A Review of Search- Based Path Planning Algorithms for Indoor Navigation of UV Sterilization Robots

Nupur Nimbekar
Robotics Engineering,
A. James Clark School of Engineering
University of Maryland,
College Park, USA
ndnupur@terpmail.umd.edu

***Abstract*— Given the current COVID-19 pandemic, sudden importance has been assigned to various kinds of sterilization robots. But the field of path planning is not being utilized to its fullest extent to increase the level of path planning for these robots. This paper focuses of basic search-based algorithms to study the implementation in indoor environments for UV Sterilization robots and to review the results. Ultraviolet sterilization uses short wavelength ultraviolet-C (UVC), light for reduction of microorganisms that may remain on the different surfaces after a standard process of cleaning. The main aim is to implement basics so that we can further integrate already existing planning techniques and see what is best for Navigation of UV Sterilization robots in future**

Keywords—UV Sterilization Robots, UV Lights, Dijkstra Algorithm, A* Algorithm, Path planning techniques

I. Introduction

Ultraviolet sterilization uses short wavelength ultraviolet-C (UVC), light for reduction of microorganisms that may remain on the different surfaces after a standard process of cleaning. Advancing technology has led to development of UV Sterilization Robots which can use this lights and clean highly contaminated rooms while reducing human contact with the infected surfaces.[2] Ultraviolet light has energy characteristics which make it effective in destroying microorganisms such as fungi. The wavelength for UV-C ranges from 200-280nm in the spectrum of light as shown below.

When exposed to UV light, the genetic material in cells is damaged as chemical bonds are severed within the DNA structure. Prolonged exposure to UV light inflicts more damage to the point where the DNA cannot be repaired, and cells die as a result.[3] Advancing technology has led to development of UV Sterilization Robots which can use this light and clean highly contaminated rooms while reducing human contact with the infected surfaces.[2]

There are many fields in which this technology can assist us such as, air disinfection, water disinfection, wastewater treatment, medicinal facilities, and laboratories. There are UV-Sterilization robots tested and used in the field of horticulture serving the cultivation process. This automation aims to reduce the labor cost of this treatment and to improve health and safety of the production operators.[3]

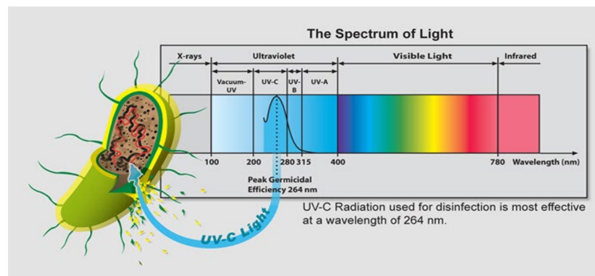

Fig 1 Spectrum of light [2]

As there are multiple applications of UV Light for sterilization, there are multiple existing technological solutions available in the US Market. However, there are few limitations to the existing solutions, and that is their lack of flexibility, mobility, and ability to autonomously position themselves in the environment they are subjected to.

Therefore, there is a huge unexplored potential for handling disinfection via a mobile robot solution with a UV light system that is designed specifically for disinfection of hospitals.[4]

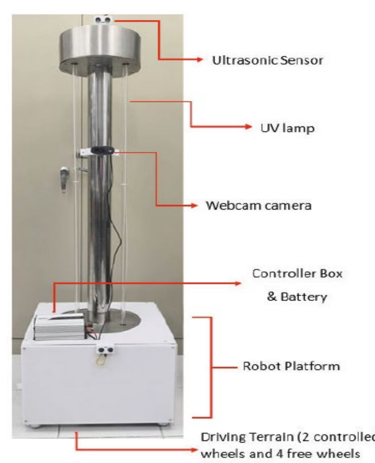

Fig 2 UV Sterilization Robot [2]

To increase the efficiency of these robots, we can study the existing path planning algorithms used in the UV Sterilization Robots and review the existing comparative study of well-known path planning algorithms to understand the advantages and disadvantages of the possible path planning techniques.

Although, Real-Time implementation would be the most efficient if a hybrid approach of navigation is implemented where the pros of both deterministic approach and Reactive approach are utilized [1], we will be studying more about the search-based algorithms such as Dijkstra and A-star (A*) which belongs to the deterministic approach of navigation for this paper.

