames as many SLAM methods, such as ORB-SLAM [35], rely on features extracted from input images. Inspired by [36], we use the number of inlier features, which are the correctly matched features that accurately represent the same points or regions across consecutive frames, combined with the ratio of inlier features to total features (inliers plus outliers), in our fitness function to emphasize both the number of inlier features and the accuracy of the features detected:

$$F_{feature} = \lambda_{inlier} n_{inlier} + \lambda_{ratio} \frac{n_{inlier}}{n_{total}},$$
(7)

where n_{inlier} and n_{total} are the average numbers of inlier features and total detected features over all images captured with one set of camera parameters, respectively, and λ s represent the weights of these two terms.

All terms of the fitness function are combined to form the total fitness function used in this optimization:

$$F = \lambda_{obstacle} F_{obstacle} + \lambda_{OD} F_{OD} + \lambda_{feature} F_{feature}, \quad (8)$$

where λs are the weights of the tasks.

IV. RESULTS

We apply our approach to a camera design problem for an indoor robot. We provide details for our implementation, validate our proposed simulator, and compare our optimized cameras to those designed by humans.

A. Implementation

Assumptions. Our experiment is conducted under the following assumptions: (1) objects' distances to the camera exceed the camera's hyperfocal distance so that the DOF is safely neglected, (2) the lens of our camera is free of geometric distortion and chromatic aberration, and (3)



Fig. 3. Captured and synthetic images of the colorbar test target and their histograms. Despite the differences in colour intensities caused by the manufacture of the test target, the distribution of the pixel values in the synthetic image matches with the captured image, validating the accuracy of our noise model.

the height of the robot is between 1m to 2m, which is addressed by randomly changing the camera's height during optimization.

Scene Generation. We establish an indoor environment to evaluate our approach, set to a common size of 15 meters in width, 15 meters in length, and 3 meters in height. During optimization, the floorplan and object locations are randomly generated to introduce variability. The objects belong to 10 classes, including sofa, bed, table, chair, bathtub, bathroom basins, computer/TV, plant, lamp, and toy. The objects are 3D assets obtained from the UE marketplace.

Design Space. We focus on designing geometric parameters determining the camera's FOV and photometric parameters determining the resolution as they affect the robot's mobility and the effective resolution of objects.

For FOV, we consider the mounting angle (θ) of the camera on the robot, which is the pitch angle dictating the FOV's orientation, along with the focal length (f) and image sensor dimensions (width w and height h) dictating the FOV's size. The number of pixels is determined by the image sensor's pixel size (p) and its dimensions.

We restrict our design to readily available sensors by optimizing the image sensor (*i*) as a categorical variable, selecting from a catalogue of available parts. We collect a catalogue of 54 commercial CMOS image sensors (i_{c1} , i_{c2} ,..., i_{c54}), where each *i* comprises a set of sensor-related parameters (*w*, *h*, and *p*). We compare two optimisation techniques, treating *i* as fully discrete and using the quantized continuous approach.

Data Collection. We collect a dataset of 2000 images using our simulator to pre-train the object detector before the optimization phase. These images were captured with 10 random camera configurations. Throughout the optimization process, the camera designs in each iteration are assessed in real time. We collect 500 images from the virtual environment for each camera design, evaluating them through perception tasks. The images are captured in three distinct procedurally generated environments for each camera design.

Noise Synthesis. For noise synthesis, we calibrate the noise characteristics of a FLIR Flea3 Camera [37] with a Sony IMX172 sensor and fit it to the affine noise model. This model is then generalized to other image sensors in our catalogue, which comprises 54 sensors from five manufacturers. Notably, 34 sensors are from Sony due to their widespread

use in robotic and machine vision cameras. This intentional bias toward sensors from the same manufacturer aligns with our assumption about image noise being closer to the truth for sensors from the same manufacturer. Hence, we also selected a Sony sensor to calibrate and use as the baseline. We validate our noise model together with our rendering in Sec. IV-B.

Obstacle Positioning. To assess the camera's capability to detect obstacles that might impact the mobility of the robot, we position gold-coloured thresholds at the entrance of each room within our simulation environment, as depicted in the tasks section of Fig. 2. These thresholds represent low-height obstacles that could potentially pose a danger to robots, emphasizing the need for a camera with an appropriate FOV. We solely place low-height obstacles, as the detection of taller obstacles is constrained by the object and feature detection tasks. The determination of whether the obstacle is visible to the camera is based on whether the obstacle is visible in the rendered imagery. Additionally, the auto-agent is programmed to recognize and respond to stepping on a threshold, even if it is not rendered visually.

