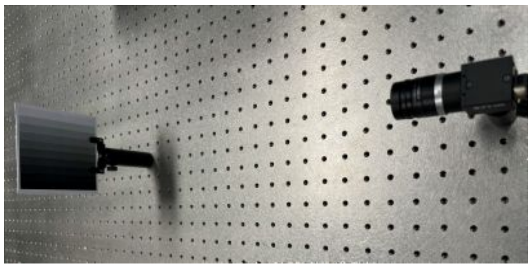
TaCOS: Task-Specific Camera Optimization with Simulation

Supplementary Material

(a)



(b)

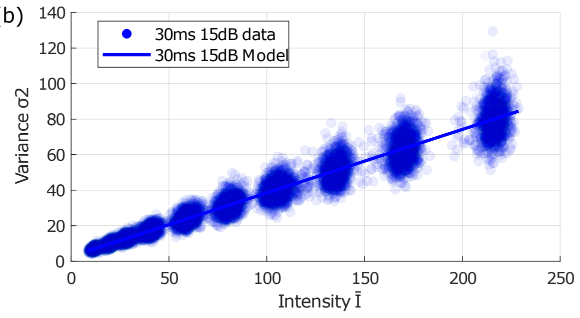


Figure 1. We calibrate the affine noise model [3] with a FLIR Flea3 Camera [2] using a greyscale test target with colorbars. The camera is configured to have a 30 ms exposure time and 15 dB gain. The calibration setup is illustrated in (a) and the plot of the pixel variances against mean intensities is displayed in (b).

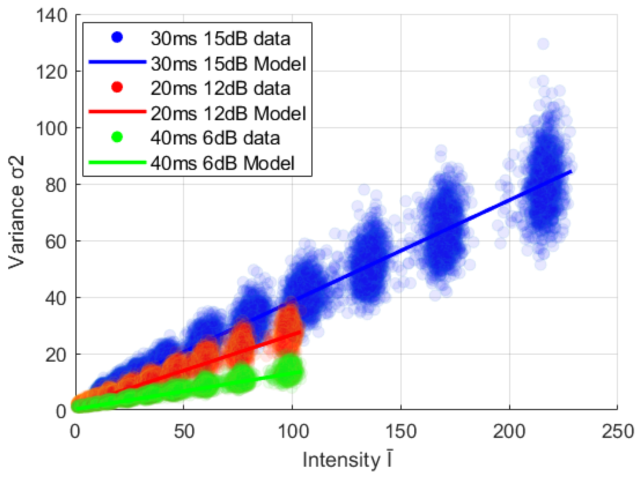


Figure 2. The noise model calibrated with 30 ms exposure time and 15 dB gain (blue) is generalized to the 20 ms exposure time and 12 dB gain (red), as well as 40 ms exposure time and 6 dB gain (green). The data on the plot are captured by the physical camera, the noise model for 30 ms exposure time and 15 dB gain are obtained from the data as in Fig. 1 (b), while noise models for the latter two exposure settings are obtained via our generalization method. The plot shows that generalized noise models match the captured data accurately. The intensities for the latter two exposure settings are smaller, which is caused by lower exposure settings.

1. Code

Our code and data are available through our GitHub page at `https://anonymous.4open.science/r/TaCOS-4403/`. We include the implementation of our camera design method for both the stereo camera and monocular camera design experiments, the source code and guidance for creating the indoor virtual environment used in the monocular camera design experiment, and the catalog of commonly available image sensors that we collected.

2. Additional Details on Noise Synthesis

We provide additional details on our image noise calibration and generalization method.

2.1. Noise Model Calibration

We adopt the affine noise model [3] in this work. The noise model describes a linear relationship between the variance (σ^2) in image pixel intensities for different mean intensity values ($\bar{I}$) in terms of constant thermal noise (σ_t) and intensity-varying photon noise ($\sigma_p^2 \bar{I}$):

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

Calibrating the noise model follows established methods [5, 6, 8]. 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.

We calibrate the noise model with a FLIR Flea3 Camera [2] with a Sony IMX172 image sensor. The exposure time is set as 30 ms and the gain is set as 15 dB. Note that the dark current noise is safely neglected in this work as the exposure time used (30 ms) is relatively short. The setup of this calibration is demonstrated in Fig. 1 and we display the plot of the obtained noise model in Fig. 1 (b).

2.2. Noise Model Generalization

The noise model in Eq. 1 is calibrated using a specific image sensor and exposure setting. We expand the equation so that it can be generalized to different exposure settings and image sensors.