II. Observation and Assumptions

A. Literature Survey

Literature Survey will present us with basic idea of path planning algorithms used and the result of comparative studies conducted so far to give us an overview of which algorithms are we going to study for this review paper. First, let us look at general ideas implemented for UV Sterilization robots as the main path planning technique.

Title	Path Planning Algorithm	Summary of the Paper
HCTNav: A Path Planning Algorithm for Low-Cost Autonomous Robot Navigation in Indoor Environments.[1]	• HCTNav • Comparison between HCTNav and Dijkstra	This paper presents the *HCTNav* path-planning algorithm. This algorithm was designed to be run in low-cost robots for indoor navigation. The results of the comparison between *HCTNav* and the Dijkstra's algorithms show that *HCTNav's* memory peak is nine times lower than Dijkstra's in maps with more than 150,000 cells
An Ultra-violet sterilization robot for disinfection.[2]	• Navigation based on Raspberry Pi UI	This paper presents a radical changes and improvement in the UV Sterilization process with 360-degree vision and has Wi-Fi/UI based navigation technique implemented to avoid obstacles. This was implemented because the robot was advance enough to have vision sensors and high sensing capabilities
UV-Robot Supervision System Design and Development.[3]	• Pre-set path	This paper is mainly based on developing an efficient Human-Robot interface to an existing UV-Robot for usage in the field of Horticulture and innovative farming. The robot used in this paper based on a pre-set path.

Table 1. Existing path planning techniques used in UV Sterilization Robots that are presented on various platforms

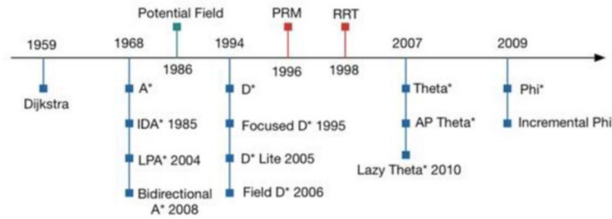

Fig 3. Mobile Robot path planning algorithm development [8]

As we can see that there is a vast area yet undiscovered or not implemented in the field of UV Sterilization robot and that is the path planning techniques. Many Robots being used in the hospitals are non-autonomous but have certain degrees or freedom associated with it. They are moved manually from one place to another demeaning the main purpose of sterilization robots which is to maintain certain amount of distancing between the infected regions and humans.

If a proper study of Indoor navigation techniques using various existing path planning algorithms is conducted, it can contribute with absolute knowledge regarding the possible ways of making these existing robots autonomous and more efficient. Let us look at existing studies based on various path planning algorithms for autonomous robots.

Title	Path Planning Algorithm	Summary of the Paper
A Review of Path Planning and Control for Autonomous Robots.[5]	• A-star • Probabilistic Roadmap • Genetic Algorithms	This paper presents a review of some well-known motion planning techniques, which are: A*, PRM and GA, they are applied to a mobile robot operating into a given environment which contains random shaped obstacles located arbitrarily. Simulation results show that PRM has the best performance in time computing, that it has path lengths close to A* algorithm results (constraints implied).
A Survey of Autonomous Mobile Robot Path Planning Approaches.[6]	• Heuristic Approach • Classical Approach	In this paper, a brief introduction and review of some of the path planning approaches (both classical and heuristic) is provided. The advantages and disadvantages of the methods are discussed.
A New Approach of Path Planning for Mobile Robots.[7]	• Visibility graph • Dijkstra • A*	This paper presents the new approach of path planning technique in which the virtual size of the obstacle present in the environment is assumed to be increased approximately *(2n+1)* times of the size of the cell. The experimental analysis of the proposed

		method shows the improvement in the path planning which reduces the chances of collisions. The simulations of this paper show the improved results for indoor environmental path planning
A Review of Representation, Model, Algorithm and Constraints for Mobile Robot Path Planning.[8]	• Graph Based • Sample Based • Searching based	This paper studies the path planning techniques in many details and provide states that Graph-based map representation and probabilistic method (random sample) is a very promising technology to solve the path planning problem.

Table. 2 Existing studies on path planning techniques used and presented on various platforms

Path planning algorithms aim to find the optimal path between two points, so the robot consumes as little energy as possible.[1] The literature survey shows us how path planning can be categorized on various basis, but we shall consider various searching algorithms and their study to understand the advantages and disadvantages of various searching algorithms.

