Optimized Coverage Planning for UV Surface Disinfection

João Marcos Correia Marques\textsuperscript{1}, Ramya Ramalingam\textsuperscript{2}, Zherong Pan\textsuperscript{1} and Kris Hauser\textsuperscript{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 \cite{4,21}, bacteria and protozoans \cite{22}. Existing UV delivery approaches include air and water disinfection systems used in filtration and waste processing \cite{21}, surface disinfection systems in the form of wands \cite{29}, overhead lights, pushcarts, and mobile robots carrying high-power UVC lamps \cite{31}. Hospital testing \cite{1} has shown that a combination of standard manual cleaning followed by UVC surface irradiation is more effective than manual cleaning alone in disinfecting environments.

Dosing is an important factor in effective use of UVC. However, 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 \cite{27}. 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 shaded regions. These effects are illustrated in Figure 1 which shows a simulation of the irradiation of a hospital infirmary by a static UV tower, showing sub-standard disinfection of bedsides and occluded equipment.

*J. Marques and K. Hauser are partially supported by NSF Grant NRI-2025782. R. Ramalingam is supported by a CRA-WP DREU grant.
\textsuperscript{1}J. M. C. Marques, Z. Pan and K. Hauser are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA \{jmc12,zherong,kkhauser\}@illinois.edu.
\textsuperscript{2}R. Ramalingam is with the Dept. of Computer Science and the Dept. of Mathematics, Harvey Mudd College, Claremont, CA, USA rramalingam@g.hmc.edu.

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 reachable surfaces. 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).

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 \cite{8,12,15,32} is for every point in the freespace to be covered by the robot, while the goal of inspection planning \cite{5,6,7,13,20} 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 in known environments, which is the focus of this paper, is more important than adapting to unknown environments or online re-planning as done in Refs. [15, 20].

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 [2, 14, 26, 33], rendering of Lambertian surfaces using boundary element method [10, 24, 25, 34] (otherwise known as radiosity), and optimization of light placements [35, 37]. The radiation dosage planning problem has the same goal as our problem, ensuring the delivery of sufficient amount of dosage to target volumes with the additional goal of reducing collateral radiation damage. This field, however, neglects motion planning aspects of the problem and mostly resorts to heuristics to derive its dosage plan due to the difficulty in obtaining accurate 3D organ data online. Radiosity is used to only model light transport, reflection, and absorption. Of particular interest is GPU-accelerated radiosity [10] 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 [35, 37] also consider 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 UV disinfection as a continuous, infinite-dimensional trajectory optimization problem, and then as a discrete approximation.

A. Continuous Formulation

Let \( E \subset \mathbb{R}^3 \) 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 \( C \) is the robot’s configuration space and \( C_{\text{free}} \) is the freespace. When the robot assumes any collision-free configuration \( x \in C_{\text{free}} \), each infinitesimal surface patch \( ds \in E \) will receive a certain amount of radiant fluence per second. We model the radiative fluence distribution using a Poynting vector function \( I(x, ds) \), such that the infinitesimal surface patch \( ds \) receives the following irradiance:

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

where \( ds \) is the infinitesimal surface patch with outward normal \( n(s) \) and \( \langle \cdot, \cdot \rangle \) is the inner product. Note that \( I(x, ds) \) 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 \( \tau(t) : \mathbb{R} \rightarrow C_{\text{free}} \) as the trajectory in the robot configuration space parameterized in time \( t \in [0, T_{\text{final}}] \). The radiant fluence (also known as radiant exposure) of an infinitesimal surface patch \( ds \) from a trajectory \( \tau \), denoted by \( \mu_{ds} \), is described by:

\[
\mu_{ds}(\tau) = \int_0^{T_{\text{final}}} I_{ds}(\tau(t)) \, dt. \tag{2}
\]