Consider the intensity (I) in Eq. 1 as the measured pixel

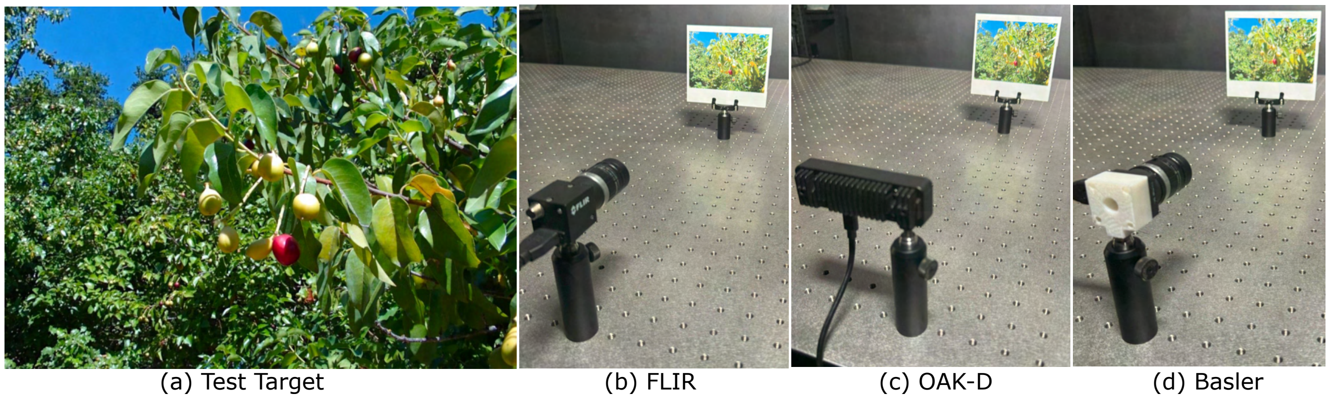


Figure 3. We adopt a test target (a) from [1] to validate our simulator in terms of the performance of feature extraction. We compare the relative performance of three off-the-shelf cameras, the experiment setups in the real world are shown in (b), (c), and (d), which are then duplicated in our virtual environment.

intensity by the camera: $I = EG\phi$, where E and G are the exposure time and gain respectively, and ϕ is the scene radiance. Intensity changes due to exposure time setting are reflected in the measured intensity value, 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. Then we can transform Eq. 1 to obtain the noise model with new exposure settings. For the photon noise term, replacing $\bar{I}$ in Eq. 1 with the new observed intensity and multiplying it with the gain ratio to scale the value of σ_p^2. For the thermal noise term, scaling σ_t^2 with the second order square of the ratio between the new gain and the calibrated gain, the noise model becomes Eq. 2 of the main paper.

In Fig. 2, we show the generalization of the calibrated noise model using 30 ms exposure time and 15 dB gain to two other exposure settings with the same camera, which are 20 ms exposure time with 12 dB gain and 40 ms exposure time and 6 dB gain. The data on the plot are captured with the camera, while the noise models are obtained with our derived generalization equation, which is Eq. 2 of the main paper. This experiment validates both the noise calibration and the generalisation of the noise model.

3. Additional Details on Simulator Validation

We provide implementation details for the simulator validation experiments. The results of the experiments are illustrated in Fig. 3 of the main paper.

3.1. Image Statistics

This experiment reuses the same test target and the FLIR camera as the noise level calibration experiment shown in Fig. 1 (a). In this experiment, we strictly control the distance between the test target and the camera to 50 cm, and the illumination level at the test target as 2000 lux (measured with a light meter) using an LED panel light. The light is placed above the camera with an 80 cm distance and an angle of approximately 25°, facing downward to the test target.

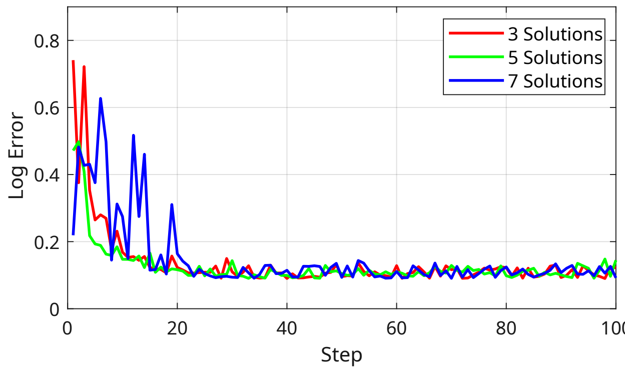


Figure 4. Training curves of 3 different solution numbers per generation, which are 3 solutions per generation, 5 solutions per generation (used in the main result), and 7 solutions per generation. The training curves indicate that all 3 settings can achieve a similar final performance, where the setting of 5 solutions per generation gives the smoothest and fastest convergence as it provides a search space that is more diverse compared to 3 solutions but also less to explore compared to 7 solutions. The experiment runs for 1000 steps but we only display the first 100 steps for visualization.

The same setup is then duplicated in Unreal Engine (UE), which is the simulator used in our experiments. In UE, the scene capture camera is set to have the same focal length, sensor size, pixel number, exposure time, and aperture size as the physical camera. However, we manually tune the ISO setting in the simulator to achieve the same brightness level as the physical camera. The rendered images are then applied with the noise model calibrated with the physical camera.