B. *Objectives and Assumptions*

Main aim of the project is to achieve a review paper, studying various Search-Based Path Planning algorithms for indoor navigation of UV Sterilization Robots. To conduct that study, we will be visualizing the search algorithm to check its efficiency and how it can be implemented in an indoor environment with multiple points affected by bacterial infestation. The two algorithms are to be compared based on their performance and efficiency while computing the path to goal.

In order to do so, certain assumptions have been made. First, we assume that the entire UV Sterilization Robot System consists of a Mobile Robot, sensor system and a well-designed GUI. With communication established between all three, robot should always be able to access all sensory data from the sensor system.

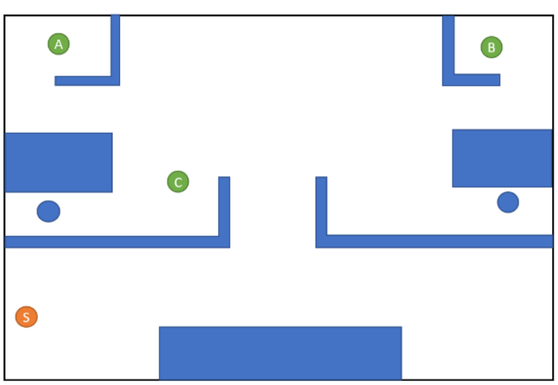

Fig 4. Map with Contamination points

The sensor system should have calibrated cameras able to capture the dimensions of environment and account for the changes in the position of furniture (Beds, medical monitoring devices etc.) and Bio-Sensors capable of sensing contamination point specification (Position in room, percentage, kind of bacterial infestation etc.).

We will be considering three contamination points in the map. The map here resembles a basic hospital ward.

III. METHODOLOGY

Path planning is considered as one of the most important tasks for any kind of navigation-based robot. The main requirement of the accurate and optimal path planning is the accessibility of environmental and odometric information. This depends upon how accurate the environmental and odometric information is attained by the robot. In such situations the sensors play fundamental role.[7] While enhancing the sensing capability, we are more focused on deterministic approach of navigation.

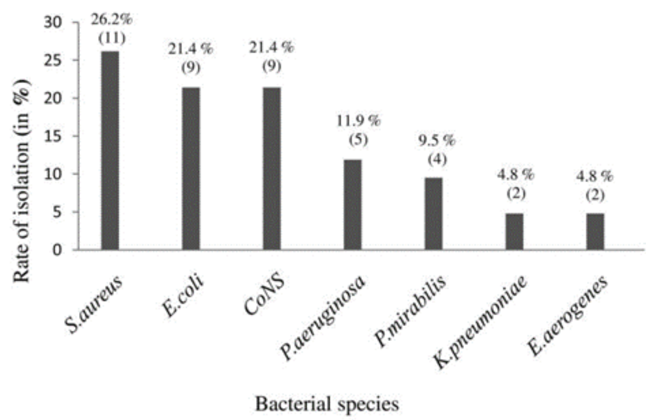

Fig. 5. Bacteria that cause infection in operation room.[2]

The attempt of the proposed algorithm will be to add priority to goal based on the contamination percentage. The study of various research paper was done, but modification based on basic search-based algorithms to provide with base-level path planning for UV Sterilization robot is not available. Hence, we will be conducting our own study to reflect how the modifications might alter the basic deterministic navigation techniques.

A. *Exposure Time*

The disinfection capability of robot will vary based on how far it is from the contamination point.[4] Also, a halt will be added when the robot actually reaches a contamination point such that robot waits for the time period which is equal to the exposure time required for the inactivation of infestation in that particular zone.[2] Based on the kind of infestation (bacteria), brightness of the light can also be modified to increase or reduce the exposure time.