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

\[
\arg\min_{T_{\text{final}}, \tau} \mu_{ds}(\tau) \quad \text{s.t.} \quad \mu_{ds}(\tau) \geq \mu_{\text{min}}(ds) \quad \forall ds \\
\quad \quad \quad \quad \forall t \in [0, T_{\text{final}}] \quad \| \tau(t) \| \leq u_{\text{max}}
\]

where \( \mu(\tau, \tau, u) \) encodes the robot dynamics , \( u(t) \) is the control signal, \( u_{\text{max}} \) is the control limits and \( \mu_{\text{min}}(ds) \) 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 \( \mu_{\text{min}} \) 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 \( E \) is discretized using a simplicial complex with \( N \) triangles, \( \{ s_i | i = 1, \ldots, N \} \). The robot can only take a discrete set of \( K \) configurations \( \{ x_1, \ldots, x_K \} \subset C_{\text{free}} \). 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 \( t \) be the vector of \( K \) dwell times. We then discretize (1) and (2) as:

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

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

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 \( z \) collecting each indicator. Then the discrete version of (3) is defined as:

\[
\arg\min_{t, z} \sum_{k=1}^{K} t_k + \frac{1}{v_{\text{max}}} \sum_{k=1}^{K} \sum_{l=1}^{K} d_{kl} z_{kl} \quad \text{s.t.} \quad \mu_i(t) \geq \mu_{\text{min}} \quad \forall i = 1, \ldots, N \tag{6}
\]

\[
z \text{ connected} \\
t_k > 0 \text{ iff } z_{kl} = 1 \text{ or } z_{lk} = 1 \text{ for some } l.
\]

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 ALGORITHM

Here we describe our 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 \( I_1(x_k) \) (Sec. III-B)
4) Solve a LP for optimal dwell times \( t \) (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 in our extended report [11], 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, the MILP is very high dimensional, so we propose a two-stage LP+TSP approach which sacrifices optimality for feasibility.

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 “teleport” between configurations. Then, 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 should produce near-optimal results.

Another issue to be addressed is that the integral in [4] does not have a closed form. We 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 the 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 \( E \) 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 \( E \) in \( \mathbb{R}^2 \).

B. Roadmap Computation

This component creates a PRM [23] to attempt to connect the vantage configurations \( \{x_1, \cdots, x_K\} \) with feasible paths. The PRM is an undirected graph \( \mathcal{R} = (V, E) \) consisting of configurations \( q \in C_{free} \), 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, \( ab \in C_{free} \).

We construct \( \mathcal{R} \) 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 some configurations uniformly at random, and some configurations near milestones within 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 \( \mathcal{R} \), add more samples near this edge. This approach helps the planner focus its sampling on narrow passages in \( C_{free} \). Edge feasibility checking is done by checking the configuration space interpolation 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 \( \mathcal{R} \) is computed, vantage points that are not in the largest connected component are discarded. For the remaining points, the shortest paths in \( \mathcal{R} \) 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 \( I_1(x_k) \). 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 \( I_1(x_k) \) is large and its calculation can be a bottleneck. We provide a GPU-based implementation that can calculate each column \( I_1(x_k) \) in milliseconds.

The irradiance is a measure of the rate of radiant exposure, and is given in the units of \( W/m^2 \). We first describe the simple case where the robot is a point light source, i.e. \( X = \mathbb{R}^3 \). We assume that reflections are negligible, so that the radiation received by the infinitesimal patch \( ds \) is given...
according to the inverse square law:
\[
(I(x_k, ds), n(s)) = \begin{cases} 
0 & \text{ds not visible from } x_k \\
\frac{Ps-s \cdot n(s)}{4\pi|s-x_k|^2} & \text{otherwise},
\end{cases}
\]
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 \[10\], i.e. the integral of Equation \[4\] has closed form solution. However, when occlusions occur, 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 \[9, 10\] disregarding Lambertian reflectance. Our implementation (Fig. 2) is comprised of the following steps:

1. 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 \[16\]. 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.

