Optimized Coverage Planning for UV Surface Disinfection

João Marcos Correia Marques[1], Ramya Ramalingam[2], Zherong Pan[1] and Kris Hauser[1]

***Abstract*— UV radiation has been used as a disinfection strategy to deactivate a wide range of pathogens, but existing irradiation strategies do not ensure sufficient exposure of all environmental surfaces and/or require long disinfection times. We present a near-optimal coverage planner for mobile UV disinfection robots. The formulation optimizes the irradiation time efficiency, while ensuring that a sufficient dosage of radiation is received by each surface. The trajectory and dosage plan are optimized taking collision and light occlusion constraints into account. We propose a two-stage scheme to approximate the solution of the induced NP-hard optimization, and, for efficiency, perform key irradiance and occlusion calculations on a GPU. Empirical results show that our technique achieves more coverage for the same exposure time as strategies for existing UV robots, can be used to compare UV robot designs, and produces near-optimal plans.**

I. Introduction

The Covid-19 pandemic has encouraged worldwide innovation in methods for reducing the risk of disease transmission in hospitals, public transportation and other public spaces. One promising technology is ultraviolet (UV) disinfection of surfaces, which has strong antimicrobial properties particularly in the UVC (200 nm to 280 nm) spectrum. UVC has long been known to deactivate a wide range of pathogens, such as Coronaviruses [1, 2], bacteria and protozoans [3]. Existing UV delivery approaches include air and water disinfection systems used in filtration and waste processing plants [2], as well as surface disinfection systems in the form of wands [4], overhead lights, pushcarts, and mobile robots carrying high-power UVC lamps [5]. Hospital testing [6] has shown that a combination of standard manual cleaning followed by UVC surface irradiation has shown to be more effective in disinfecting environments than manual cleaning alone.

Dosing is an important factor in effective use of UVC, and is usually performed by following manufacturers' guidelines. Although some UV disinfection robots also feature sensors that measure reflected radiant energy as an approximation of surface dosage, existing methods fail to disinfect certain parts of the environment [7]. Two pitfalls are noted. The radiant fluence received by a surface is affected by the inverse square law, so fluence drops quickly as distance increases. Second, occlusions also affect the delivery of light into back-facing or shadowed regions. These effects are illustrated in Figure 1, which shows a simulation of the irradiation of a hospital infirmary under a static UV tower, demonstrating sub-standard disinfection of bedsides and occluded equipment.

We present a method for planning optimal trajectories of a mobile UV disinfection robot with dosing constraints. Our optimization can be configured to prioritize coverage of high-touch surfaces under a fixed time budget, or to guarantee the eventual full disinfection of all surfaces reachable by irradiation. The robot's movement must be collision-free while conforming to the dosing constraints. We solve the problem by building a probabilistic roadmap in the robot's configuration space, and then finding a tour of a subset of configurations that optimizes the dose. The coverage problem on the roadmap can be cast as an NP-hard Mixed-Integer Linear Programming (MILP), but we propose an approximate two-stage solver that uses a Linear Program (LP) to find dwell times followed by a Traveling Salesman Problem (TSP) to find the tour. Experiments show that our solver is orders of magnitude faster than MILP with a loss of less than 3% of optimality. Moreover, dosage planning requires determination of an *irradiance matrix* that considers visibility and exposure of every surface patch from each candidate UV light pose, and we propose an approach that efficiently calculates this large matrix using a Graphics Processing Unit (GPU).

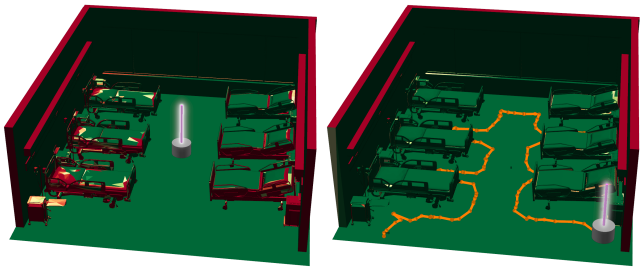


Fig. 1: Comparison of a standard stationary mobile robot (left) against an optimized motion (right). Robot carries a tower light. Surfaces are color coded by UV fluence received, with red indicating $0\,\mathrm{J/m^2}$ and green indicating $280\,\mathrm{J/m^2}$ or higher. A stationary light is unable to disinfect much of the environment after 30 minutes, while a mobile robot following our optimally computed trajectory (in orange) achieves almost complete coverage. (Best seen in color)

II. Related Work

Motion planning for UV disinfection bears a resemblance to two well-studied problems: coverage and inspection planning. The goal of coverage planning [8, 9, 10, 11] is for every point in the freespace to be covered by the robot, while the goal of inspection planning [12, 13, 14, 15, 16] is for every point on an object surface to be visible from some point on the robot trajectory. The disinfection planning problem introduced in this paper adds an additional layer of complexity to inspection planning, where every point in an object surface must receive a certain amount of irradiance exposure. This scenario induces a joint problem of robot trajectory planning and disinfection time assignment. Compared with standard coverage and inspection planning, UV disinfection is applied routinely in healthcare facilities, public spaces, and food

industries, and can take tens of minutes to ensure enough dosage. Therefore, achieving (near) optimality in reducing the disinfection time for a known environment, which is the focus of this paper, is more important than adapting to unknown environments or online re-planning as done in [13, 11].