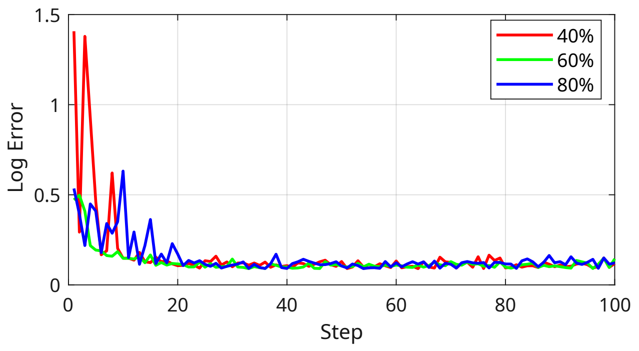


Figure 5. Training curves of 3 percentages of the population for offspring generation, which are 40% of the population, 60% of the population (used in the main result), and 80% of the population. The curves show that using 60% of solutions provides the fastest and smoothest convergence by balancing between exploration and exploitation, which is significantly beneficial for faster convergence. The experiment runs for 1000 steps but we only display the first 100 steps for visualization.

Table 1. Comparison of the optimized camera parameters for the depth estimation task using the genetic algorithm with 3 initializations, including initialising all the parameters at their smallest values in the design space, initializing all the parameters at their largest values in the design space, and initializing all the parameters at random values within in the design space, which is the method taken in the main paper. The results show that the optimization is insensitive to initialization.

Initialization	Camera Parameters		Performance	
	Baseline b (m)	Horizontal FOV fov (°)	Log Error ↓	RMSE ↓
Smallest	1.67 ●	50 ●	0.14 ●	80.7 ●
Largest	1.64 ●	50 ●	0.14 ●	78.05 ●
Random	1.6 ●	50 ●	0.14 ●	79.81 ●

3.2. Perception Task Performance

To validate our simulator in the performance of extracting Oriented FAST and Rotated BRIEF (ORB) [7] features, we adopt a test target used in [1], displayed in Fig. 3 (a), that is suitable for feature extraction. We use the same illumination setup in this experiment as described in Sec. 3.1. This experiment was conducted with the RGB camera of the Luxonis OAK-D Pro Wide camera, the FLIR Flea3 Camera, and the Basler Dart DaA1280-54uc camera for comparison, which are shown in Fig. 3 (b), (c), and (d). In addition, the background of this experiment is textureless to avoid additional features, and the test target is moved and captured at 10 locations along the same horizontal line to simulate a translational motion for feature matching and determining the number of inlier features, motion blur is not considered for this experiment due to a short exposure time.

In our UE simulator, we also duplicate the setup in the physical experiment. Similarly, we configure the scene capture camera to have the same focal lengths, sensor sizes, pixel numbers, exposure times, and aperture sizes as the three physical cameras, and we manually calibrate the ISO settings to match their gain values. We calibrate the noise models for these three cameras individually for this experiment and apply their noise models to the renders. The noise model calibration method follows Sec. 2.1.