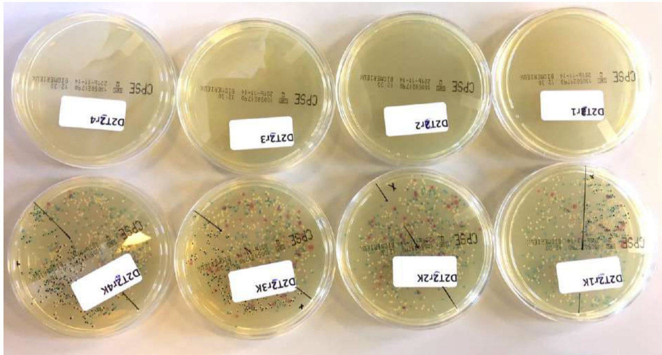

Fig. 6. Four samples showing bacteria with no UV-light exposure and after UV-light exposure from a 1.5 m. distance for 75 sec. Yellow dots = S. aureus. Red dots = E. coli. Green dots = E. faecalis.[4]

From figure 6, we can see the efficiency of UV Light at 1.5 m for 75 sec. Hence, we will be using these specifications for out experiment. Although, when we implement the algorithm practically the following formulae can be used to calculate the exposure time.[3]

$$Brightness = \frac{Luminosity\ (w)}{4\pi \times Distance^2 (cm^2)} \tag{1}$$

$$Time = \frac{UV\ Dose\ (\mu W \frac{sec}{cm^2})}{Brightness\ (w/cm^2)} \tag{2}$$

B. Dijkstra Algorithm

Before going into details of the pseudo-code of the algorithm it is important to know how the algorithm works. Dijkstra's algorithm works by solving the sub problem k, which compute the shortest path from source to vertices among the k closest vertices to the source. For the Dijkstra's algorithm to work it should be directed-weighted graph and the edges should be non -negative. If the edges are negative, then the actual shortest path cannot be obtained.

The algorithm works by keeping the shortest distance of vertex v from the source in an array, Dist. The shortest distance of the source to itself is zero. Distance for all other vertices is set to infinity to indicate that those vertices are not yet processed. After the algorithm finishes the processing of the vertices distance will have the shortest distance of vertex from source to every other vertex. Two sets are maintained which helps in the processing of the algorithm, in first set all the vertices are maintained that have been processed i.e. for which we have already computed shortest path. And in second set all other vertices are maintained that must be processed. This algorithm makes no attempt of direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. This algorithm therefore expands outward from the starting point, interactively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path. However, it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.

A simple Dijkstra Pseudocode is defined as the following:

```
Q.Insert(x_I) and mark x_I as visited
while Q not empty do
  x ← Q.GetFirst()
  if x ∈ X_G
    return SUCCESS
  forall u ∈ U(x)
    x' ← f(x, u)
    if x' not visited
      Mark x' as visited
      Q.Insert(x')
      Parent( x' ) ⟵ x
      CostToCome(x') =CostToCome(x) +l(x,u)
      Cost(x')= CostToCome(x')+CostToGo(x')
    else
      if  Cost(x') >  CostToCome(x)+l(x,u) +CostToGo(x')
            CostToCome(x') =CostToCome(x) +l(x,u)
            Cost(x')= CostToCome(x')+CostToGo(x')
            Parent(x') ⟵ x
~~Resolve duplicate x'~~
return FAILURE
```

(Source: LaValle, S. M. Planning Algorithms, 2006)

C. A* Algorithm

A robot typically wants to reach the goal node as quickly as possible from the starting node. What A* does is that at each step, it picks the node with the lowest F where F(n) = g(n) +h(n) Here, n is the previous node, g(n) is the cost of the path from the start node to n (or cost-to-come), and h(n) is the heuristic that estimates the cost of the cheapest path from n to the goal node (or cost-to-go). A* is a heuristic search, which means it searches with information about the goal. A* search finds the optimal solution to problems as long as the heuristic is admissible, which means it never overestimates the cost of the path to the from any given node. The time complexity for A* is O(N), where N is the number of edges in the graph [2]. A simple A* Pseudocode is defined as the following:

```
Q.Insert(x_I) and mark x_I as visited
while Q not empty do
  x ← Q.GetFirst()
  if x ∈ X_G
    return SUCCESS
  forall u ∈ U(x)
    x' ← f(x, u)
    if x' not visited
      Mark x' as visited
      Q.Insert(x')
      Parent( x' ) ⟵ x
      CostToCome(x') =CostToCome(x) +l(x,u)
      Cost(x')= CostToCome(x')+CostToGo(x')
    else
      if  Cost(x') >  CostToCome(x)+l(x,u) +CostToGo(x')
            CostToCome(x') =CostToCome(x) +l(x,u)
            Cost(x')= CostToCome(x')+CostToGo(x')
            Parent(x') ⟵ x
~~Resolve duplicate x'~~
return FAILURE
```

(Source: LaValle, S. M. Planning Algorithms, 2006)

D. Path Planning Algorithm for UV Sterilization Robot

The algorithm for UV Sterilization Robot was solely designed to suit the environment given the assumptions are fulfilled. Having placed the sensor system throughout the room, we will be able to gather all the necessary data.