Besides robotics, UV disinfection planning can be understood as an effort to model and control light transport. In this aspect, there is overlap with similar efforts in the field of radiation dosage planning [17, 18, 19, 20], rendering of Lambertian surfaces using boundary element method [21, 22, 23, 24] (otherwise known as radiosity), and optimization of light placements [25, 26]. The radiation dosage planning problem has the same goal as our problem, ensuring the delivery of sufficient amount of dosage to target volumes. An additional goal is to reduce the dosage as much as possible for the organs at risk. However, since geometric information of human organs is difficult to acquire, these methods are mostly heuristic and sub-optimal. Radiosity is used to only model light transportation, reflection, and absorption. Of particular interest is GPU-accelerated radiosity [23] where the occlusion map is computed using GPU rasterization. A similar technique is used in this work, while indirect light reflections are ignored by our method as their contributions are assumed neglectable. Other works on lighting optimization for urban design or scientific data visualization [25, 26] also considers moving light sources, but these lights are fixed after the design phase.

III. UV Disinfection Trajectory Planning

Here we formalize the path planning problem for targeted UV disinfection first as a continuous, infinite-dimensional trajectory optimization problem, and then as a discrete approximation.

A. Continuous Formulation

Let $\mathbb{E} \subset \mathbb{R}^3$ be the boundary of the environment, which is the surface to be disinfected. The disinfection is performed using a mobile robot equipped with a UV light, where $\mathbb{C}$ is the robot's configuration space and $\mathbb{C}_{free}$ is the freespace. When the robot assumes any collision-free configuration $x \in \mathbf{C}_{free}$, each infinitesimal surface patch $ds \in \mathbb{E}$ will receive a certain amount of radiative fluence per second. We model the radiative fluence distribution using a so-called Poynting vector function $\mathbb{I}(x, ds)$, such that the infinitesimal surface patch ds receives the following irradiance:

$$I_{ds}(x) = \langle \mathbb{I}(x, ds), n(s) \rangle, \tag{1}$$

where ds is the infinitesimal surface patch with outward normal $n(s)$ and $\langle \bullet, \bullet \rangle$ is the inner product. Note that $\mathbb{I}(x, ds)$ already encodes the effects of light mirror reflections and occlusions by the environment. For instance, in the case where there are full occlusions before reaching ds, this vector is zero. We denote $\tau(t) : \mathbb{R} \mapsto \mathbb{C}_{free}$ as the trajectory in the robot configuration space parameterized in time $t \in [0, T_{final}]$. The radiant fluence (also known as radiant exposure) of an infinitesimal surface patch ds from a trajectory τ, denoted by μ_{ds}, is described by:

$$\mu_{ds}(\tau) = \int_0^{T_{final}} I_{ds}(\tau(t)) dt. \tag{2}$$

We define the minimum-time, continuous path planning problem for UV disinfection as:

$$\begin{aligned} &\underset{T_{final}, \tau}{\mathbf{argmin}}\ T_{final} \\ &\quad \mathbf{s.t.}\ \ \mu_{ds}(\tau) \geq \mu_{min}(ds) \quad \forall ds \\ &\qquad \forall t \in [0, T_{final}] \begin{cases} \tau(t) \in \mathbb{C}_{free} \\ \dot{\tau}(t) = f(\tau, \dot{\tau}, u) \\ \|u(t)\| \leq u_{max} \end{cases}, \end{aligned} \tag{3}$$

where $f(\tau, \dot{\tau}, u)$ encodes the robot dynamics , $u(t)$ is the control signal, u_{max} is the control limits and $\mu_{min}(ds)$ is the minimum disinfection fluence (dose) prescribed to the surface. The prescribed dose can be surface-dependent (e.g., to deliver more radiation to high-touch surfaces), but we set a constant μ_{min} for notational simplicity. Eq. 3 is intractable due to the infinite number of constraints and the integral in Equation 2.

B. Discrete Formulation

Next, we formulate a discrete counterpart of (3). The surface $\mathbb{E}$ is discretized using a simplicial complex with N triangles, $\{s_i | i = 1, \cdots, N\}$. The robot can only take a discrete set of K configurations $\{x_1, \cdots, x_K\} \subset \mathbb{C}_{free}$. Each configuration x_k is called a vantage configuration. To simplify total irradiance calculations, we assume that the light source stops at each configuration x_k in its trajectory for some dwelling time, denoted as $t_k \geq 0$, and emits no radiation during the transition between vantage configurations. Let $\mathbf{t}$ be the vector of K dwell times. We then discretize (1) and (2) as:

$$I_i(x_k) = \int_{s_i} \langle \mathbb{I}(x_k, ds), n(s) \rangle ds, \tag{4}$$

$$\mu_i(\mathbf{t}) = \sum_{k=1}^{K} I_i(x_k) t_k. \tag{5}$$

Suppose there exists a network of paths between configurations that satisfies kinematics and dynamics constraints. Let $d_{kl} \geq 0$ be the distance along the network between any x_k and x_l, with $d_{kl} = \infty$ if no path connects them. We then formulate the discrete version of (3) as a path subset selection problem. We introduce binary variables $z_{kl} \in \{0, 1\}$, each indicating whether the path d_{kl} is used in the final path, and a vector $\mathbf{z}$ collecting each indicator. Then the discrete version of (3) is defined as:

$$\begin{aligned} &\underset{\mathbf{t}, \mathbf{z}}{\mathbf{argmin}}\ \sum_{k=1}^{K} t_k + \frac{1}{v_{max}} \sum_{k=1}^{K} \sum_{l=1}^{K} d_{kl} z_{kl} \\ &\quad \mathbf{s.t.}\ \mu_i(\mathbf{t}) \geq \mu_{min} \quad \forall i = 1, \cdots, N \\ &\qquad \mathbf{z} \text{ connected} \\ &\qquad t_k > 0 \text{ iff } z_{kl} = 1 \text{ or } z_{lk} = 1 \text{ for some } l. \end{aligned} \tag{6}$$

The last two conditions are consistency constraints, stating that the selected paths form a simply connected path, and the second ensures that the robot can only dwell on vantage configurations that are part of the selected path.

IV. Proposed Solution

In this section, we propose a novel approximate algorithm to search for near-optimal coverage plans. The main steps of our approach are listed below:

1) Select *vantage configurations* $\{x_1, \cdots, x_K\}$ (Sec. IV-A).
2) Compute network $\mathcal{R}$ of paths between configurations using a PRM-style approach. Retain subset of reachable configurations. (Sec. IV-B)
3) Compute irradiance matrix $I_i(x_k)$ (Sec. III-B)
4) Solve a LP for optimal dwell times $\mathbf{t}$ (Sec. IV-D)
5) Solve a TSP for a tour of all configurations x_k for which dwell time is nonzero, that is $t_k > 0$ (Sec. IV-D)
6) Execute the tour, stopping for time t_k at each visited configuration x_k

We remark that in our extended report [27], Eq. (6) can be formulated as a Mixed Integer Linear Program (MILP). As vantage configurations grow increasingly dense and paths in the network $\mathcal{R}$ approach optimal paths, the MILP solution will approach the optimal solution to the original continuous problem (3). However, MILPs are NP-hard, and our system sacrifices optimality with a two-stage LP+TSP approach.

The LP first finds an dosage plan, in the form of *dwell times* to be spent at each vantage configuration, that is optimal assuming that the robot can instantly "teleport" between configurations. Second, the TSP finds the minimum-time traversal of the configurations with non-zero dwell times. Assuming that the robot is sufficiently fast that irradiation is the limiting step, this strategy will produce near-optimal results.

Another issue to be addressed is that the integral in (4) does not have a closed form. We quickly compute an approximate irradiance vector from every vantage configuration and assemble them into an irradiance matrix using a GPU-based visibility check.

A. Vantage Configuration Selection

We first uniformly select a set of light positions in the task space, giving a superset $\{y_1, \cdots, y_{K'}\}$ of K' light positions. For each light position, we solve the inverse kinematics problem for each robot $IK(y_k) = x_k$ and insert x_k into the vantage configuration set if a collision-free IK solution can be found. During IK feasibility computation, the robot's geometry is dilated by 5 cm to discourage the use of "coiled" configurations, since these induce harder planning problems. The selection scheme of $\{y_1, \cdots, y_{K'}\}$ is robot-dependent. If the robot is able to move in 3-D, then they are drawn from an uniform grid in the bounding box of $\mathbb{E}$ in $\mathbb{R}^3$, but if the robot is constrained to 2D motion, like a mobile base, they are drawn from a gridding of the floorplan of $\mathbb{E}$ in $\mathbb{R}^2$.

B. Roadmap Computation

This component creates a PRM [28] to attempt to connect the vantage points $\{x_1, \cdots, x_K\}$ with feasible paths. The PRM is an undirected graph $R = (V, E)$ consisting of configurations $q \in \mathbb{C}_{free}$, called "milestones", and edges $(a, b) \in E$ between milestones a and b are straight line paths that are required to lie completely in the free space, that is, $\overline{ab} \in \mathbb{C}_{free}$.