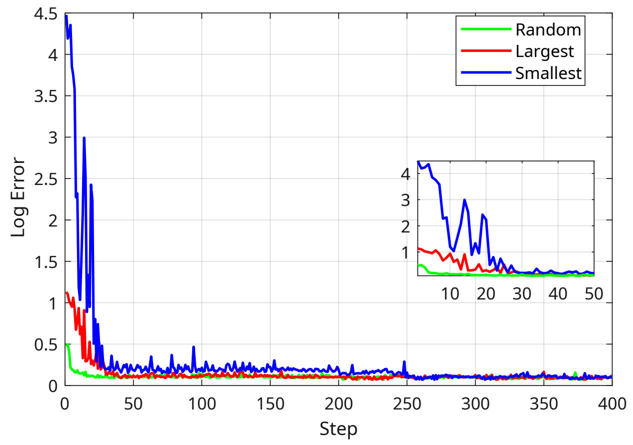
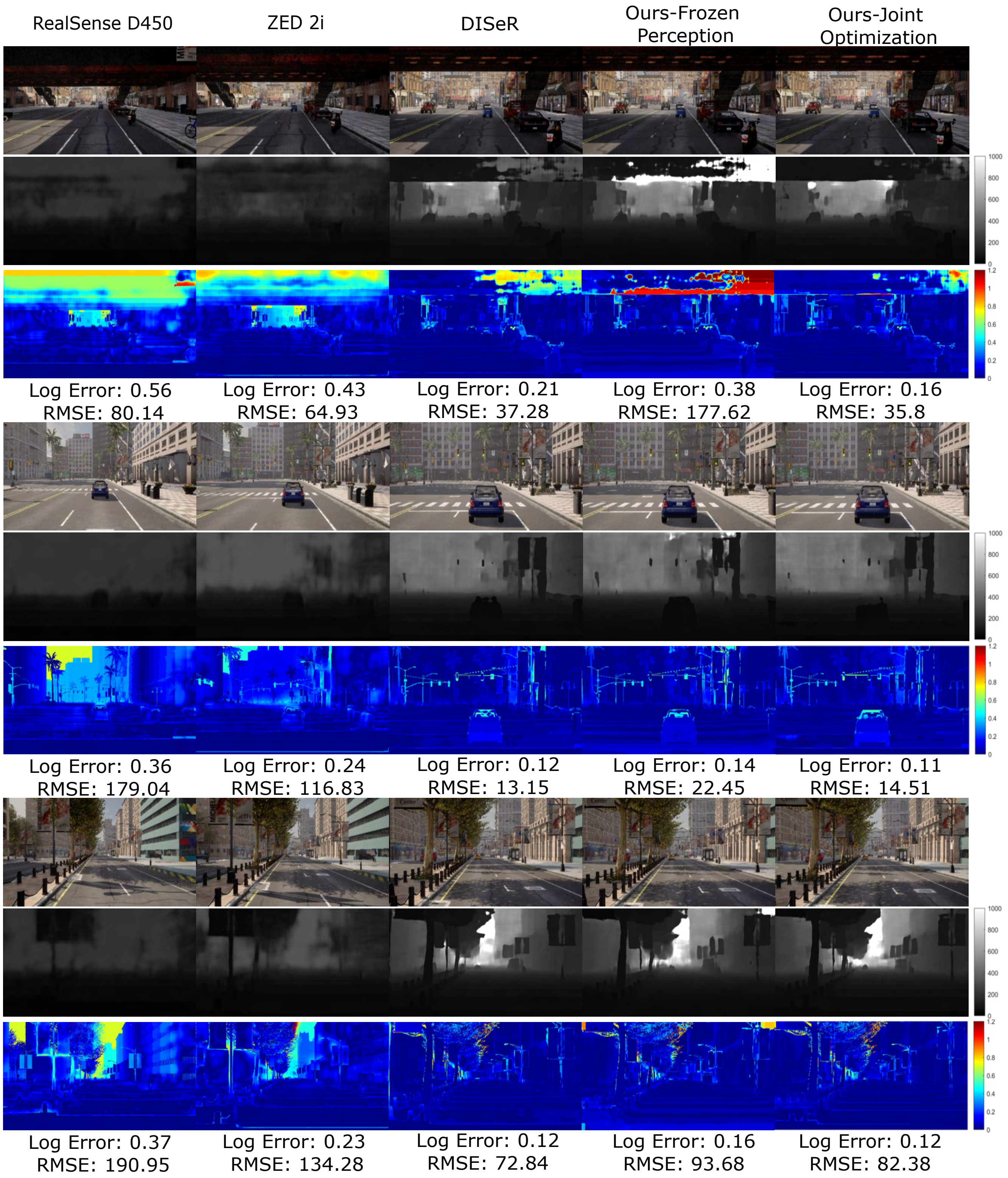


Figure 6. Training curves of 3 different initialization of camera parameters for the depth estimation task used by the genetic algorithm, which are starting from their smallest values, largest values, and random values within their design space. The curves show that the initialization does not affect the final performance but starting from random values results in faster convergence. The experiment runs for 1000 steps but we only display the first 400 steps for visualization.

4. Genetic Algorithm Implementation

4.1. Hyperparameter Selection

Population Size The number of solutions per generation is selected empirically based on the number of parameters to optimize. For example, we only optimize 2 camera parameters in the stereo camera design example so that we choose a relatively small solution number (5 solutions per generation), and for a more complex problem like the monocular camera design example, we use a larger solution number (10 solutions per generation). This scheme is selected since more complex design problems generally require more diverse solutions to search through a larger search space. However, a larger number of solutions takes longer to converge since a larger search space is explored. Conversely, a smaller solution number gives a less diverse search space and encounters the issue of local optima. Hence, we empirically select the number of solutions in this work as illustrated in Fig. 4, in which we compare the training curves of the depth estimation experiment with 3 different numbers of solutions per generation.



RealSense D450
ZED 2i
DISeR
Ours-Frozen Perception
Ours-Joint Optimization
Log Error: 0.56 RMSE: 80.14
Log Error: 0.43 RMSE: 64.93
Log Error: 0.21 RMSE: 37.28
Log Error: 0.38 RMSE: 177.62
Log Error: 0.16 RMSE: 35.8
Log Error: 0.36 RMSE: 179.04
Log Error: 0.24 RMSE: 116.83
Log Error: 0.12 RMSE: 13.15
Log Error: 0.14 RMSE: 22.45
Log Error: 0.11 RMSE: 14.51
Log Error: 0.37 RMSE: 190.95
Log Error: 0.23 RMSE: 134.28
Log Error: 0.12 RMSE: 72.84
Log Error: 0.16 RMSE: 93.68
Log Error: 0.12 RMSE: 82.38
1000
800
600
400
200
0
1.2
1
0.8
0.6
0.4
0.2
0

Figure 7. Comparison of captured left images, estimated depth maps, and log errors using cameras designed by our method with and without joint optimization, the RL method (DISeR), and off-the-shelf cameras, which are RealSense D450 and ZED2i. The depth and metrics are calculated in meters, and depth maps are capped at 1000 m. The off-the-shelf cameras fail with objects at long distances, whereas the cameras designed by our method and DISeR achieve desirable performance for all distance ranges.

