AutoSCOOP: Automated Road-Side Sensor Coverage Optimization for Robotic Vehicles on Proving Grounds

1st David Hermann
Computer Science
Technische Universität München
Munich, Germany
david.hermann@tum.de

2nd Clara Marina Martínez
Virtual Vehicle Development
Bietigheim-Bissingen, Germany

3rd Frank Sayer
Virtual Vehicle Development
Bietigheim-Bissingen, Germany

4th Gereon Hinz
Robotics, Artificial Intelligence and Real-time Systems
Technische Universität München
Munich, Germany
gereon.hinz@tum.de

5th Alois Knoll
Robotics, Artificial Intelligence and Real-time Systems
Technische Universität München
Munich, Germany
knoll@mytum.de

***Abstract*—The use of automated vehicle testing on proving grounds is increasing to enable time and cost-effective testing and reduce risks to test drivers. Robot test vehicles are used to perform various functions and load tests, even under severe conditions. Therefore, to ensure safety in proving grounds, perception and monitoring of surrounding vehicles are necessary. This requires a target-oriented, robust and foresighted perception based on road-side systems, due to the fact that test vehicles' on-board sensors are generally insufficient and short-sighted. Such a challenging sensor system has to take into account area-wide coverage, high detection probability, and low cost, for complex areas. To address this problem, we introduce AutoSCOOP, a novel method to automatically optimize sensor coverage on proving grounds. AutoSCOOP uses ray-cast sensor models and a detailed 3D environment model in a game engine to determine accurate and realistic sensor coverage. In combination with an evolutionary strategy-based method, an optimization is performed to find the optimal placement and number of road-side sensors. The methodology is successfully applied to an environmental model based on a real proving ground, and experimental evaluations are presented to show that full coverage is achieved with a minimal number of sensors.**

***Index Terms*—Optimal Sensor Coverage, Proving Grounds, Evolutionary Strategy, Game Engine, Ray-cast, Road-Side Sensor**

I. Introduction

Before vehicles are ready for production, they must be extensively tested to ensure they meet legislative and safety standard requirements. Testing is time-consuming, involves thousands of test kilometers and combines proving ground with free road driving [1]. Safety-critical testing, usually involving high-speed or risky maneuvers, takes place in controlled environments on testing tracks.

Various studies show that automated testing with robot vehicles reduces risks and costs [2], [3]. Besides, automation guarantees test reproducibility, improves safety and reduces test duration. However, it requires monitoring of the surroundings of the robot vehicles to ensure a safe and collision-free trajectory. Monitoring and safeguarding cannot be reliant upon on-board sensors, as they have a limited field of view (FoV) and their specifications depend on each test vehicle.

A solution to significantly increase foresight and robust perception even under severe weather conditions is to extend on-board sensors with road-side sensors, e.g., camera, lidar, and radar, as demonstrated in [4], [5], [6]. Placing road-side sensors on proving grounds for robust surveillance and at low cost is challenging due to topography, buildings, road geometry and existing infrastructure. The sensor coverage area is strongly influenced by the environment. Objects and topography, for example, lead to limitations in visibility and gaps in coverage. Consequently, determining the most suitable sensor number and position becomes a complex optimization problem that needs to deal with all possible solutions and fulfill all proving ground requirements and conflicting criteria.

This problem requires a target-oriented automated placement of the road-side sensors able to take all these conditions into account and determine an optimal result based on a realistic sensor coverage. In this paper, a preliminary design for multi-sensor systems using a new method called AutoSCOOP (Automated Sensor Coverage Optimization) is shown to determine and evaluate realistic coverage. This method uses sensor models and a virtual environment for sensor distribution optimization, which takes the previous boundary conditions into account.

II. Related Work

Optimizing sensor positioning and orientation for outdoor surveillance based on camera sensors has been explored for different applications in many studies [7], [8], [9].