2. 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 \[10\] and all power terms \( e(i, j) \) belonging to \( T[i, j] \) are summed up and stored in the triangle buffer \( F \). This summation of \( e(i, j) \) is performed using the GPU’s hardware accelerated pixel-blending. In particular, we first set the triangle buffer \( F \) as the render target and store \( T[i, j] \) in the GPU buffer. Then we execute a shader program for each \( e(i, j) \), where we check \( T[i, j] \) for the index in \( F \) and use geometry shader \[15\] to render a single pixel into \( F \), with color equal to \( e(i, j) \) and pixel-blending turned on. The accumulated value for each triangle is the radiant flux, which measures irradiance integrated over the non-occluded area of the triangle.

3. The radiant flux \( F[i] \) is divided by the area of each triangle to obtain the mean irradiance \( I_i(x_k) = F[i]/|s_i| \).

Since this process is 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, like an omnidirectional cylindrical light source. The surface of light sources can be approximated by a set of evenly distributed point sources, each emitting 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 \[18\] 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 \( I_i(p(x_k), n(s)) \).

### D. Approximate Two-Stage Optimization

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

\[
\text{argmin}_{t_k} \sum_{k=1}^{K} t_k \quad \text{s.t. } \mu_i \geq \mu_{min} \quad \forall i = 1, \ldots, N,
\]

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:

\[
\text{argmin}_{t_k, \sigma_k \geq 0} \sum_{k=1}^{K} t_k + \sum_{i=1}^{N} p_i \sigma_i \quad \text{s.t. } \mu_i + \sigma_i \geq \mu_{min} \quad \forall i = 1, \ldots, N 
\]

where \( p_i \) denotes the infeasibility penalty of a triangle \( s_i \) and \( \sigma_i \) 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 \( p_i > \| I_i(x_s) \|_F \) 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 \( s_j \), a larger \( p_i \) should be used so the LP tends to avoid positive \( \sigma_i \).

To solve Equation \[9\] we leverage the large-scale interior-point algorithm implemented in \[17\]. We then solve the TSP problem to find \( \{z_{ik} | t_k > 0 \land t_j > 0 \} \). 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 \[19\], 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 compared to 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 3D environment to answer question 3.

In each experiment, all surfaces require a minimum disinfection fluence $\mu_{min} = 280 \text{ J/m}^2$, a conservative estimate of the necessary fluence to induce a $3 \log_{10}$ reduction in infectivity of SARS-CoV2 [21]. In addition, the light is assumed to have a constant radiant flux, $P$, of 80 W and the maximum speed of all robot end-effectors is 0.5 m/s.

A. Comparison with static illumination

First we evaluate disinfecting the walls of a 5m\times5m empty room as a 2.5D problem. Walls are 2m 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 [36]. 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. 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 0.1m 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 4m\times4m area and with 2m 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 between vantage points and segment midpoints [28]. 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 [11]. We limited the point robot to travel along a grid with 0.5m spacing (note the spacing is coarser than the 0.25m experiments above, since MILP performance degrades sharply with the number of vantage). These results are illustrated in Figure 5 (b). Columns 1, 2 show the percent difference between the two strategies in path length and total time. 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 6 times faster to compute than the optimal MILP solution, even on a coarse grid with only 64 possible vantage points.

C. Comparing Robot Designs

Our 3D tests were performed in a hospital infirmary’s CAD model\(^1\) (Figure 8), simplified to 35 thousand trian-
of the surface, as illustrated in Figure 1. The asymptotic 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 trajectory length. More details about the trajectories can be found in the attached supplemental 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 an 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 much faster. We also confirm real-world experiments \[27\] that show limitations of stationary UV disinfection robots. Furthermore, our algorithm is general enough to analyze different robotic disinfection designs. Code for the method is available at https://github.com/joaomcm/Optimized-UV-Disinfection

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