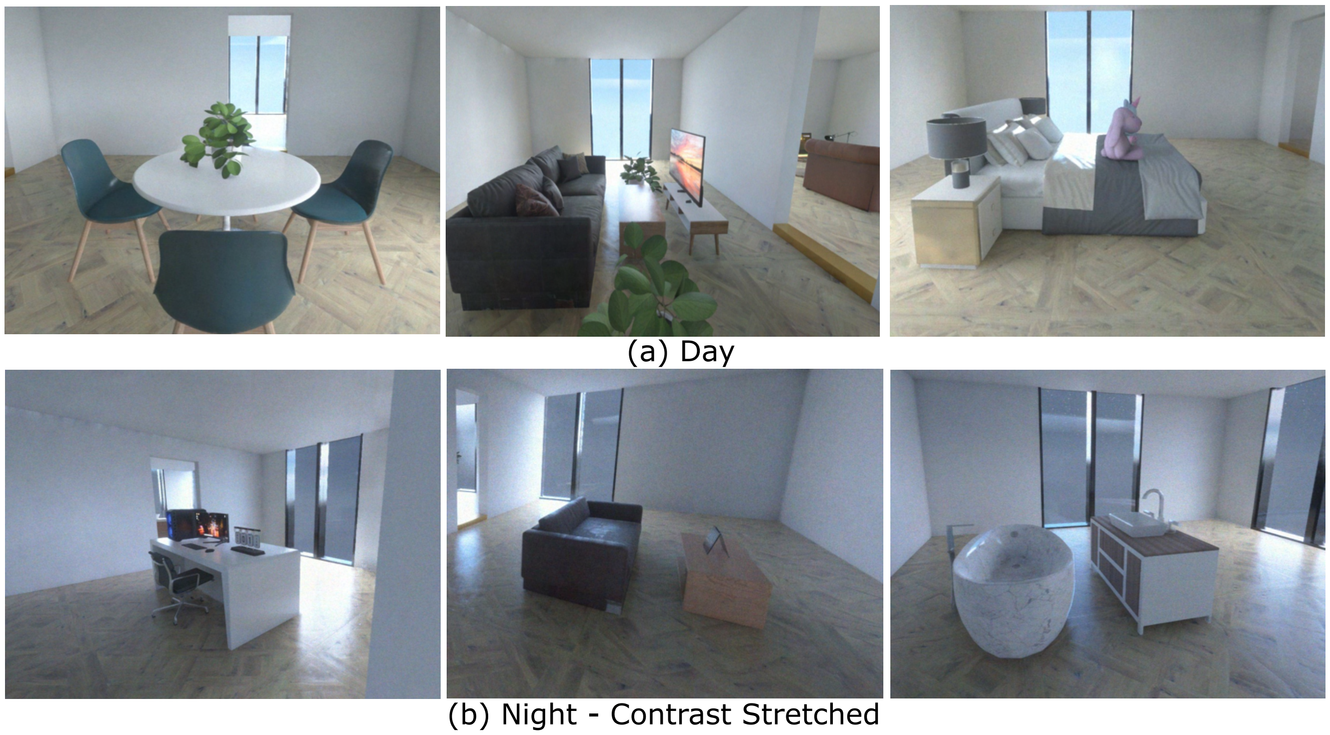


Figure 8. The example renders from our indoor virtual environment. The top 3 images (a) are captured in the daytime scenario (20 lux) using a lower camera gain (5 dB), and the lower 3 images (b) are captured in the nighttime scenario (2 lux) with a higher camera gain (15 dB). The nighttime images have stronger noise due to higher camera gain.

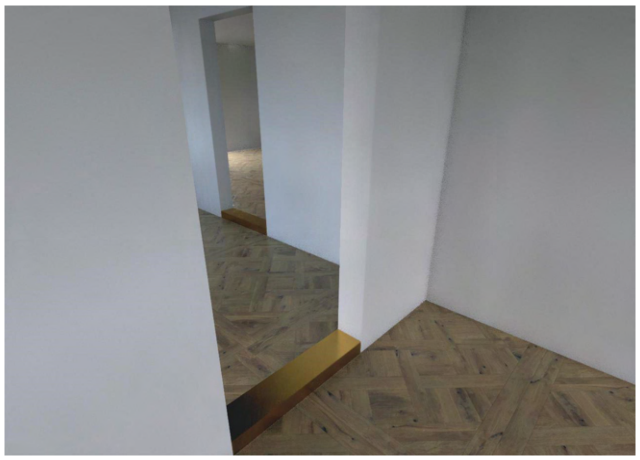


Figure 9. The low-height gold-colored thresholds act as obstacles that may pose a danger to the users.