The environment is partially represented by the Two-dimensional (2D) or Three-dimensional (3D) area to be monitored, in which sensors can be placed. It usually contains objects or boundaries that can cause occlusion in the sensor FoV. Two-dimensional grid-based environments are proposed in [10], [11], [8]. These environments can approximate occlusion and are simple enough to apply powerful optimization methods. However, these approaches imply strong environment simplifications, undetailed maps, unwanted occlusions, and unobstructed views, making it difficult to determine the quality of coverage transferred to a complex real area.

In [12], 3D environments are used to monitor ground-level airspace in urban areas again with the goal of sensor coverage. This environment is based on a precise mesh and allows for the representation of complex environments and thus detailed sensor coverage. A 3D environment for perception in autonomous vehicle applications is also presented in [13]. In this article, the authors make use of game engine technology to power the simulation environment.

The sensor FoV depends on the sensor specifications as well as the sensor parameters and is crucial for the representation of the area that can be covered. In previous work, the FoV is often assumed to be a geometric surface to simplify the optimization problem. Usual shapes are circular sectors [10], [11], triangles and trapezoids [14]. A more realistic representation is proposed in [15] using probabilistic distance- and angle-dependent sensing models. This approach provides the necessary differentiability for the optimization and describes realistic transition at the edges of the sensor FoV.

A further complexity degree is presented in [16], where the authors propose using ray-cast sensor models. Ray casting is a common technique for map scanning and to get occlusion areas. Sensors can be modeled with a number of rays, which shoot in specific directions, and can emulate occlusion, object detection and blind spots. This generates a visibility matrix that is used in [16] as input for the optimization.

Eventually, the sensor network can be designed by solving an optimization problem to maximize environmental coverage while minimizing the number of sensors. Metaheuristic methods for optimization such as genetic algorithm [17], [18] and particle swarm method [14], [8] are used to maximize coverage of an area for a given number of sensors. Other approaches implement the Global Greedy Algorithm either combined with metaheuristic approaches [18] or alternated with Greedy Grid Voting with predefined camera locations [19]. An alternative method is proposed in [16], where Binary Integer Programming is applied with constraints on the sensor distances and the coverage of the target points.

The consideration of realistic constraints, taking into account an accurate sensor coverage for optimization and implementation on real grounds, remains a challenge.

III. Methodology

Previous work only allows limited predictions about the accuracy of coverage for real outdoor applications due to simplifications in the modeling of the sensor coverage and the environment. Influences by topography or 3D objects on the sensor FoV are only insufficiently taken into account. In addition, requirements and constraints of real infrastructures are not or only partially considered in sensor placement.

Therefore, we propose a novel method for automated sensor coverage optimization based on a 3D environment and accurate sensor coverage areas, enabling more precise prediction for a real outdoor application. Furthermore, all relevant degrees of freedom and constraints of the sensors are included in the optimization to achieve an optimal result in terms of full coverage with a minimum number of sensors.

First, we define the problem of searching for an optimal solution. Then we introduce our methodology to address the challenge based on dynamic sensor coverage and optimization method.

A. Problem Definition

The goal of this work is the automatic placement and orientation of the minimum number of sensors to surveil a given area. Thus, the degrees of freedom of each sensor n are given by the position in x, y and z directions and rotations pitch ϕ, roll θ and yaw ψ within a georeferenced local coordinate system. This can be simplified into the solution vector $\vec{p}_{\text{opt}}(n) = (x, y, \phi, \psi)$ as the sensor roll does not affect coverage and the sensor height z is constrained by the infrastructure. The optimum of $\vec{p}_{\text{opt}}(n)$ with $\min n$ for full coverage of the area has to be found given the boundary conditions.

Considering the environmental conditions and constraints, two areas are specified, target and sensor area. The target area is defined as $\mathbf{b}_t = [\vec{x}_r, \vec{y}_r]$ for r reference points, which is to be monitored and the sensor area limits where sensors can be positioned. Depending on the existing infrastructure and environmental conditions n_s number of sensor areas can be defined. They are usually selected as close as possible to target areas. The boundary of the sensor areas are defined by j reference points $\mathbf{b}_{s,i} = [\vec{x}_{i,j}, \vec{y}_{i,j}]$ for $i = \{1, 2, \cdots, n_s\}$ in the xy-plane.