Feature Detector and Inlier Feature Matching. In this work, We extract Oriented FAST and Rotated BRIEF (ORB) features [38], and we employ the Brute-Force Matcher in OpenCV [39] to identify inlier features across consecutive frames.

Object Detector. We utilize a Faster-RCNN [40] object detector with a ResNet-50 [41] backbone. The object detector is pre-trained on the image dataset of 2000 images we collected and is fine-tuned for each camera design during the optimization process. Training and fine-tuning are carried out using an Adam optimizer [34] with a batch size of 8. The learning rate is set to $1 \cdot 10^{-4}$ with a decay rate of 0.5 for every 5 steps in both pre-training and fine-tuning stages.

Derivative-Free Optimization. The genetic algorithm is implemented with 20 generations and 10 solutions per generation. For each generation, the top 3 solutions are kept in the next generation, while the top 5 solutions are used as parents to produce the offspring via a uniform crossover method in the optimization, where each parameter in the offspring is randomly selected from each parent. Then the offspring are mutated by applying a multiplication factor and an addition value, the multiplication factor is randomly selected between



Fig. 4. The number of inlier ORB features versus the ratio of inlier features in real and synthetic images for the OAK-D, FLIR, and Basler cameras. The graph shows the ranking of the cameras' performance in our simulation aligns with the physical cameras, and the differences in their performance between the captured and synthetic images are consistent.

0.8 and 1.2, whereas the addition value is randomly selected from -3 to 3.

For the weights in our fitness function (Eq. 7 and Eq. 8), we balance the weights of all terms by setting λ_{inlier} to 0.0025 (inlier number in an image is typically 100-200), λ_{ratio} to 0.5, $\lambda_{obstacle}$, λ_{OD} , and $\lambda_{feature}$ to 1 as we consider these tasks equally significant.

B. Simulator Validation

To validate our simulator, we compare synthetic images from our simulator with those captured by physical cameras in terms of image statistics and task performance.

Image Statistics. We employ a test target with greyscale colorbars in a controlled illumination environment in the physical world. Images are captured using the FLIR camera used for noise model calibration. Subsequently, the same test target is recreated in our UE5 simulator, ensuring consistency in illumination, relative camera position, and camera parameters. Fig. 3 displays the captured and synthetic images along with their histograms.

The synthetic image exhibits a linear intensity response since our simulator operates in a linear colorspace without introducing gamma correction, as indicated by the uniform greylevel distribution of colorbars in the test target. In contrast, the captured image shows non-linear colour distribution introduced by the printer when manufacturing the test target. Despite histogram differences, the comparison reveals that, for a given mean intensity, pixel values in the synthetic image align with the distribution observed in the captured image. This alignment validates the accuracy of our noise model.

Perception Task Performance. To validate task performance, we focus on the feature detection task, as obstacle avoidance is accurately evaluated by the physical interaction between objects and the auto-agent in our simulator, and the object detection task presents challenges in aligning real and virtual objects. The feature detection task is chosen for its suitability for physical and virtual experiments.

We utilize a test target employed in [42], which incorporates features with varying scales and depths, placed against a texture-less background. Ten consecutive frames are captured with a constant translational motion, maintaining consistent



Fig. 5. Example images captured in (a) daytime scenario (20 lux) with a lower baseline camera gain (5 dB), and in (b) a nighttime scenario (2 lux) with a higher baseline camera gain (15 dB).

illumination, camera parameters, and relative positioning between the target and the camera for both real and virtual experiments, the result is illustrated in Fig. 4.

The comparison involves three robotic/machine vision cameras: the RGB camera of the Luxonis OAK-D Pro Wide camera [43] with a Sony IMX378 sensor, the FLIR Flea3 Camera [37] with a Sony IMX172 sensor, and the Basler Dart DaA1280-54uc camera [44] with an Onsemi AR0134 sensor. Specific noise models are applied to these cameras in our simulator, calibrated through the method detailed in Sec. III-B. The feature detection task provides a reliable measure of performance consistency between physical and simulated environments.