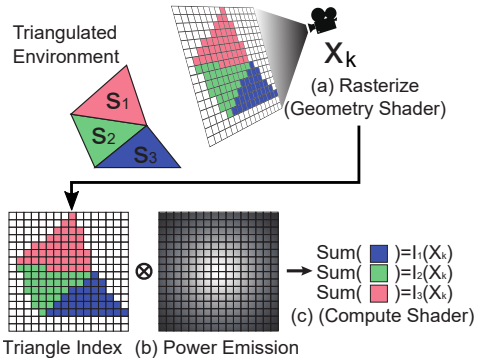


Fig. 2: Illustrating the GPU-based irradiance calculation. (a): The triangle index is rasterized into an environment map using geometry shader. (b): The power emission $e(i, j)$ is precomputed. (c): The operation $F[T[i, j]]$ += $e(i, j)$ is performed for all pixels on a compute shader. (Best seen in color)

We construct R with the following sampling scheme: 1) Add $\{x_1, \cdots, x_K\}$ as initial milestones of the PRM and try to connect pairs of nearby milestones if the edge between them is feasible. 2) Sample a some number of configurations uniformly at random, and configurations near milestones with a given radius. Connect nearby pairs of milestones with edges if feasible. 3) For pairs of neighboring vantage points that lie in different connected components of R, add more samples near this edge. This latter approach helps the planner focus its sampling to narrow passages in configuration space. Edge feasibility checking is done by checking the configuration space line segments and for collisions at regular intervals. The distance between two configurations is calculated by the length of the robot's end-effector trajectory induced by the interpolation.

After R is computed, vantage points that are not in the largest connected component are discarded. For the remaining points, the shortest paths in R between all pairs (x_k, x_l) are computed to form the distance matrix d_{kl}.

C. Discrete Radiative Fluence

We approximately calculate the radiative fluence matrix with entries $I_i(x_k)$. Note that a typical environment in 3D contains millions of triangles (N) and we will sample tens of thousands of potential vantage configurations (K). Therefore, the matrix size $I_i(x_k)$ is huge and its calculation can typically become the bottleneck. We provide a GPU-based implementation that can calculate each column $I_*(x_k)$ in milliseconds.

The irradiance is a measure of the rate of radiant exposure, and is given in the units of watts per square meter. We first describe the simple case where the robot is a point light source, i.e. $\mathbf{X} = \mathbb{R}^3$. We assume that reflected light is not a major source of illumination, so that the irradiance density received by the infinitesimal patch ds is given according to

the inverse square law:

$$\langle \mathbb{I}(x_k, ds), n(s) \rangle = \begin{cases} 0 & ds \text{ visible from } x_k \\ \frac{P\langle s - x_k, n(s) \rangle}{4\pi \|s - x_k\|^3} & \text{otherwise,} \end{cases} \tag{7}$$

where P is the power (or radiant flux) of the light source and s is the location of the infinitesimal surface patch. A patch is considered visible only if $< y - x_k, n(s) >> 0$ and no other surface lies closer to x_k along the ray $y - x_k$.

If no other triangles are in the way from x_k to the entire triangle s_i, then the irradiance can be calculated according to [29], i.e. the integral of Equation 4 has closed form solution. However, when occlusion occurs, no closed form solution can be found for the per-triangle irradiance. Instead, our GPU-based implementation calculates the irradiance $I_i(x_k)$ by rasterization. This roughly follows the pipeline for radiosity calculations used in computer graphics [30, 23] disregarding Lambertian reflectance. Our implementation (Fig. 2) is comprised of the following steps:

- The scene is rasterized using a standard graphics pipeline, with the camera centered at x_k. Each triangle's index is rendered into the pixel buffer T bound to a cubemap texture (the *visibility cube*) using framebuffer object and a geometry shader [31]. In the meantime, a Z-buffer is used for visible surface determination. After rasterization, we store the value $T[i, j]$ for each pixel (i, j) on the image plane. $T[i, j]$ is the index of the closest triangle intersecting the ray from pixel (i, j) to x_k. A void pixel indicates that no triangle is occupying the pixel.
- For each pixel $T[i, j]$ containing a visible triangle, the amount of power $e(i, j)$ emitted over the solid angle subtended by the pixel is calculated using [29] and accumulated into a Shader Storage Buffer Object (SSBO) denoted as the triangle buffer F. In particular, the operation $F[T[i, j]]+ = e(i, j)$ is performed for each pixel (i, j) in a compute shader [32] using the parallel prefix-sum algorithm [33]. The accumulated value for each triangle is the radiant flux, which measures irradiance integrated over the non-occluded area of the triangle.
- The radiant flux $F[i]$ is divided by the area of each triangle to obtain the mean irradiance $I_i(x_k) = F[i]/|s_i|$.

Because this process will be performed repeatedly, the power emission $e(i, j)$ for each pixel is precomputed and stored in a separate texture of the same dimensions as the rendered buffers, denoted as E, so that it can be retrieved with a single memory lookup. A note-worthy caveat of our method is the use of mean irradiance $I_i(x_k) = F[i]/|s_i|$ to replace the true uneven irradiance distribution within a single triangle, which can be remedied by having more finely discretized meshes.

Non-Point Light Sources: Our procedure to compute $I_i(x_k)$ can be naturally extended to non-trivial light source shapes, such as an omnidirectional cylindrical light source. In those instances, the surface of light sources can be approximated by a set of evenly distributed point sources, where each point source emits an equal fraction of the light's total radiant power. The total radiant flux is accumulated for each point before dividing by the area of each triangle to obtain the irradiance. More advanced shader programs such as [34] can also be used to approximate the continuous integration of light contributions along the light source's surface area on GPU. For light sources with uneven irradiance distribution, such as shielded or mirrored lights, we can replace the power emission texture E with a precomputed custom distribution.