B. Dynamic Sensor Coverage

The accuracy of sensor coverage depends on two factors, the environment model and the sensor model. As the level of detail of these two factors increases, a more realistic representation is achieved.

In this work, we base our sensor coverage calculation on the open-source simulator CARLA [20] in combination with Epic Games' open-source game engine, Unreal [21]. This simulation environment provides the accuracy for appropriate, detailed sensor coverage based on a 3D environment built in Unreal and a ray-cast sensor model.

A model of a real proving ground based on accurate environmental data for automated testing is shown in Fig. 1. It describes details of the proving ground, such as track geometry, elevation, infrastructure and vegetation that are relevant for sensor placement. Based on the 3D environment, the sensor and target areas are locally defined, as shown in

Fig. 1: Proving ground sections modeled in Unreal with various test routes, intersections, and safety-related structures.

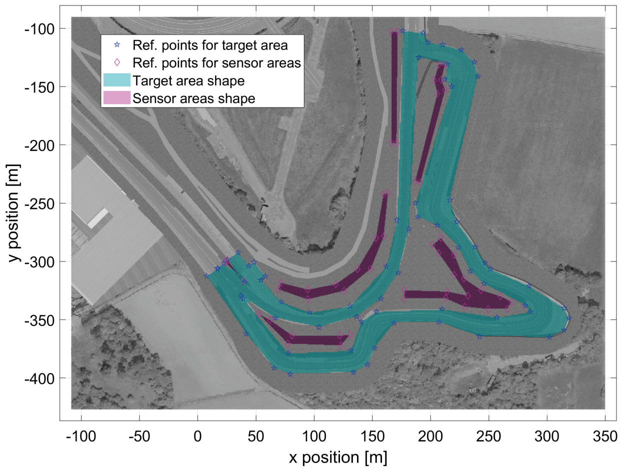

Fig. 2: Virtual section top view of a real proving ground with two routes. Overlaid are the sensor $(\mathbf{b}_{s,i})$ and the target points $(\mathbf{b}_t)$, as well as the corresponding surface shapes.

Fig. 2. A subsequent discretization with an equidistant grid of the target area leads to $\mathbf{A}_t = [\vec{x}_t, \vec{y}_t]$ and is used for the later evaluation of the sensor coverage.

The combination of the environment model and ray-cast sensor models allows coverage to be calculated dynamically with the position and orientation of the sensors. Ray-cast sensor models are suitable for different sensor technologies, including cameras, lidars and radars. Hereby each ray-cast model is parametrized for horizontal α_H and vertical α_H FoV, range and number of rays casted per second, that is sample points, and represent a typical image sensor. Each ray-cast model returns a 3D point cloud with the relative coordinate vectors to the sensor that discretizes the environment.

For simplification, the projection of the point cloud in the xy-plane is used in the following. The sensor coverage is calculated as a function of the point cloud density and implements a minimum probability of detection. Fig. 3 illustrates a filtered sensor FoV, including sensor shadow areas caused by a wall.

Next, we calculate the α-shape v_s according to [22] based on the outer FoV points, to determine later the points on $\mathbf{A}_t$ that lie within the sensor FoV. This results in an envelope of α-shape that can be adjusted to provide a more accurate representation of coverage.

