2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI) | 978-1-7281-9544-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/SACI51354.2021.9465624

# Computational Capacity Analysis of Platforms for Low-Cost Autonomous Ultraviolet Germicidal Robots

Sergio Genilson Pfleger
*Graduate Program of Computer Science (PPGCC)*
*Federal University of Santa Catarina*
Florianópolis, Santa Catarina, Brasil
sergio.genilson@posgrad.ufsc.br

Patricia Della Mea Plentz
*Graduate Program of Computer Science (PPGCC)*
*Federal University of Santa Catarina*
Florianópolis, Santa Catarina, Brasil
patricia.plentz@ufsc.br

***Abstract*—The use of Ultraviolet Germicidal Irradiation (UVGI) to disinfect environments gained prominence during the current pandemic of COVID-19, as a method to reduce the contamination. Given the risks that the technique can bring to human health and other living beings, autonomous robots become useful for the application of UVGI. This document presents a study of the technical feasibility of using some platforms as a processing and control unit for low-cost autonomous disinfectant robots. For this, each platform is subjected to the UVGI irradiance calculation, a task that requires a high processing capacity. The results show that all tested platforms meet the processing capacity criteria, being eligible to compose a low-cost disinfectant robot.**

***Index Terms*—UVGI Robot, Low-cost Disinfectant Robot, Autonomous Robot, Irradiance Calculation**

## I. Introduction

Disinfection by Ultraviolet Germicidal Irradiation (UVGI) is a known technique and used for more than 60 years for several proposes of many sectors of the industry [13]. However, the application of UVGI also poses risks to human health and other living things [13], making this one of the areas where autonomous robots can be immediately useful during the COVID-19 pandemic [19]. Although commercial solutions for these robots already exist, the cost to acquire them is impractical for most institutions [4], mainly in developing countries. In addition to costs, unfortunately, there are not enough of these robots to meet demand at the moment and, in many cases, the project is private and cannot be analyzed or used by other researchers. In this way, the development of low-cost robots for this purpose becomes relevant.

Ultraviolet light (UV) can be divided in four primary bands: UV-A (320nm - 400 nm); UV-B (280nm - 320nm); UV-C (200nm - 280nm); and VUV (100nm - 200nm). Only UV-C and UV-B have germicidal effects and therefore considered UVGI [13]. UV-C light are absorbed by nucleic acids, which makes it especially damaging to cells and effective against micro-organisms [13].

For the disinfection process to be effective, that is, for the elimination of viruses, fungi or bacteria, it is necessary that UVGI be applied in adequate doses [13]. The UV doses can be different for microorganisms on surfaces, suspended in the air or immersed in water [13]. These doses are tabulated for most of the most common microorganisms. For the SARS-CoV-2 virus, which caused the COVID-19 pandemic, the application of UV-C light proved to be a reliable method for disinfection [12]. For complete virus inactivation, the required dose is 1,048 mJ/cm$^2$ [12].

Given the context of the pandemic, mobile robots prove to be an interesting solution, as they may be able to calculate the ideal dose to be applied in the target environment, making them effective in the objective of sterilizing environments. With the ideal dose calculated, it will be possible to determine the amount of time required in each mapped region of the scenario. This mapping allows defining trajectories for the robot and some optimizations, such as minimizing time and energy to sterilize the environment.

Several efforts have been made to develop vaccines in record time. However, making billions of doses available for immunization on a global scale takes time, planning, financial and chemical resources. In addition, no long-lived antibody responses are induced by human coronaviruses infections, and re-infections can occur after an extended period of time [1]. Also, new variants of the virus are emerging all the time. Studies are investigating whether the mutations impact natural and vaccine-mediated immunity [9]. For these reasons, the development of complementary technologies, with the capacity to reduce the spread of the virus has its importance.

This work aims to explore the processing capacity of some common computational platforms, which are easily found on the market and which can compose a low-cost robot. To explore the platforms capacity, the UVGI dose calculation for each point in the scenario will be performed. This task requires high processing capacity and that configures a critical requirement for the project. Parallel computing techniques will

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and CAPES/PRINT - Call no. 41/2017 Senior Visiting Professor at Biomimetics and Intelligent Systems Group - BISG, University of Oulu, Finland.

be used in order to explore all processing units (cores) of each platform. So, it will be possible to determine if each platform has sufficient processing capacity to compose a low cost robot, which is the general objective of this research where this work is inserted.