Offspring Generation The offspring generation process contains three steps: parent selection, crossover, and mutation. We select the top solutions from the current generation as parents for offspring generation. The number of parents is chosen as 50% or 60% of the size of the population, 50% for an even number of solutions and 60% for an odd number of solutions. For example, we use the top 3 solutions out of a total of 5 solutions per generation for the stereo camera design problem and use the top 5 solutions out of a total of 10 solutions per generation for the monocular design problem. We compare the optimization curves from 3 different portions of populations used for offspring generation (40%, 60%, 80%) in the depth estimation task. The results are displayed in Fig. 5, which indicate that although the final performance is not affected when using a large number of training steps, using 60% of the population to generate offspring yields the fastest and smoothest convergence. Since using 40% of the population as parents emphasizes exploitation over exploration, the optimizer may get stuck at local optima with fewer training steps. On the other hand, using 80% of the population as parents creates more diverse offspring, which emphasizes exploration over exploitation and leads to slower convergence. Hence, we choose an intermediate percentage for offspring generation that is roughly half of the solution number per generation.

We then apply a uniform crossover scheme using the selected parents to produce the offspring for the next generation of solutions, indicating that each parameter for the offspring is randomly selected from the parents. For mutation, we apply a multiplication factor from the same range, which is a random number between 0.8 to 1.2, for all the parameters in our experiments and apply an addition value

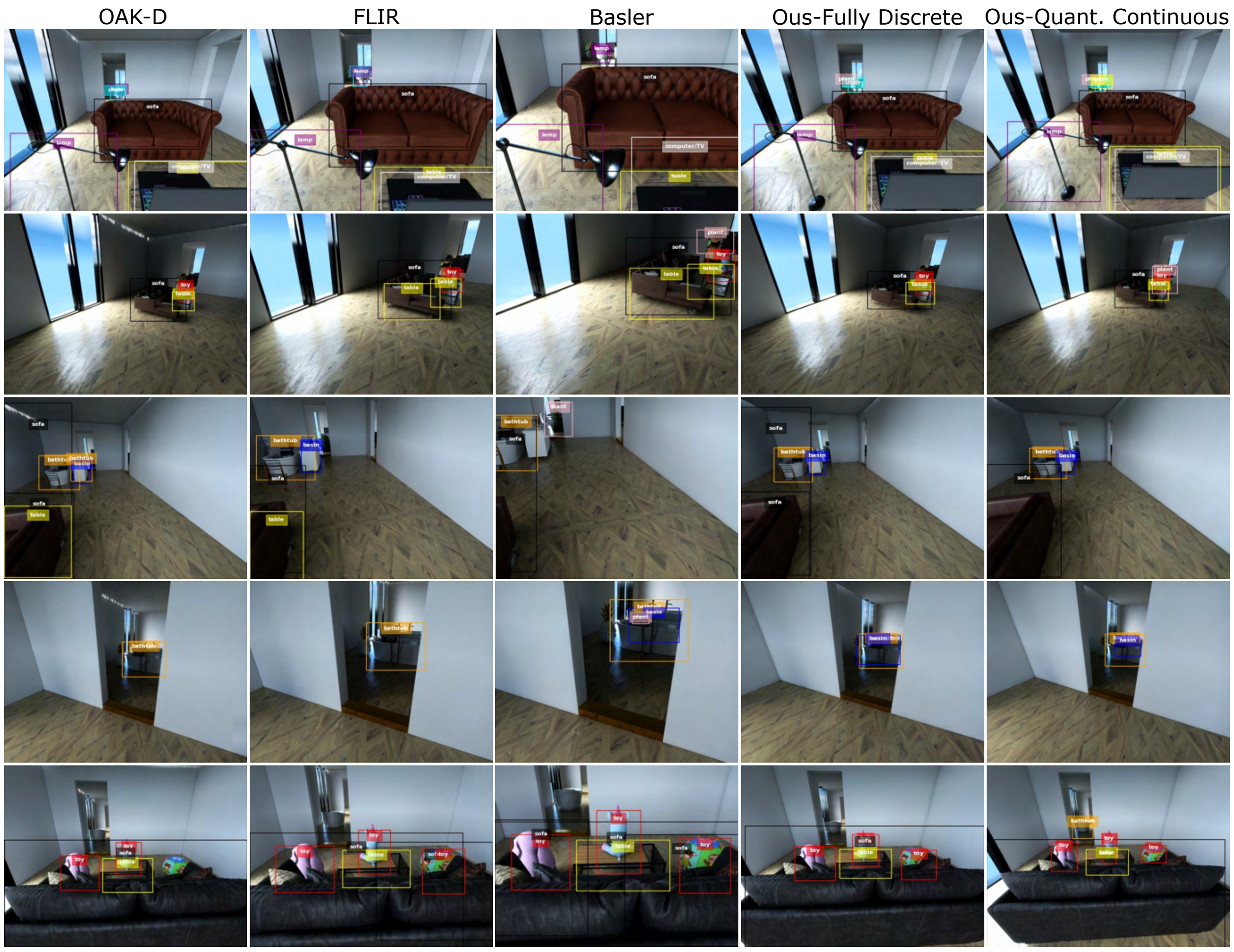


Figure 10. Comparison of object detection performance using cameras designed by our method and the off-the-shelf cameras. The cameras designed by our method show improved performance with small objects, objects at long distances, and objects that are partly occluded by optimizing the FOV and pixel size to obtain a more suitable effective resolution and signal-to-noise ratio for the task.

whose range is customised for different parameters depending on their available design space. All these hyperparameters used to implement the genetic algorithm are selected empirically.