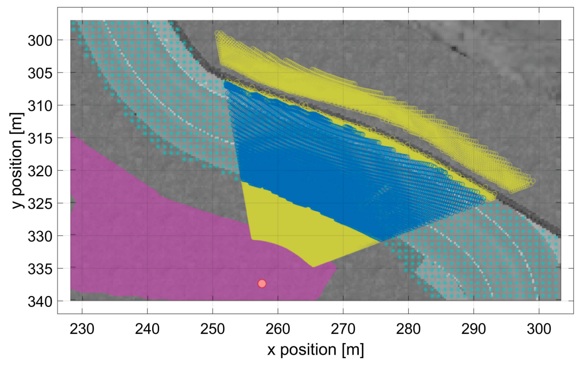

Fig. 3: Coverage sections of a $80°$ horizontal and vertical viewing angle mounted at $7m$ height on a track section with sensor position (red dot), target area (turquoise stars), sensor area (purple area), sensor FoV outside the target area (yellow circles), and sensor area inside the target area (blue circles).

C. *Optimization Method*

The optimization of the sensor distribution in AutoSCOOP is based on the Evolutionary Strategy (ES). ES can search for an optimum in n-dimensional search space, which can be easily extended, without prior application knowledge [23]. Furthermore, ES offers low dependencies on initial conditions and has no restrictive requirements on the objective function. For AutoSCOOP, the ES algorithm is adapted in the following for fast and optimum sensor coverage to take into account the defined constraints and sequentially optimize the sensors.

In this population-based method, individuals include parents μ and their offspring λ as well as their behavioral characteristics $\vec{p}_\mu$ and $\vec{p}_\lambda$ derived from the solution vector. Like in ES, AutoSCOOP optimization consists of four steps to optimize one sensor: reproduction, mutation, evaluation, and selection.

Reproduction of n_μ individuals gives n_λ new individuals with $\vec{p}_\lambda$. Mutations of (λ, μ) with widths $\vec{o}_\lambda(t)$, $\vec{o}_\mu(t)$ are determined for each element in $\vec{p}_{\text{opt}}$ allowing selective adaptation of individuals with Gaussian phenotypic change. The mutation width is obtained using a Gaussian distribution $N(0, \vec{\sigma}^2)$ with variance $\vec{\sigma}$. In the AutoSCOOP algorithm, $\vec{o}_\lambda(t)$, $\vec{o}_\mu(t)$ are saturated to consider the boundary conditions, such as sensor area or pitch angle constraints. In the evaluation step, the fitness value for each individual (λ, μ), is calculated as follows:

1) The first fitness function evaluates the sensor coverage:

$$f_{ca} = \begin{cases} \sum_{i=1}^{q} \chi(i), & \text{if } \mathbf{A}_t(i) \text{ inside } v_s \\ \zeta, & \text{if } \mathbf{A}_t(i) \text{ is covered by multiple sensors} \\ \sum_{j=1}^{h} \tau(j), & \text{otherwise} \end{cases} \quad (1)$$

depending on target area points $\mathbf{A}_t(i)$ with $i = \{1, 2, \cdots, q\}$, the sensor point cloud density $\chi(i)$, the coverage overlap constant ζ and the function $\tau(j)$. The density is used here to score areas with a high point cloud density of the sensor better, as they have a higher probability of detection. In the third case, for each grid point h within the sensor coverage and outside of $\mathbf{b}_t$, a fitness value depending on

$$\tau(j) = \exp\left(-\frac{d_{pp}(j)^2}{2\vartheta^2}\right) \tag{2}$$

determined as a function of the distance $d_{pp}(j)$ with $j = \{1, 2, \cdots, h\}$ to the nearest point on the target surface. The distance can be additionally weighted with the constant ϑ.

2) The second fitness function alludes to the minimum distances $d_{\text{sensor}}(i)$ of the sensors $(\mu + \lambda)$ to the already optimized sensors $\vec{p}_{\text{opt}}$:

$$f_{ds} = \frac{f_{ca}}{z} \cdot \sum_{i=1}^{z} \exp\left(-\frac{d_{\text{sensor}}(i)^2}{2\vartheta^2}\right) \tag{3}$$

3) The third fitness function evaluates for seamless coverage by the minimum Euclidean distance $d_{\text{sensor\_area}}(i)$ of the centroid of gravity of the FoV area to the optimized sensors:

$$f_{sc} = \frac{f_{ca}}{z} \cdot \sum_{i=1}^{z} \exp\left(-\frac{d_{\text{sensor\_area}}(i)^2}{2\vartheta^2}\right) \tag{4}$$

The dependence in (3) and (4) on f_{ca} is for prioritization of target area coverage.