The experiment demonstrates that synthetic images from our simulator yield a larger number of inlier features and a higher ratio of inlier features to the total number of features extracted due to the reduction in quality of the test target resulting from the manufacturing process. However, the relative performance of cameras, critical for optimization purposes, is maintained across simulated and physical settings. This consistency validates that our camera simulations accurately reflect the real-world performance of the cameras and effectively evaluate the performance of camera designs.

C. Performance Evaluation

We present the optimized set of camera parameters obtained through our approach and evaluate the camera's performance in Tab. I, considering the obstacle (threshold) detection accuracy, the AP score, the average number of inlier ORB features across consecutive frames, and the ratio of the average number of inlier features to the average number of total ORB features detected. Additionally, we report the optimized set of parameters under two different application scenarios. One scenario involves daytime operation in a well-illuminated simulation environment, while the other pertains to nighttime operation with limited light in the environment, illustrated in Fig. 5. We observed that different application scenarios led to distinct camera designs. Our approach designed a camera with a larger pixel size for nighttime applications, which is expected as a sensor with larger pixels delivers higher SNR since larger pixels gather more light. Hence, sensors with larger pixels require lower gain to capture images with the comparable measured intensity compared to sensors with smaller pixels.

We compare our optimized camera with three human-





92 Ratio:0.04 Inlier:60 Ratio:0.03 Inlier:104 Ratio:0.05 Inlier:194 Ratio:0.10 Inlier:214 Ratio:0.11 (c) Performance on Example Tasks

Fig. 6. The vertical (a) and horizontal (b) FOVs of the camera optimized by our method those designed by humans under the daytime scenario, and example evaluation results (c) with these cameras are presented. Our method designs a camera with the largest FOV, this FOV allows the camera to capture all obstacles and objects, extracting more features while maintaining sufficient resolution to maintain effective object and feature detection.

TABLE I

The comparison of the parameters and performance of cameras designed with our method using fully discrete and quantized continuous schemes and 3 robotic/machine vision cameras. The optimized parameters via our method are labelled with \bigcirc , including all parameters of our cameras, the object detection network, and mounting angles of off-the-shelf cameras. Our approach designs cameras with higher performance in both scenarios, while the quantized continuous scheme achieves higher performance compared to the fully discrete scheme in this experiment due to its consideration of parameters' interdependencies.

	Camera	Camera Parameters				Performance			
Scenario		Pitch Angle	Focal Length	Sensor Size	Pixel Size	Obstacle Detect.	Object Detect. 😑	Inlier Number	Inlier Ratio
		θ (°)	f (mm)	$w \times h \pmod{m}$	p (µm)	Accuracy	AP@0.5IoU		
Day	OAK-D [43]	-20.04 🔴	2.75 •	6.29×4.71 ●	1.55 🔴	1	0.37	115	0.07
	FLIR [37]	-23.28 😐	3.6 🔴	6.2×4.65 ●	1.55 🔴	1	0.37	77	0.05
	Basler [44]	-25.90 😑	3.6 🔴	4.8×3.6 ●	3.75 🔴	1	0.23	131	0.08
	Ours - Fully Discrete	-24.69 😑	3.77 😑	8.45×6.76 😑	6.6 😑	1	0.51	189	0.12
	Ours - Quantized Continuous	-26.34 🔵	2.88 😑	7.31×5.58 😑	4.5 😑	1	0.64	224	0.13
Night	OAK-D [43]	-19.72 🔵	2.75 ●	6.29×4.71 ●	1.55 🔴	1	0.34	117	0.06
	FLIR [37]	-22.71 😑	3.6 🔴	6.2×4.65 ●	1.55 🔴	1	0.36	69	0.03
	Basler [44]	-25.83 😑	3.04 🔴	4.8×3.6 ●	3.75 🔴	1	0.16	122	0.07
	Ours - Fully Discrete	-23.66 😑	3.10 😑	7.2×5.4 😐	4.5 😑	1	0.42	170	0.11
	Ours - Quantized Continuous	-22.58 😑	3.49 😑	14.48×9.94 😑	9 😑	1	0.57	172	0.11

designed robotic and machine vision cameras, OAK-D, FLIR, and Basler, used in previous experiments. This comparison takes place within our simulation environment, where we apply specific noise models to these cameras as before, and we optimize the mounting angles of these cameras as it is configurable, while the others remain fixed. Configurable parameters in this experiment are indicated with \bigcirc , and fixed parameters are marked with \bigcirc in Tab. I. Fig. 6 illustrates the FOVs and example evaluations of our proposed cameras and the robotic/machine vision cameras.