The algorithm will be as following:

- **Start**
- Goal points (Contamination points) will be computed from sensor. *(In our prototype-code we will set this information as default values)*
- **Step 0:** Centroid will be calculated considering every goal point as a vertex of polygon
- Priority Queue will be made based on Contamination Points gathered.
- **Step I:** Path will be planned to reach that centroid (Dijkstra/A* algorithms)
- Start Navigation to Centroid. (*Actual Implementation*)
- **Step II:** Check the contamination percent priority queue to select goal point with the highest percent and set that as our goal point while removing it from the queue and centroid as our start point
- **Step III:** Path will be planned to reach 1st Goal Point (Dijkstra/A* algorithms)
- Start Navigation to 1st Goal Point. (*Actual Implementation*)
- Repeat Step II
- **Step IV:** Path will be planned to reach 2nd Goal Point (Dijkstra/A* algorithms)
- Start Navigation to 2nd Goal Point. (*Actual Implementation*)
- Repeat Step II
- **Step V:** Path will be planned to reach nth Goal Point (Dijkstra/A* algorithms)
- Start Navigation to 3rd Goal Point. (*Actual Implementation*)
- **End**

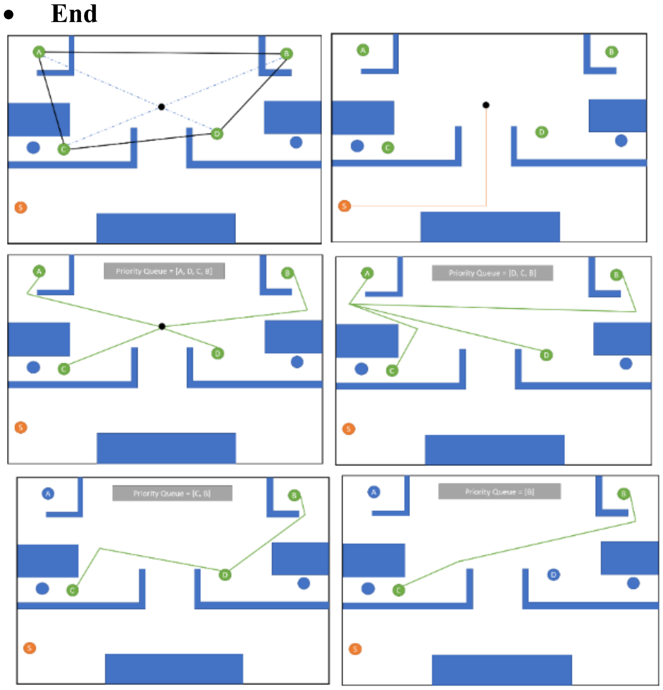

Fig 7. Visualization of Algorithm

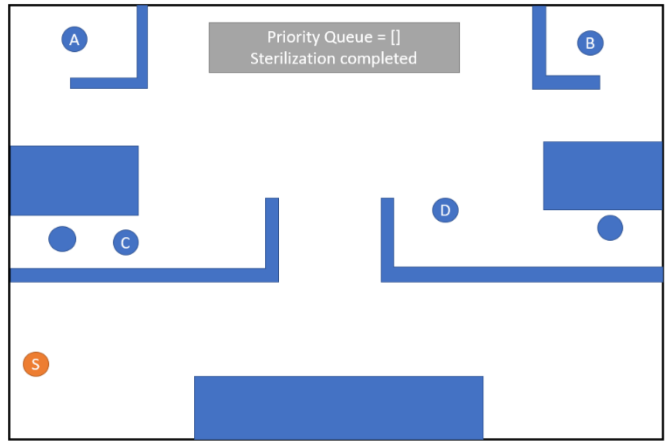

Fig 8. Environment after Sterilization

While implementing we figured out that there are good chances that the step 1, where we calculate the centroid of the plane formed by the contamination point might lie in obstacle space. To overcome it, a default point was set such that it is equal to the center of the map which does not lie in obstacle space and this point will be accessed by the algorithm if the calculated point is in obstacle space or not valid.