4) Finally, the total fitness value is calculated with:

$$f_s = \begin{cases} f_{ca} \cdot w_{ca}, & \text{n = 1} \\ f_{ca} \cdot w_{ca} + f_{ds} \cdot w_{ds} + f_{sc} \cdot w_{sc}, & \text{n >1} \end{cases} \tag{5}$$

where the weights w_{ca}, w_{ds} and w_{sc} are used to prioritize the individual fitness functions and are adjusted according to the boundary conditions.

In the final step, selection, k number of sensors with the best f_s values are selected for the next generation as μ.

The termination criterion is set based on the convergence ϵ_{sensor} of the individuals and the threshold ϵ_{thres} in the current epoch t. The convergence depends on the individuals with the best and worst fitness values. Once an optimum is found for $\vec{p}_{\text{opt}}$, the sensor is fixed and the optimization of the next sensor initiates. The optimal number of sensors is achieved when the ratio of the target area a_t and the already covered target area by optimized sensors a_c exceeds the threshold ϱ_{thres}. Finally, the clustering of sensors close to each other is performed. The AutoSCOOP algorithm is shown in the following table Algorithm 1.

Algorithm 1: AutoSCOOP Algorithm

```
input : $\mathbf{b}_{s,i}$, $\mathbf{A}_t$, $n_\mu$, $n_\lambda$
output: $n$ sensors with solution vector $\vec{p}_{\text{opt}}$
begin
    Sensor counter setting $n \leftarrow 1$ ;
    while $\frac{a_t}{a_c} < \varrho_{thres}$ do
        Generation of $n_\mu$ individuals with random $\vec{p}_\mu$ ;
        while $\epsilon_{sensor} > \epsilon_{thres}$ do
            $\lambda \leftarrow \mu$ with $\vec{p}_\lambda$ random $n_\lambda$ number ;
            Mutation of $\vec{p}_\lambda \leftarrow \vec{p}_\lambda + \vec{o}_\lambda$ ;
            Mutation of $\vec{p}_\mu \leftarrow \vec{p}_\mu + \vec{o}_\mu$ ;
            if $\lambda$, $\mu$ are outside of $\mathbf{b}_{s,i}$ then
                Saturation of $\vec{o}_\lambda$, $\vec{o}_\mu$ ;
                New Mutation step ;
            end
            Determination of FoV $v_s$ for $\lambda$, $\mu$ ;
            Fitness calculation $f_s(v_s, \mathbf{A}_t)$ ;
            Selection of $k$ sensors with best $f_s$ ;
        end
        $\vec{p}_{\text{opt}}(n) \leftarrow$ sensor with best $f_s$ ;
        Fixed placement of $\vec{p}_{\text{opt}}$ in map ;
        $n \leftarrow n + 1$ ;
    end
    Clustering of sensors $\vec{p}_{\text{opt}}$ ;
    return $\vec{p}_{\text{opt,n}}$
end
```

IV. Simulation Experiment

A. Setup Environment

AutoSCOOP is created using CARLA version 0.9.11, connected to MATLAB R2021a where it is implemented. For the evaluation, we use the environment model Fig. 2 with 6 predefined sensor areas and a target area discretized with $1m$x$1m$ grid. The ray-cast sensor model is parameterized according to Tab. I based on a typical camera. The effective range of the sensor is chosen so that monitoring is possible even in poor visibility conditions. The sensor angle ψ is further restricted upper and lower depending on the vertical FoV angle of the sensor.