## II. Related works

The literature presents recent research work on the use of UVGI for disinfecting various types of environments. The majority of the papers are focused on present solutions to the health environments: hospitals [6], operating rooms [3], and masks as the respirators N95 [14] [11]. Few papers consider other environments, like schools, train and subway stations which are environments with a huge people agglomeration capacity [18] [4].

The authors [6] and [3] investigate the use of UV-C light for disinfecting hospital environments. The first one proposes the development of a robot for disinfecting operating rooms and hospital rooms. This robot uses UV light and can be operated manually or autonomously. In the proposal, the authors describe that navigation will happen throughout the environment at the same time that UV light is emitted in 360$^{o}$ around the robot. The focus of the research is the disinfection of Staphylococcus aureus bacteria (S. aureus), which is quite common in operating rooms and can cause serious surgical complications.

In [3], the authors presents a review article on service robots in medicine, with an emphasis on robots for disinfection in medical institutions. The author describes a set of patented technologies that together form a robot with a UV-C disinfection system that automatically measures the conditions of the environment (room size, temperature and humidity), in order to determine, in real time, the intensity and the appropriate duration of UV-C light, time, number and power of the lamp that are necessary for complete disinfection of the environment. As the technologies used in the solution are proprietary, the scientific community has no access (the codes are closed). In addition, there is no discussion of possible navigation algorithms that can maximize results (improve navigation time in the environment, optimize the use of UVC lamps, perform efficient scanning of the environment, etc.).

In [14] and [11] the use of UV-C light is presented as a possibility for the elimination of SARS-CoV-2 respirators N95, used by health professionals. There is also a discussion about the dosages of UV-C light needed for this purpose and the damages that high dosages can cause to the material of the masks.

Vachhani [18] present research on the challenge of intensive online computing required for robotic navigation when using a Light Detection and Variation (LiDAR) system for Simultaneous Location and Mapping (SLAM) and obstacle detection, for autonomous UV-C based robots. The navigation algorithm is developed for indoor environments, more specifically small rooms.

Bentancor and Vidal [4] propose a programmable low-cost UV-C robot for disinfecting environments. The project costs around USD 180 and can be operated remotely via Bluetooth, using an Android device. However, the correct positioning of the robot and the correct definition of the radiation time depends on human action. An error in the amount of time can lead to a waste of resources and time, when configured an excessive time. Or ineffective sterilization can occur when time is not sufficient and positioning is not adequate. Thus, an experienced operator is required to operate the equipment.

In [8], the authors analyze the cleanup process inside hospitals, assessing that disinfection robots complement the manual cleaning routine. Moreover, they ask for disinfection processes should be validated, reproducible, and clearly documented. A hardware overview of an autonomous mobile robot platform for medical purposes is presented in [2]. That robot and can be manually controlled, and its modular design can receive different subsystems. One of these is a UV-C lamp irradiation subsystem. The focus of the paper is on hardware components and how to integrate them. There are no descriptions of the software and how the disinfection process is carried out

In [10], the authors describe different devices for UV-C disinfection and propose a UVC-robot that operates manually and autonomously. Concerning UV-C devices, they present the following classification: disinfection unit, complementary devices, combined disinfection agents, mobility devices, and order type. For each category, it is presented its operation and some characteristics. The I-Robot UV-C is the mobile robot proposed in that paper, and the authors list the main steps to construct it. There is no detailed explanation about the modules that compose the robot. Moreover, nothing is said about the navigation algorithm and its relation to the disinfection process.

A comparison between traditional disinfection methods with static UV-C lamps (static process) against the use of mobile robots with coupled UV-C lamps in a continuous movement (dynamic process) is shown in [17]. Through experiments, the authors conclude that the dynamic process improves disinfection accuracy and reduces task execution time. This assumption proposes a disinfection motion-planning strategy based on a genetic algorithm (GA) to optimize robot navigation. Case studies were applied in a real environment where the UV-C robot shows its effectiveness. Although the proposed strategy presents promising results, the authors do not comment on the UV-C robot's financial cost. There are several solutions in the market with variable costs. However, none of them delivers low-cost robots targeting developing countries where outbreaks as coronavirus will take more time to be solved.