IV. Results

The tests were conducting by writing a code which justifies the algorithm suggested in Methodology. Let us look at the results step by step to see if basic search-based algorithms can be implemented for path planning in indoor environments.

Here, three contamination points were selected from entire map randomly.

When the tests were conducted for Dijkstra algorithm, on an average total time taken for calculating the path for every point was 330 secs.

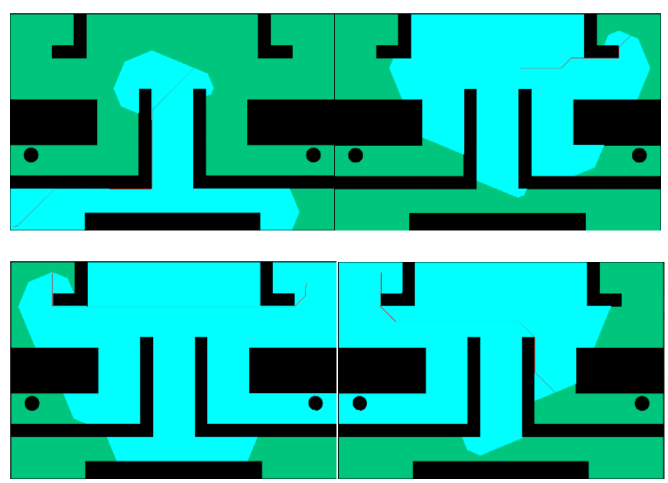

Fig 9. Results demonstrating Explored nodes for Algorithm using Dijkstra

The Dimensions of the map are (500×750) nodes, hence in case where the calculated Centroid falls in the obstacle space, the default co-ordinates for centroid will be $(200,325)$

```
Start point: (10, 10)

Contamination points (x co-ordinate, y co-rodinate, contamination percent) Before Prioritizing based on Contamination percentage
(475, 100, 30)
(450, 680, 50)
(200, 500, 20)

The Priority Queue based on Contamination Percent [(50, (450, 680)), (30, (475, 100)), (20, (200, 500))]

contamination Point 1:  (450, 680)

contamination Point 2:  (475, 100)

contamination Point 3:  (200, 500)

Centroid for all the contamination point is:  (375, 426)
```

Fig 10. The calculations in Dijkstra algorithm.

When the tests were conducted for A* algorithm, on an average total time taken for calculating the path for every point was 140 secs.

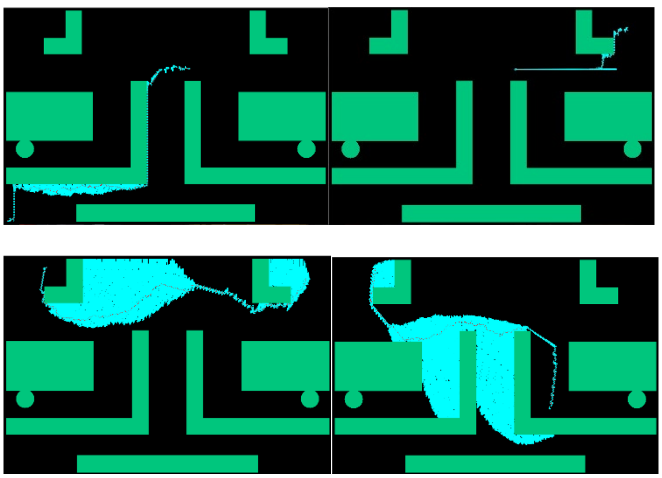

*Fig 11. Results demonstrating Explored nodes for Algorithm using A**

Comparing Figures 9 and 11, we can conclude that the points explored in both are different. When we use Dijkstra, the number of nodes increases which also increases the processing speed. Whereas, for A* it is less, hence decreasing the processing time.

Figures 10 and 12, demonstrated the calculations used for computing during test cases. And Figure 13 displays the planned path.

Let us compare both search-based algorithms for our case. (Demonstrated in Table 3)

Property	Dijkstra	A*
Search Algorithm	Greedy Best First search	Best First Search
Platform	Python	Python
Time efficiency	Less	High
Number of explored nodes	High	Less
Usage of Heuristics Function	No	Yes
Time Complexity	$O(n \log n)$	$O(n^2)$
Solution	Optimal	Optimal

Table 3. Comparison of observations and key features.