TABLE I: Parameterset for Sensor

α_H *FoV*	α_V *FoV*	*Eff. Range*	z	*Sample Points*	ϕ *Range*
80°	80°	$80m$	$7m$	$30720pps$[a]	[31° 70°]

[a]points per seconds

TABLE II: Parameterset for the AutoSCOOP algorithm

Parameter	λ	μ	ϱ_{thres}	ϵ_{thres}	w_{ca}	w_{ds}	w_{sc}
Value	20	40	97%	0.2	1	0.5	2.5

AutoSCOOP parameters are listed in Tab. II. The parameter λ depends on the size of the search space, i.e. the length of the solution vector. We chose the parameter to achieve a good compromise between fine coverage of the search space and acceptable computation time due to the number of individuals. Here, full coverage was achieved when at least 97% of the target area was covered.

B. Results and Discussion

The evaluation of the AutoSCOOP algorithm is based on the optimized solution vectors in Table Tab. III using the following criteria: cluster ratio, relative coverage, number of epochs, and overlapping factor.

AutoSCOOP algorithm achieves coverage of 97.61% with 27 sensors in the previously detailed setup, Fig. 4. The sensors were distributed to 18 different sensor locations, resulting in a cluster ratio of ≈0.66 with the number of sensors. A cluster ratio of 1 means that all sensors have different locations and thus require extensive infrastructure. Placing multiple sensors in one location reduces costs by using one sensor mast and simultaneous wiring for multiple sensors.

The relative sensor coverage is given by the quotient of a reference sensor area of $1229.95 m^2$, the maximum coverage on flat terrain, and the mean value of the individual sensor coverage of $1236.02 m^2$. The higher the quotient, the more effective the sensors are and cover a larger area. The quotient here is 1.005, indicating effective use of the sensor areas. On average, the covered area is slightly higher than the reference area, which can be explained by a positive terrain elevation and thus an even larger sensor range.

Here, it is shown that the algorithm takes the environment into account via the sensor coverage during the optimization and places the sensors in such a way that occlusion of the target area by objects is avoided. In addition, the optimization finds solutions where the target area is effectively covered even with complex shapes. This can be seen in Fig. 4 at the narrow point of the two routes, where a sensor covers more than one route of the target area at the same time.

TABLE III: Optimized sensor solution vectors, fitness values and covered target area

Sensor n	x $[m]$	y $[m]$	ϕ $[°]$	ψ $[°]$	***Fitness*** [-]	***TA***[a] ***covered*** $[m^2]$
1	169.1	-125.5	35.9	339.2	2823.0	1007
2	169.1	-125.5	33.0	45.9	5862.8	822
3	204.7	-139.5	37.9	48.1	5949.0	1080
4	73.8	-322.4	31.0	174.3	2519.1	1295
5	73.8	-322.4	33.5	94.8	6603.1	724
6	78.4	-361.8	34.4	54.6	6066.7	1210
7	78.4	-361.8	42.2	156.3	4629.2	599
8	196.6	-193.1	37.5	50.1	4971.6	1082
9	111.3	-363.7	37.7	345.3	5129.1	1035
10	187.7	-227.0	39.6	84.8	6186.0	1230
11	162.5	-242.0	31.8	325.7	4647.7	834
12	97.6	-331.4	31.1	57.6	4038.4	622
13	26.4	-302.6	31.0	100.0	3196.8	877
14	160.0	-273.1	37.8	53.6	3398.0	542
15	208.7	-282.3	40.6	3.4	3564.3	614
16	232.4	-317.4	35.1	71.4	4203.9	1040
17	248.5	-326.5	31.4	355.4	6097.1	1051
18	220.5	-335.0	36.2	132.5	4169.4	534
19	204.7	-139.5	38.4	325.7	2690.7	583
20	232.4	-317.4	31.0	353.3	2415.5	632
21	111.3	-363.7	38.1	187.1	1728.8	954
22	269.0	-335.0	31.9	41.1	2696.0	1057
23	187.7	-227.0	34.3	284.1	2341.9	923
24	130.0	-363.2	31.1	312.6	2241.7	819
25	26.4	-302.6	35.7	339.5	1727.8	187
26	26.4	-302.6	31.0	145.9	1355.7	201
27	191.2	-214.0	31.0	86.3	1659.5	1063