If the light source is not standalone but mounted on a robot, then the position of the light source p is determined by its forward kinematics, which is denoted as $p(x_k)$ and plugged into Equation 7 in the place of x_k, arriving at $\mathbb{I}(p(x_k), n(s))$.

D. Approximate Two-Stage Optimization

At this point, all related variables of Equation 6 have been calculated. We proceed by relaxing all $z_{kl} = 1$ and derive our first linear program in the following form:

$$\begin{aligned} \underset{t_k}{\textbf{argmin}} \ & \sum_{k=1}^{K} t_k \\ \textbf{s.t.} \ & \mu_i \geq \mu_{min} \quad \forall i = 1, \cdots, N, \end{aligned} \tag{8}$$

A potential issue with Equation 8 is that it does not account for partially infeasible problems, which frequently occur in practice because some triangles s_k are totally invisible from all vantage configurations. In these cases, Equation 8 will report infeasibility and return unusable solutions. Instead, we propose the following relaxed LP that always returns feasible solutions:

$$\begin{aligned} \underset{t_k, \sigma_k \geq 0}{\textbf{argmin}} \ & \sum_{k=1}^{K} t_k + \sum_{i=1}^{N} p_i \sigma_i \\ \textbf{s.t.} \ & \mu_i + \sigma_i \geq \mu_{min} \quad \forall i = 1, \cdots, N \\ & \sum_{k=1}^{K} t_k \leq T_{max}, \end{aligned} \tag{9}$$

where p_k denotes the infeasibility penalty of a triangle s_k and σ_i is a slack variable allowing all constraints to be satisfied in the worst case. We further constrain the time budget for disinfection to T_{max}. With large penalties $p_k > \|I_*(x_*)\|_F$ and sufficiently large T_{max}, LP solver tends to set all $\sigma_i = 0$ and Equation 9 is identical to Equation 8. When some surfaces are totally invisible or disinfection cannot be accomplished within the time budget, the LP has to set $\sigma_i > 0$ for some i and take penalty $p_i \sigma_i$. For prioritized surface patches s_i, a larger p_i should be used so the LP tends to avoid positive σ_i.

To solve Equation 9 we leverage the large-scale interior-point algorithm implemented in [35]. We then solve the TSP problem to find $\{z_{kl} | t_k > 0 \wedge t_l > 0\}$. While this TSP is NP-hard, it is solved over a much smaller set of candidate paths. In addition, since it fits the traditional TSP formulation, we are able to leverage polynomial-time approximate TSP solvers, such as [36], which have near-optimal performance for relatively small euclidean TSP instances as the ones we encounter. Once the tour is found, the final disinfection trajectory is obtained by linearly interpolating in configuration space along the edges of the roadmap.

V. Experiments

Our experiments aim to answer the following questions:

1) How much better is the coverage of an optimally planned disinfection trajectory vs a single-point strategy?
2) How large is the optimality penalty incurred by solving the problem sequentially vs using an optimal MILP formulation?
3) How do different robot designs compare in terms of maximum disinfection coverage and efficiency?

We use a simplified 2.5D experiment to test questions 1 and 2, and a realistic environment in 3D for question 3.

In each experiment, all surfaces require a minimum disinfection fluence μ_{min} = 280 J/m^2, which is a conservative estimate of the necessary fluence to induce a 3 log10 reduction in infectivity of SARSCov2 [2]. In addition, the light is assumed to have a constant radiant flux, P, of 80 W and that the max speed of all robots is 0.5 m/s.

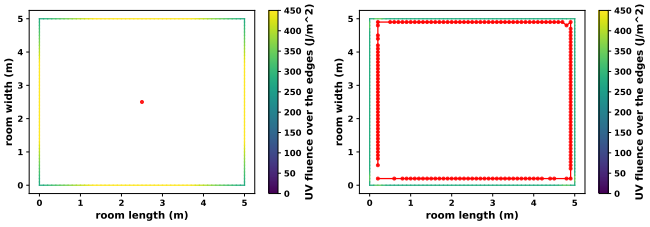
A. Comparison with static illumination

First we evaluate disinfecting the walls of a 5m×5m empty room as a 2.5D problem. Walls are 2 m meters tall and a spherical point light source is used. We consider a discretized version of the room where each wall is subdivided into fixed-length subsegments, and irradiance from a point can be calculated analytically for rectangles [37]. We consider a static illumination strategy that places the disinfection light to have maximum coverage over the obstacle space, allowing it to irradiate the surfaces for as long as necessary to fully disinfect its visible surfaces. In our method, we treat the robot as a cylindrical base of radius 10 cm, and constrain the movement of the light to a plane at height 1 m. Vantage points are sampled along a 0.1 m grid. The static method takes 143.7 minutes to reach full room disinfection, while ours does so in 95.6 minutes, including movement time between vantage points - the contrast between solutions is illustrated in Figure 3

Next, we randomly generate 25 2.5D rooms in a 4 m×4 m area and with 2 m tall polygonal obstacles. Each world contains a random number of obstacles between 7 and 19, with each obstacle randomly generated by scaling, shearing and displacing regular polygons. Visibilities of each segment from a given vantage point are determined by creating a visibility graph amongst vantage points and segment midpoints [38]. Figure 4 shows the output for one example. Note that for our method, all segments are covered, and few segments are overexposed. Figure 5 (a) shows results averaged over all rooms, indicating that our proposed method consistently disinfects 100% of the environment, whereas the optimal static illumination only disinfects 35%. Moreover, to disinfect the visible segments, static illumination requires approximately 2 orders of magnitude more time.