The results of the comparison demonstrate improvements in the performance of all tasks, with performance being notably lower under the nighttime scenario due to a reduced SNR, as well as with the fully discrete optimization scheme due to the disregard of parameters' interdependencies. Additionally, we show our method achieves compelling results with only 20 iterations, requiring 200 camera design evaluations in total. In contrast, the RL method proposed in [23] requires 30000 to 40000 camera design evaluations for a task and design space of lower complexity.

V. CONCLUSION

In this work, we present a novel end-to-end approach that combines derivative-free and gradient-based optimizers to co-design cameras with multiple robotic perception tasks. Utilizing UE5 and an affine noise model, we construct a camera simulator tailored for robotics, subsequently validating its accuracy through comparison with physical cameras. Our method takes into account continuous, discrete, and categorical camera parameters, and advances a quantized continuous approach to discrete variables that allows consideration of the interdependencies between them. We believe this work is an important step toward principled and automated design of cameras for robots that accounts for the interdependency between the cameras and the algorithms that interpret them.

We expect our method can immediately generalize to more complex camera design problems involving multiple cameras and more camera parameters by including more discrete variables and fully utilizing the configurable camera parameters in our simulation. For future work, we aim to extend to a task-driven control algorithm that dynamically adjusts camera parameters such as gain and exposure settings in an online manner.

ACKNOWLEDGMENT

We would like to thank both ARIA Research Pty Ltd and the Australian government for their funding support via a CRC Projects Round 11 grant.

REFERENCES

- [1] M. F. Land and D.-E. Nilsson, Animal eyes. OUP Oxford, 2012.
- [2] H. Blasinski, J. Farrell, T. Lian, Z. Liu, and B. Wandell, "Optimizing image acquisition systems for autonomous driving," *Electronic Imaging*, vol. 2018, no. 5, pp. 161–1, 2018.
- [3] Z. Liu, S. Minghao, J. Zhang, S. Liu, H. Blasinski, T. Lian, and B. Wandell, "A system for generating complex physically accurate sensor images for automotive applications," *Electronic Imaging*, vol. 2019, pp. 53–1, 01 2019.
- [4] Z. Liu, T. Lian, J. Farrell, and B. Wandell, "Soft prototyping camera designs for car detection based on a convolutional neural network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.
- [5] K. Weikl, D. Schroeder, D. Blau, Z. Liu, and W. Stechele, "Endto-end imaging system optimization for computer vision in driving automation," in *Proceedings of IS&T Int'l. Symp. on Electronic Imaging: Autonomous Vehicles and Machines*, 2021.
- [6] Z. Liu, D. Shah, A. Rahimpour, D. Upadhyay, J. Farrell, and B. A. Wandell, "Using simulation to quantify the performance of automotive perception systems," *arXiv preprint arXiv:2303.00983*, 2023.
- [7] 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 superresolution imaging," ACM Trans. Graph, vol. 37, no. 4, pp. 1–13, 2018.
- [8] Q. Sun, C. Wang, F. Qiang, D. Xiong, and H. Wolfgang, "End-toend complex lens design with differentiable ray tracing," *ACM Trans. Graph*, vol. 40, no. 4, pp. 1–13, 2021.
- [9] L. He, G. Wang, and Z. Hu, "Learning depth from single images with deep neural network embedding focal length," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4676–4689, 2018.
- [10] J. Chang and G. Wetzstein, "Deep optics for monocular depth estimation and 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10193– 10202.
- [11] H. Ikoma, C. M. Nguyen, C. A. Metzler, Y. Peng, and G. Wetzstein, "Depth from defocus with learned optics for imaging and occlusionaware depth estimation," in 2021 IEEE International Conference on Computational Photography. IEEE, 2021, pp. 1–12.
- [12] S.-H. Baek and F. Heide, "Polka lines: Learning structured illumination and reconstruction for active stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5757–5767.
- [13] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. S. Arnaud, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, "Hyperparameter optimization in black-box image processing using differentiable proxies." ACM Trans. Graph., vol. 38, no. 4, pp. 27–1, 2019.
- [14] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide, "Differentiable compound optics and processing pipeline optimization for end-to-end camera design," *ACM Trans. Graph.*, vol. 40, no. 2, pp. 1–19, 2021.
- [15] N. Robidoux, L. E. G. Capel, D.-e. Seo, A. Sharma, F. Ariza, and F. Heide, "End-to-end high dynamic range camera pipeline optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision* and Pattern Recognition, 2021, pp. 6297–6307.
- [16] G. Côté, F. Mannan, S. Thibault, J.-F. Lalonde, and F. Heide, "The differentiable lens: Compound lens search over glass surfaces and materials for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20803–20812.
- [17] C. A. Metzler, H. Ikoma, Y. Peng, and G. Wetzstein, "Deep optics for single-shot high-dynamic-range imaging," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1375–1385.
- [18] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, "Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors," *IEEE transactions* on pattern analysis and machine intelligence, vol. 42, no. 7, pp. 1642– 1653, 2020.
- [19] J. Chang, V. Sitzmann, X. Dun, W. Heidrich, and G. Wetzstein, "Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification," *Scientific reports*, vol. 8, no. 1, p. 12324, 2018.