Through this literature review, it was possible to observe the research gap in the proposal and development of low-cost autonomous mobile robots for disinfection purposes in general environment. This is even more important for public environments, like schools and trains / subway stations, in developing countries.

The project in which this work is inserted proposes the development of a low-cost robot, which can operate autonomously, capable of defining the positioning and dosage of UVGI necessary for a specific type of microorganisms.

Low-cost platforms will be evaluated in order to verify that they have satisfactory computational capacity to be the main processing unit of a low-cost robot in the UVGI application task necessary to exterminate the target microbe in the target scenario. In addition, it will be shown which of them can be used as a main processing unit in the computational platform.

## III. Problem Formulation

Autonomous robotic navigation is a complex activity, which depends on several factors, such as characteristics of the environment in which the robot must navigate and the characteristics of the robot itself. In the case of a disinfectant robot, the task that the robot needs to complete is: application of the UVGI dose necessary to exterminate the target microbe, at each point in the scenario. In this way, the calculation of the amount of light that hits each point of the scenario is crucial to define whether the task was completed or not.

According to [13], the dose of light that hit a point in the scene can be calculated by the equation

$$D = T \cdot I \tag{1}$$

where $T$ is the exposure time ($s$) and $I$ is the irradiance ($\mu W/cm^2$).

Irradiance, considering a cylindrical lamp, can be calculated using the equation

$$I = \frac{E_{uv}}{2\pi r l} F \tag{2}$$

where $E_{uv}$ is the UV power of lamp ($\mu W$), $r$ and $l$ are the radius and length of the lamp ($cm$), respectively, and $F$ is the radiative irradiance.

The fraction of radiative irradiance, in turn, according to [13], can be calculated using the equation:

$$F = \frac{L}{\pi H}\begin{bmatrix} \frac{1}{L} atan\left(\frac{L}{\sqrt{H^2-1}}\right) - atan(M) \\ + \frac{X-2H}{\sqrt{XY}} atan\left(M\sqrt{\frac{X}{Y}}\right) \end{bmatrix} \tag{3}$$

where the parameters of (3) are defined as follows:

$$H = x/r$$
$$L = l/r$$
$$X = (1+H)^2 + L^2$$
$$Y = (1-H)^2 + L^2$$
$$M = \sqrt{\frac{H-1}{H+1}}$$

and $x$ is the distance between the lamp and the point in the scenario where the light flux will be calculated.

There are several types of sensors on the market capable of determining the distance between the robot and each point in the scenario. Figure 1 shows a frame of a depth map captured using a Xbox 360 Kinect V1 sensor, used in the experiments. Darker colors represent closer objects. Lighter colors represent more distant points. Finally, points where it was not possible to determine the distance are represented in white.

From the depth map, the irradiance for each point on the map can be calculated, using (2). The Algorithm 1 illustrates

Fig. 1. Example of a depth map captured using Xbox Kinect.

the implementation of this calculation. First of all, the depth map is loaded from a video file with the depth map (in a experimental case, like described below) or directly from the depth sensor (in a real case robot navigation).

Then, for each new depth map frame, the irradiance is calculated, pixel by pixel. The *x* value of the pixel is proportional to the depth (distance between the imaged object and the sensor) and is used as an input parameter to calculate irradiance. The *calcIrradiance(x)* function is a simple implementation of (2), using the distance *x* as a parameter. Variables related to the lamb specifications, such as power, length and radius are constant.

**Algorithm 1** Irradiance Calculation

```
1:  video ← loadVideo(videoFile)
2:  while frame ← video.getFrame() do
3:      show(frame)
4:      irradiance ← emptyFrame()
5:      for i = 0 to frame.height do
6:          for j = 0 to frame.width do
7:              x ← frame[i, j]
8:              irradiance[i, j] ← calcIrradiance(x)
9:          end for
10:     end for
11:     show(irradiance)
12: end while
```

The Algorithm 1 also shows the captured depth frame, with *show(frame)*, in line 3, and the result of the irradiance calculation, with *show(irradiance)*, in line 11. The time used to perform this function is small, compared to other processing times, and can be considered equal to zero, in the experiments presented in the results section.