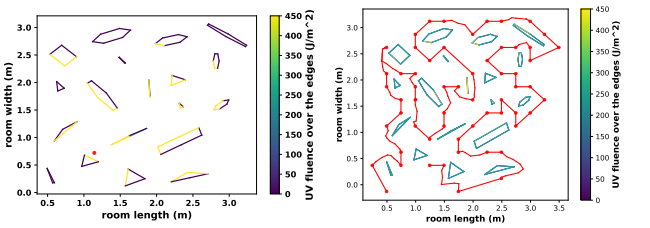
B. Comparison against MILP formulation

Next, we compare the two-stage scheme and the globally optimal MILP formulation described in our extended report [27]. We limited the point robot to travel along a 10x10 uniform grid of waypoints (note the 0.4 m spacing is coarser than the 0.25 m experiments above, since MILP performance



(a) Stationary illumination (b) Our method

Fig. 3: Empty room disinfected by the best stationary point (red dot) and by our method. Each surface is colored by its received fluence, and the optimized trajectory is drawn in red. (Best seen in color)



(a) Stationary illumination (b) Our method

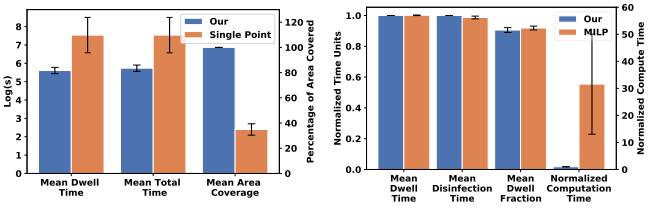
Fig. 4: Same room disinfected by the best stationary point (red dot) and by our method. Each surface is colored by its received fluence and the trajectory is drawn in red. (Best seen in color)

degrades sharply with the size of the problem). On 15 of the random 2.5D rooms, the MILP failed to converge within 24 h of computation, so we exclude these examples from the remaining discussion. These results are illustrated in Figure 5 (b). Columns 1, 2, and 4 are normalized to the LP+TSP results. Observe that the dwell times and disinfection times are nearly identical, with less than 3% difference between the two-stage approach and the optimal solution. Dwell times take approximately 90% of total disinfection time, and the paths computed by the two-stage approach are nearly identical to the optimal solution. Moreover, the two-stage approach is 30 times faster than the optimal MILP solution, even on a coarse grid and excluding cases where MILP took too long.

C. Comparing Robot Designs

Our 3D tests were performed in a hospital infirmary's CAD model[1] (Figure 8), simplified to 60k triangles using quadric edge collapse decimation. The method described

[1]https://grabcad.com/library/hospital-ward-2-2



(a) Ours vs Static (b) Ours vs MILP

Fig. 5: Evaluation results on 25 random 2.5D room. Error bars denote standard deviation.

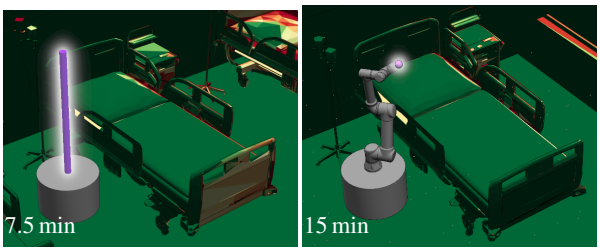


Fig. 6: Towerbot (left) and Armbot (right), mid-disinfection.

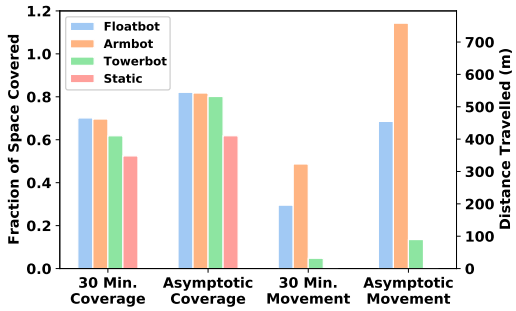


Fig. 7: Performance of robot designs disinfecting an infirmary.

in Section IV-C is configured to use 512x512 resolution framebuffers for computing irradiances.