V. Conclusions

These algorithms can be integrated with Robot Control Software with multi-functional GUI and take out the load of manually guiding the robot at every step and reduce Human error.[3] By making the manual navigation secondary and converting that into more of human monitoring at most of the steps, increases the accuracy of system.

Based on the hardware specification (UV light used in robot), a manual radius can be set such that the path between contamination point can be further optimized. This helps us consider which algorithm should be implemented if we choose to use basic search-based path planning algorithm for navigation in UV sterilization robots.

By considering clearance values in the code, we can ensure that while determining the shortest path, the robot does not collide with the obstacles in the environment.[7] Also, the accessibility of data between the trio makes sure that the information with respect to the obstacles is always up-to-date.

Dijkstra is the best uninformed search algorithm to reach the goal point, but it only considers cost required to come to the next node. As it expands in every direction equally, it increases speed required to reach goal node and the number of explored nodes.

A* is much better fit in our situation as we can compute the cost to come and cost to goal i.e. heuristic function, and this can be calculated in our case. Although, information required to compute is less in Dijkstra than A*, the processing speed for A* is much faster than Dijkstra.

Hence, we can see that both algorithms have their own unique importance. A* best fits the indoor environment as the case of infinite points is rare (as the indoor are will be finite), also heuristic approaches are close to human way of behavior learning which makes it more efficient if implemented in a robot system which have machine learning capabilities.[5]

This will help us understand how we can choose between the search-based algorithms while implementing them in UV sterilization robots for improving the navigation technique.

```
Start point: (10, 10)

Orientation of start point: 60

RPM 1: 30

RPM 2: 60

Contamination points (x co-ordinate, y co-rodinate, orientation, contamination percent) Before Prioritizing based on Contamination percentage
(475, 100, 60, 30)
(450, 690, 30, 40)
(150, 500, 90, 20)

The Priority Queue based on Contamination Percent [(40, (450, 690)), (30, (475, 100)), (20, (150, 500))]

contamination Point 1:  (450, 690)

contamination Point 2:  (475, 100)

contamination Point 3:  (150, 500)

(X,Y) Co-ordinates for the centroid for all the contamination point is:  (358, 430)

Orientation for centroid is default at:  30

Centroid for all the contamination point is:  (358, 430, 30)
```

Fig 12. The calculations in A algorithm*

Fig 13. Demonstration of path for both the algorithms

ACKNOWLEDGMENT

I would like to thank Professor Reza Monfaredi for his constant support and guidance. We are grateful for having him as a mentor and for helping us in every aspect of this project. He has provided useful incites on improving performance of the project and in making it more robust.

REFERENCES

[1] Pala, Marco, et al. "HCTNav: A Path Planning Algorithm for Low-Cost Autonomous Robot Navigation in Indoor Environments." *ISPRS International Journal of Geo-Information*, vol. 2, no. 3, Sept. 2013, pp. 729–748., Accessed 11 Apr. 2019.

[2] P. Chanprakon, T. Sae-Oung, T. Treebupachatsakul, P. Hannanta-Anan, and W. Piyawattanametha, "An Ultra-violet sterilization robot for disinfection," *2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, 2019.

[3] El Zant, Chowkis, et al. "*UV-Robot Supervision System Design and Development*.", July 2018, www.researchgate.net/profile/Mhammed_Sahnoun2.

[4] Rubæk, Thomas, et al. "*Evaluation of the UV-Disinfection Robot*.", Blue Ocean Robotics - for Humans, 2016, prhoinsa.com/images/pdf/wuvd/UVDR-Whitepaper.pdf.

[5] B. Hernandez and E. Giraldo, "A Review of Path Planning and Control for Autonomous Robots," *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, 2018.

[6] Anantha Sai Hari Haran V Injarapu and S. K. Gawre, "A survey of autonomous mobile robot path planning approaches," *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, 2017.

[7] J. K. Goyal and K. Nagla, "A new approach of path planning for mobile robots," *2014 International Conference on Advances in Computing, Communications, and Informatics (ICACCI)*, 2014.

[8] L. Ou, W. Liu, X. Yan, Y. Chen, and J. Liang, "A Review of Representation, Model, Algorithm and Constraints for Mobile Robot Path Planning," *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2018.