[a]Target Area

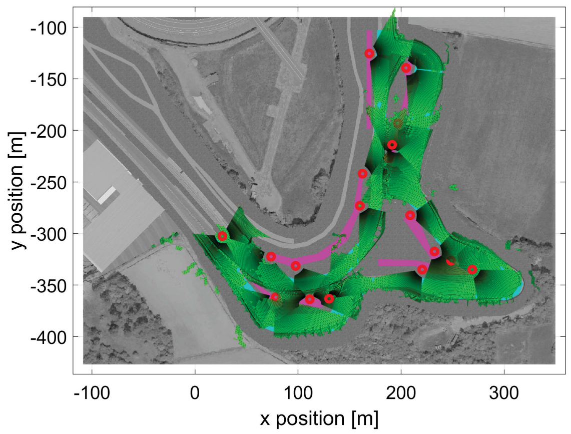

Fig. 4: Visualization of the optimization result with the v_s for the sensors in green and the corresponding sensor positions shown as red circles.

From the table Tab. III it can be seen that certain sensors, such as #25 or #26, cover a small part of the target area exclusively. This is the result of penalizing covering same areas with several sensors and encouraging coverage of small area gaps between two consecutive already placed sensors. According to (1), an individual is rated better if an uncovered target area is covered with a high point cloud density.

On average, 26.96 epochs are required until the optimization converges to a solution for sensor placement. This demonstrates the robustness of the optimization method, allowing a solution to be found even for complex areas. The pitch angle constraint and the limitation of the sensor areas for the individuals are taken into account when saturating the mutation width and also lead to valid results.

The covered target areas from Table Tab. III are used to determine the overlapping factor by taking the quotient of the sum of these areas and the total target area by $17221 m^2$, which returns 1.31. A factor higher than 1 indicated overlapping, which is desired to some extent for gapless coverage, also shown in (4).

With the required overlapping and the high relative sensor coverage, it also shows that a minimum number of sensors were found to achieve the target coverage.

V. Conclusion and Future Work

Optimal road-side sensor placement is important for automated tests on proving grounds to achieve a far-reaching and robust perception. The challenge consists of automatic placement of road-side sensors even on complex areas considering realistic constraints. In this paper, we present the novel method AutoSCOOP that solves the optimization problem based on evolutionary strategy for maximum coverage with a minimum number of road-side sensors. Here, we use accurate sensor coverage for optimization based on a detailed 3D environment model and ray-cast sensor models in a game engine. This allows automatic placement and evaluation of sensor positions while taking into account relevant optimization degrees of freedom and constraints.

We have shown in experimental simulations that the method applies to complex virtual routes of a real proving ground and determines robust and valid sensor coverage. Considering the constraints of the sensor itself, as well as the sensor area and target area for optimization, an optimal solution is automatically found, resulting in cost and time savings for a proving ground sensor infrastructure design.

Future work is planned on the sensor modeling for a more realistic coverage under poor visibility and adverse weather conditions. Besides, further improvements of the AutoSCOOP algorithm concerning a more effective closing of coverage gaps is another planned task.

References

[1] Hans-Hermann Braess et al. “Produktentstehungsprozess. In: Braess HH., Seiffert U. (eds) Vieweg Handbuch Kraftfahrzeugtechnik”. In: 2013. ISBN: 978-3-658-01691-3. DOI: 10.1007/978-3-658-01691-3_11.

[2] Alessia Knauss et al. “Proving Ground Support for Automation of Testing of Active Safety Systems and Automated Vehicles”. In: Sept. 2017.

[3] Jong Won Jeong et al. “The implementation of the autonomous guided vehicle driving system for durability test”. In: *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.00TH8493)*. 2000, pp. 101–106. DOI: 10.1109/ITSC.2000.881025.