We compare three models for the disinfection robot: "Floatbot", a freely-moving spherical light source, "Towerbot", a cylindrical mobile base that moves in the plane, is 55 cm in diameter and 37 cm tall, and has a 1.2 m tall cylindrical light source attached to its top (Figure 1); and "Armbot", a mobile base upon which a UR5e 6-DOF manipulator is mounted and holds a spherical point light source, both seen in (Figure 6), with their lamps highlitghted. Floatbot is an idealized model of maximum performance. Towerbot is a model for commercially available mobile disinfection robots (like the Akara Violet and UVD robot's Model B and C) [2], while Armbot represents a potential advancement that can access more hard-to-reach surfaces than a tower design. All solutions were computed within 50 minutes, with the irradiance matrix calculation taking over 80% of the time on all 3D experiments.

Our experiments designate an irradiation time limit of T_{max} = 30 minutes and 100 hours for evaluating asymptotic performance. For vantage point selection, we define a 3D grid with resolution 0.25 m (resulting in 8547 vantage candidates). We also compare with the strategy of placing Towerbot in the center of the room for the prescribed time budgets to mimic the static status-quo. During motion planning, 4k feasible samples are drawn to create the PRM (with additional samples added in increments of 20 if full milestone connectivity is not achieved).

Results are shown in Figure 7. We find that robots with more freedom to explore the free space, like Armbot and Floatbot, disinfect a larger area under a given time budget. Matching experimental results, [7], the status-quo of stationary placement of Towerbot fails to cover much of the surface, as illustrated in Figure 1. The asymptotic

[2]https://www.uvd-robots.com/robots — https://www.akara.ai/violet.html

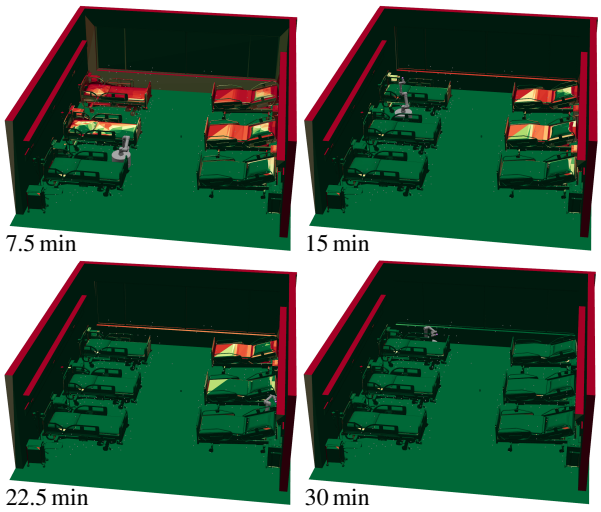


Fig. 8: Time lapse of Armbot's disinfection progress for an infirmary within a time budget of 30 minutes. (Best seen in color)

performance is nearly identical among all mobile solutions, whereas the disinfection efficiency comes with a tradeoff in total distance travelled, among which Towerbot has the smallest trajectory length and Armbot has the longest. This is presumably due to two factors. First, distances in higher dimensions tend to be higher (3D vs 2D) and, second, motion planning for Armbot involves many steps that are prone to sub-optimality, such as vantage configuration selection given a desired lamp position and high-dimensional multi-query path planning. Floatbot's trajectory length is a trivial lower bound to Armbot's trajecory length. More details about the trajectories can be found the attached suplemental video.

VI. Conclusion & Future Work

We presented a targeted approach to solve coverage planning problems for UV light disinfection. Our optimization minimizes the disinfection time while ensuring maximum coverage by imposing constraints of minimal irradiance exposure of surfaces. We show that globally optimal solutions can be found by solving a NP-hard MILP and propose a two-stage approximation scheme that can find near optimal solutions with less than 3% sacrifice of optimality while being orders of magnitude faster. We also confirm real-world experiments [7] that show limitations of stationary UV disinfection robots. Furthermore, our algorithm is general enough to analyze different robotic disinfection mechanisms. Code for the method is available at https://github.com/joaomcm/summer-2020.

In future work, we would like to analyze the MILP formulation and its interaction with the continuous path planning component. Second, we hope to test the proposed pipeline in a physical system to evaluate how positioning errors from SLAM algorithms and reconstruction errors affect disinfection performance. Third, our vantage configurations are sampled along a uniform task space grid, which may not be the most efficient choice. Finally, we would like to study how joint optimization of vantage configurations, task-space points, and paths could yield more efficient traversals.

References

[1] K. Bedell, A. H. Buchaklian, and S. Perlman, "Efficacy of an automated multiple emitter whole-room ultraviolet-C disinfection system against coronaviruses MHV and MERS-CoV," *Infection Control & Hospital Epidemiology*, vol. 37, no. 5, pp. 598–599, 2016.

[2] M. Heßling, K. Hönes, P. Vatter, and C. Lingenfelder, "Ultraviolet irradiation doses for coronavirus inactivation - review and analysis of coronavirus photoinactivation studies.," *GMS hygiene and infection control*, vol. 15, Doc08, 2020.

[3] W. A. M. Hijnen, E. F. Beerendonk, and G. J. Medema, "Inactivation credit of UV radiation for viruses, bacteria and protozoan (oo)cysts in water: A review," *Water Research*, vol. 40, no. 1, pp. 3–22, 2006.