4.2. Parameter Initialization

In our experiments, all the camera parameters that are optimized by the genetic algorithm are initialized with random values within their design space. However, we compare the optimization results using the proposed method with different initialization by setting the camera parameters in the depth estimation task to be initialized at their smallest values (baseline: 0.01 m, FOV: 50°) and their largest values in the design space (baseline: 3 m, FOV: 120°).

The results are displayed in Tab. 1. The experiment uses the same hyperparameters for the genetic algorithm as described in Sec. 4.1 to optimize the camera parameters. The perception network is jointly trained during optimization, indicated by ● in the table. The results indicate that the initialization of camera parameters does not have a significant impact on the final camera parameters and the downstream task performance.

In addition, we illustrate the training curves of the above-mentioned 3 camera parameter initialization schemes in Fig. 6, which shows that random generation results in faster convergence compared to initialising from extreme values. This is because the optimal solution is usually not the extreme values, and starting from random values between the extremes gives values that are relatively closer to the optimal solution. It is observed that starting from the smallest parameter values results in the slowest convergence. This

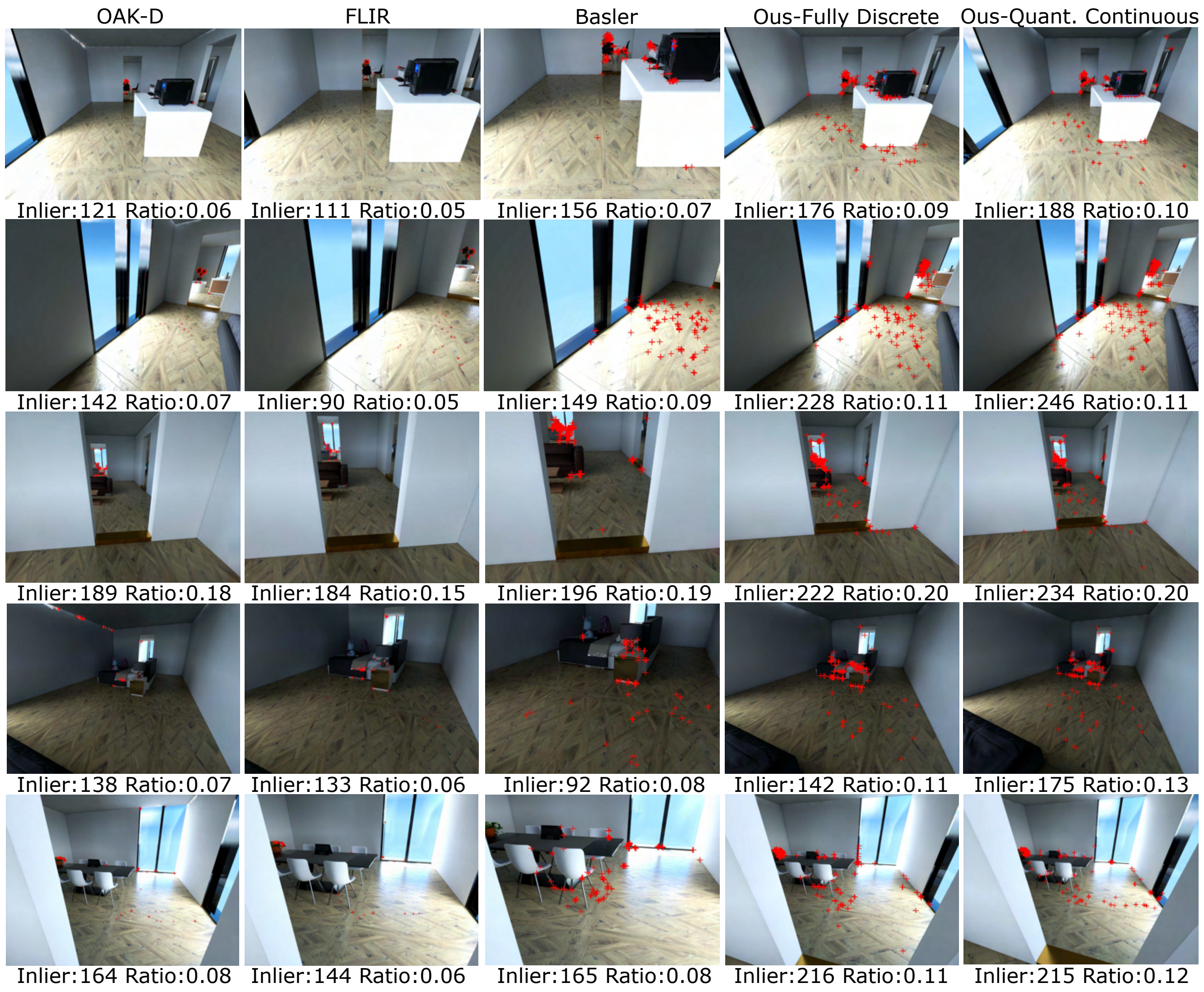


Figure 11. Comparison of feature extraction and matching with images captured by cameras designed by the proposed method and the off-the-shelf cameras. We display the features that are successfully matched with the features in the next frame and filtered (inliers) on the images, which shows that the cameras designed by our method contain the highest number of inliers.

is because the offspring at every iteration is generated from the top solutions of the previous generation with a mutation process, where the parent solutions are multiplied by a random factor as the first mutation step. Therefore, smaller parameters change on a smaller scale compared to larger values, which slows down the exploration process and leads to slower convergence.

5. Additional Results on Stereo Camera Design

We provide additional qualitative results for our stereo camera design experiment in Fig. 7. The figure displays the captured left images, the estimated depths, and the log error between the estimated maps and the ground-truth maps. We compare the images and results by using the stereo cameras designed by the proposed method with and without joint optimization, the Reinforcement Learning (RL) method (DISeR) [4], and two off-the-shelf cameras, which are Intel RealSense D450 and ZED 2i. The configurations of these cameras are listed in Tab. 1 of the main paper. The metrics displayed in this figure are the same as Tab. 1 of the main paper, which is log error and RMSE error in meters.

The results show that in our application scenario of the outdoor environment, the off-the-shelf cameras get low performance since their baselines are relatively low (0.095 m and 0.12 m), but many objects, such as the buildings and the footbridge, are far from the stereo camera. However, it is observed that these off-the-shelf cameras perform well in short distances, indicating that they can be beneficial for an application that does not involve long-distance objects. On the contrary, the cameras designed by our method and DISeR perform well across both long and short distances.