Fig 2 displays the irradiance map, calculated for the depth map of Fig 1. Lighter colors represent a higher incidence of light. Darker colors represent less incidence of light.

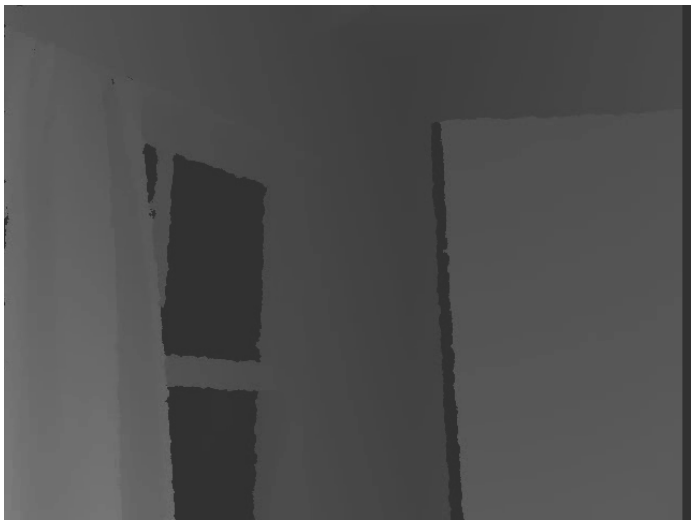

Fig. 2. Irradiance map calculated from the depth map.

## IV. Parallelization of the Algorithm

Modern processors can have multiple cores and threads and to exploit their computational capacity it is necessary to use parallel computing techniques.

The Algorithm 1 has been implemented[1] in C++, using OpenCV (*Open Source Computer Vision Library* [5]) for the operations over the depth map frames.

To implement the parallelization of Algorithm 1, the API (*Application Programming Interface*) OpenMP (*Open Multi-Processing* [7]) was used. OpenMP allows you to define directives in the original code that determine the pieces of code that must be executed in parallel. Execution starts at the main thread and, at run time, when the execution reaches the parallel region, OpenMP creates a group of threads (worker threads) to execute the code for this region. At the end of the parallel code snippet, the threads synchronize again, and sequential execution continues.

The OpenMP directives were added in order to run in parallel the code between Lines 5 and 10 of the Algorithm 1. Thus, considering the use of $n$ threads, each thread processes a block in the depth map of size $1/n$.

## V. Tested Platforms and Experimental Environment

To carry out the experiments, 3 different machines were used, which can be shipped in a low cost robotic prototype:

- Notebook computer equipped with Intel 4$^{th}$ generation processor, model i7-4710MQ, with 4 cores and 8 threads, 16 GB of RAM and mechanical HD as a storage unit, running Ubuntu 20.04.
- Raspberry PI 2 equipped with ARMv7 rev 5 processor, with 4 cores and 4 threads, 1GB of RAM and Memory Card as a storage unit, running Ubuntu Mate 18.04.5 LTS.
- Notebook computer equipped with a 10$^{th}$ generation Intel processor, model i3-1005G1, with 2 cores and 4 threads, 8 GB of RAM and SSD as a storage unit, running Linux Mint 20.

Considering that the objective is to compare the performance of the processing of the light irradiance model, a sequence of depth maps was prepared, with 746 frames, with 640x480 pixels, obtained using an Xbox Kinect sensor. The sequence was stored as a video[2] so the experiment can be repeated, and was used in the three test environments, in order to have the same scenario for comparison. As the objective of this work is to explore the computational capacity of the different platforms to calculate the applied light dose and not the sensors themselves, the sensor that was available during this papper (Xbox Kinect) was used. Considering the general objective of developing a low cost robot, other lower cost sensors can perfectly take the place of the Kinect sensor.

The determining factors for the experiment are the number of frames and the resolution of the video. So, it is not necessary to use others depth map sequences for the experiment. Likewise, the accuracy of the depth map and the ranges of values are not relevant in this experiment itself, since the objective is to explore the processing capacity of the platforms.

On all test machines, the version of OpenCV used was version 4.2, installed via .deb packages available directly in the application repository of each of the respective Linux distributions.