- [20] S. Diamond, V. Sitzmann, F. Julca-Aguilar, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Towards end-to-end image processing and perception," ACM Trans. Graph., vol. 40, no. 3, pp. 1–15, 2021.
- [21] Y. Zhang, B. Dong, and F. Heide, "All you need is raw: Defending against adversarial attacks with camera image pipelines," in *European Conference on Computer Vision*. Springer, 2022, pp. 323–343.
- [22] C. M. Nguyen, J. N. Martel, and G. Wetzstein, "Learning spatially varying pixel exposures for motion deblurring," in 2022 IEEE International Conference on Computational Photography. IEEE, 2022, pp. 1–11.
- [23] T. Klinghoffer, K. Tiwary, N. Behari, B. Agrawalla, and R. Raskar, "Diser: Designing imaging systems with reinforcement learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 23 632–23 642.
- [24] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Processing*, vol. 89, no. 12, pp. 2609–2629, 2009.
- [25] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.
- [26] B. Wandell, D. Cardinal, T. Lian, Z. Liu, Z. Lyu, D. Brainard, H. Blasinski, A. Ni, J. Dowling, M. Furth, T. Goossens, A. Ji, and J. Maxwell, "Iset/iset3d," 2 2024. [Online]. Available: https://github.com/ISET/iset3d
- [27] B. Wandell, D. Cardinal, D. Brainard, Z. Lyu, Z. Liu, A. Webster, J. Farrell, T. Lian, H. Blasinski, and K. Feng, "Iset/isetcam," 2 2024. [Online]. Available: https://github.com/ISET/isetcam
- [28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [29] Y. Hou, X. Leng, T. Gedeon, and L. Zheng, "Optimizing camera configurations for multi-view pedestrian detection," *arXiv preprint* arXiv:2312.02144, 2023.
- [30] Epic Games, "Unreal engine."
- [31] N. Ratner and Y. Y. Schechner, "Illumination multiplexing within fundamental limits," in 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007, pp. 1–8.
- [32] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise estimation from a single image," in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1. IEEE, 2006, pp. 901–908.
- [33] T. Wang and D. G. Dansereau, "Multiplexed illumination for classifying visually similar objects," *Applied Optics*, vol. 60, no. 10, pp. B23–B31, 2021.
- [34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [35] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [36] J. Tomasi, B. Wagstaff, S. L. Waslander, and J. Kelly, "Learned camera gain and exposure control for improved visual feature detection and matching," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2028–2035, 2021.
- [37] FLIR FLEA®3 USB3 Vision, FLIR Integrated Imaging Solutions Inc., 1 2017, rev. 8.1.
- [38] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in 2011 International conference on computer vision. Ieee, 2011, pp. 2564–2571.
- [39] Itseez, "Open source computer vision library," https://github.com/ itseez/opencv, 2015.
- [40] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards realtime object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer* vision and pattern recognition, 2016, pp. 770–778.
- [42] D. G. Dansereau, B. Girod, and G. Wetzstein, "Liff: Light field features in scale and depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8042–8051.
- [43] OAK-D Pro (USB boot with 256Mbit NOR flash), Luxonis, 12 2021.
- [44] Basler dart daA1280-54uc, Basler AG, 3 2024, rev. 85.