[4] D. E. Lyon, *Uv Sterilizing Wand*, 2008.

[5] J.-s. Park, J.-S. Lee, J.-y. Ko, Y.-i. Cho, and J.-G. Song, *Robot cleaner having floor-disinfecting function*, 2007.

[6] D. Armellino, K. Goldstein, L. Thomas, T. J. Walsh, and V. Petraitis, "Comparative evaluation of operating room terminal cleaning by two methods: Focused multivector ultraviolet (FMUV) versus manual-chemical disinfection," *American Journal of Infection Control*, vol. 48, no. 2, pp. 147–152, 2020.

[7] M. Lindblad, E. Tano, C. Lindahl, and F. Huss, "Ultraviolet-C decontamination of a hospital room: Amount of UV light needed," *Burns*, 2019.

[8] P. Cheng, J. Keller, and V. Kumar, "Time-optimal UAV trajectory planning for 3D urban structure coverage," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2008, pp. 2750–2757.

[9] C Das, A Becker, and T Bretl, "Probably approximately correct coverage for robots with uncertainty," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1160–1166.

[10] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2013.

[11] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles," *Journal of Field Robotics*, vol. 32, no. 7, pp. 952–983, 2015.

[12] B. Englot and F. Hover, "Inspection planning for sensor coverage of 3D marine structures," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, pp. 4412–4417.

[13] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, Institute of Electrical and Electronics Engineers Inc., Jun. 2015, pp. 1071–1078.

[14] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, Institute of Electrical and Electronics Engineers Inc., Jun. 2015, pp. 6423–6430.

[15] A. Bircher, K. Alexis, U. Schwesinger, S. Omari, M. Burri, and R. Siegwart, "An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees," *Robotica*, vol. 35, no. 6, pp. 1327–1340, 2017.

[16] B. Bogaerts, S. Sels, S. Vanlanduit, and R. Penne, "A gradient-based inspection path optimization approach," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2646–2653, 2018.

[17] E. K. Lee, T. Fox, and I. Crocker, "Integer programming applied to intensity-modulated radiation therapy treatment planning," *Annals of Operations Research*, vol. 119, no. 1-4, pp. 165–181, 2003.

[18] G. K. Bahr, J. G. Kereiakes, H Horwitz, R Finney, J Galvin, and K Goode, "The method of linear programming applied to radiation treatment planning," *Radiology*, vol. 91, no. 4, pp. 686–693, 1968.

[19] G. A. Ezzell, "Genetic and geometric optimization of three-dimensional radiation therapy treatment planning," *Medical Physics*, vol. 23, no. 3, pp. 293–305, 1996.

[20] H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, and A. Kumar, "A new linear programming approach to radiation therapy treatment planning problems," *Operations Research*, vol. 54, no. 2, pp. 201–216, 2006.

[21] F. X. Sillion and C. Peuch, "Radiosity & global illumination," 1994.

[22] A. Keller, "Instant radiosity," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 49–56.

[23] G. Coombe, M. J. Harris, and A. Lastra, "Radiosity on graphics hardware," in *Proceedings of Graphics Interface 2004*, Citeseer, 2004, pp. 161–168.

[24] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila, "Incremental instant radiosity for real-time indirect illumination.," in *Rendering Techniques*, 2007, pp. 277–286.

[25] I. Stefan and H. Haas, "Analysis of optimal placement of led arrays for visible light communication," in *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*, IEEE, 2013, pp. 1–5.

[26] Y. Zhang and K.-L. Ma, "Lighting design for globally illuminated volume rendering," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 12, pp. 2946–2955, 2013.

[27] Z. P. João Marcos Correia Marques Ramya Ramalingam and K. Hauser, "A targeted approach to uv disinfection of surfaces," *https://uofi.box.com/s/gk7o1vyqkiggayquwkeedkhpk05j78oi*, 2020.

[28] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[29] J. C. Mosher, R. M. Leahy, and P. S. Lewis, "EEG and MEG: forward solutions for inverse methods," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 3, pp. 245–259, 1999.

[30] M. F. Cohen and D. P. Greenberg, "The hemi-cube: A radiosity solution for complex environments," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 31–40, 1985.

[31] S. Green, "The OpenGL framebuffer object extension," in *Game developers conference*, vol. 2005, 2005.

[32] M. Bailey, "OpenGL Compute Shaders," *Oregon State University*, 2016.

[33] M. Harris, "Parallel prefix sum (scan) with CUDA,"

[34] E. Heitz, J. Dupuy, S. Hill, and D. Neubelt, "Real-time polygonal-light shading with linearly transformed cosines," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–8, 2016.

[35] L. L. C. Gurobi Optimization, *Gurobi Optimizer Reference Manual*, 2018.

[36] K. Helsgaun, "Effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, Oct. 2000.

[37] A Van Oosterom and J Strackee, "The Solid Angle of a Plane Triangle," *IEEE Transactions on Biomedical Engineering*, vol. BME-30, no. 2, pp. 125–126, 1983.

[38] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.