6. Additional Details on Monocular Camera Design

6.1. Environment

We construct the indoor virtual environment in UE 5 with a procedural generation technique, which generates random floorplans and object locations. The size of the environment is configured to be 15 m in width and length and 3 m in height for our experiment, and adjusting to different dimensions is trivial. However, each room in the environment needs to have a minimum length of 5 m to fit the furniture. The environment contains objects from 10 classes, which are sourced from the UE marketplace, encompassing 10 classes: sofa, bed, table, chair, bathtub, bathroom basins, computer/TV, plant, lamp, and toy.

We illustrate some example renders from our virtual environment in Fig. 8, including a comparison of the day and night design scenarios used to validate our method.

6.2. Image Sensor Catalog

The image sensor catalog we collected contains 43 image sensors, 28 of which are from Sony, 10 from Onsemi, 3 from Luxima, and 2 from CMOSIS. The pixel sizes of these sensors vary from 1.12 μm to 9 μm. The smallest sensor has a dimension of 3.07 mm$\times$2.3 mm, while the largest has a dimension of 16.13 mm$\times$12.04 mm.

6.3. Obstacle Placement

To restrict the camera's Field-of-View (FOV), we place low-height gold-colored thresholds at the entrance of all the rooms in our virtual environment. An example of the threshold is shown in Fig. 9. The thresholds act as obstacles that may put the users at risk. They are configured to be interactive actors in UE, making the auto-agent aware of them even though they are not captured within the FOV of the scene capture camera.

6.4. Qualitative Results

We visualise the performance of object detection, as well as the feature extraction and matching task, with images captured by the cameras designed by the proposed method and the off-the-shelf machine/robotic cameras in Fig. 10 and Fig. 11 respectively. The off-the-shelf cameras are the same cameras used to validate our simulator in Sec. 3.2, which are the OAK-D Pro Wide camera, FLIR Flea3 camera, and Basler Dart camera.

References

[1] Donald G Dansereau, Bernd Girod, and Gordon Wetzstein. LiFF: Light field features in scale and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8042–8051, 2019. 2, 3

[2] FLIR Integrated Imaging Solutions Inc. *FLIR FLEA®3 USB3 Vision*, 1 2017. Rev. 8.1. 1

[3] Alessandro Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009. 1

[4] Tzofi Klinghoffer, Kushagra Tiwary, Nikhil Behari, Bhavya Agrawalla, and Ramesh Raskar. DISeR: Designing imaging systems with reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23632–23642, 2023. 7

[5] Ce Liu, William T Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 901–908. IEEE, 2006. 1

[6] Netanel Ratner and Yoav Y Schechner. Illumination multiplexing within fundamental limits. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 1

[7] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. 3

[8] Taihua Wang and Donald G Dansereau. Multiplexed illumination for classifying visually similar objects. *Applied Optics*, 60(10):B23–B31, 2021. 1