[4] Gereon Hinz et al. “Designing a far-reaching view for highway traffic scenarios with 5G-based intelligent infrastructure”. In: *8. Tagung Fahrerassistenz*. Lehrstuhl für Fahrzeugtechnik mit TÜV SÜD Akademie. München, 2017.

[5] Annkathrin Krämmer et al. “Providentia – A Large Scale Sensing System for the Assistance of Autonomous Vehicles”. In: June 2019.

[6] Yanghui Mo et al. “A method of vehicle-infrastructure cooperative perception based vehicle state information fusion using improved kalman filter”. In: *Multimedia Tools and Applications* (Feb. 2021). DOI: 10.1007/s11042-020-10488-2.

[7] Jian Zhao et al. “Approximate techniques in solving optimal camera placement problems”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 1705–1712. DOI: 10.1109/ICCVW.2011.6130455.

[8] N. Conci and L. Lizzi. “Camera placement using particle swarm optimization in visual surveillance applications”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 3485–3488. DOI: 10.1109/ICIP.2009.5413833.

[9] Benoit Debaque, Rym Jedidi, and Donald Prevost. “Optimal video camera network deployment to support security monitoring”. In: *2009 12th International Conference on Information Fusion*. 2009, pp. 1730–1736.

[10] Altahir A. Altahir et al. “Modeling Multicamera Coverage for Placement Optimization”. In: *IEEE Sensors Letters* 1.6 (2017), pp. 1–4. DOI: 10.1109/LSENS.2017.2758371.

[11] Jose-Joel Gonzalez-Barbosa et al. “Optimal camera placement for total coverage”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 844–848. DOI: 10.1109/ROBOT.2009.5152761.

[12] Indu Sreedevi et al. “Camera Placement for Surveillance Applications”. In: Feb. 2011. ISBN: 978-953-307-436-8. DOI: 10.5772/14806.

[13] Francisca Rosique et al. “A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research”. In: *Sensors* 19 (Feb. 2019), p. 648. DOI: 10.3390/s19030648.

[14] Xiaohui Wang, Hao Zhang, and Haoran Gu. “Solving Optimal Camera Placement Problems in IoT Using LH-RPSO”. In: *IEEE Access* 8 (2020), pp. 40881–40891. DOI: 10.1109/ACCESS.2019.2941069.

[15] Vahab Akbarzadeh et al. “Probabilistic Sensing Model for Sensor Placement Optimization Based on Line-of-Sight Coverage”. In: *IEEE Transactions on Instrumentation and Measurement* 62.2 (2013), pp. 293–303. DOI: 10.1109/TIM.2012.2214952.

[16] Roshan Vijay et al. “Optimal Placement of Roadside Infrastructure Sensors towards Safer Autonomous Vehicle Deployments”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 2589–2595. DOI: 10.1109/ITSC48978.2021.9564822.

[17] Jinwoo Kim et al. “Camera Placement Optimization for Vision-based Monitoring on Construction Sites”. In: July 2018. DOI: 10.22260/ISARC2018/0102.

[18] Florian Geißler and Ralf Graefe. “Optimized sensor placement for dependable roadside infrastructures”. In: Oct. 2019.

[19] M. S. Sumi Suresh, Athi Narayanan, and Vivek Menon. “Maximizing Camera Coverage in Multicamera Surveillance Networks”. In: *IEEE Sensors Journal* 20.17 (2020), pp. 10170–10178. DOI: 10.1109/JSEN.2020.2992076.

[20] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: (Nov. 2017).

[21] Epic Games. *Unreal Engine*. Version 4.27.2. Jan. 19, 2022. URL: https://www.unrealengine.com.

[22] N. Akkiraju et al. “Alpha shapes: definition and software”. In: *presented at the GCG International Computational Geometry Software Workshop*. 1995, pp. 63–66.

[23] Thomas Bäck, Christophe Foussette, and Peter Krause. “Contemporary Evolution Strategies”. In: 2013, pp. 7–45. ISBN: 978-3-642-40137-4. DOI: 10.1007/978-3-642-40137-4.