Considering the dose calculations presented, the time required to sterilize an environment, like a room, for example, depends on the UV lamps specifications and the size of the room. It is desirable that the robot be able to sterilize an room as quickly as possible. However, that can't be done instantly due the technical limitations. In this way, was defined as a project requirement that the robot need to be capable to sterilize a room with 20 m$^2$ in less than 10 minutes. Also, the robot must be able to process the irradiance 360$^o$ around the robot at minimum of 8 times during the sterilization process, each one from a another position in the room, to minimize shadow areas. That is, times less than 1 minute and 15 seconds to process the irradiance throughout the robot's surroundings, can be considered acceptable for the processing time. Processing times longer than 1 minute and 15 seconds will increase the total time for the sterilization and will not be satisfactory.

The use of Graphics Processing Units (GPUs) in the experiments could bring an expressive processing capacity. However, as the goal is to keep the cost of the project low, the use of conventional GPUs has been discarded in the first analysis.

According to [16], Xbox Kinect has a 57$^o$ Field of View (FoV). Therefore, approximately 6.3 depth map captures would be needed to cover the 360$^o$ arround the robot. In this way, if a platform is capable of processing 6.3 depth maps in 1 minute and 15 seconds (or 5.04 per minute), it can be

[1]Both single thread and multi threaded versions are available at https://github.com/sergiogenilson/irradiance-calculation.

[2]Available at https://youtu.be/w3iO_29kMOY.

considered as a platform capable of calculating the UVGI dose at run time.

## VI. Results

Figures 3, 4 and 5 shows the average processing time for each number of threads, for the 746 depth map frames, as specified previously, using Intel i7-4710MQ, Raspberry PI 2 ARM and Intel i3-1005G1 processors, respectively. The experiments were repeated four times for each number of threads on each processor.

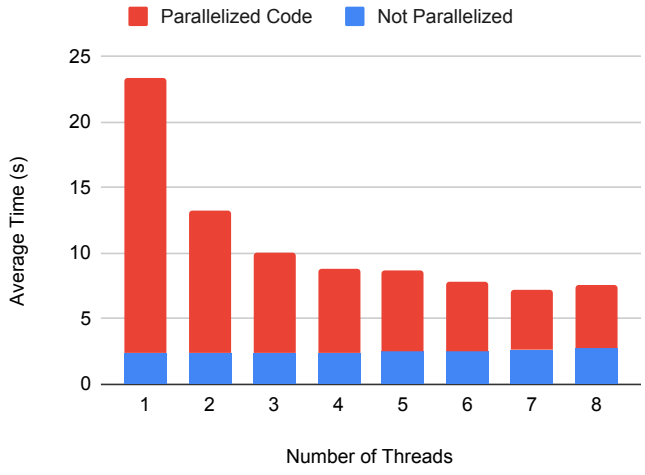


Fig. 3. Processing time, in seconds, by the number of threads, for the Intel i7-4710MQ processor.

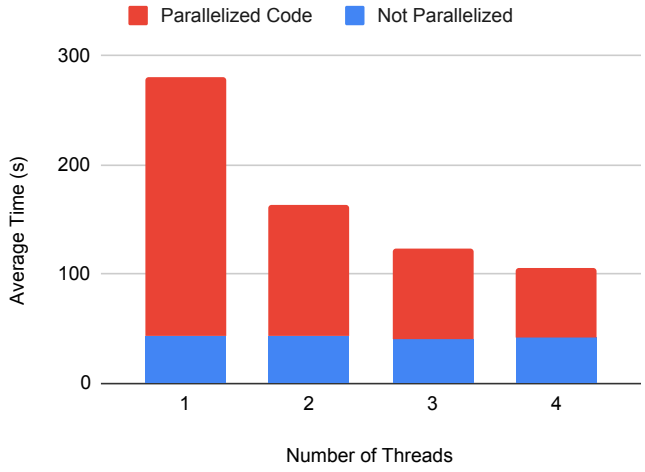


Fig. 4. Processing time, in seconds, by the number of threads, using Raspberry PI 2 ARM processor.

The parallelization of the algorithm resulted in a performance gain, as expected. The red part refers to the time needed to execute the code portion responsible for calculating the irradiance in the environment, which was parallelized in this work. The blue part refers to the time needed to execute

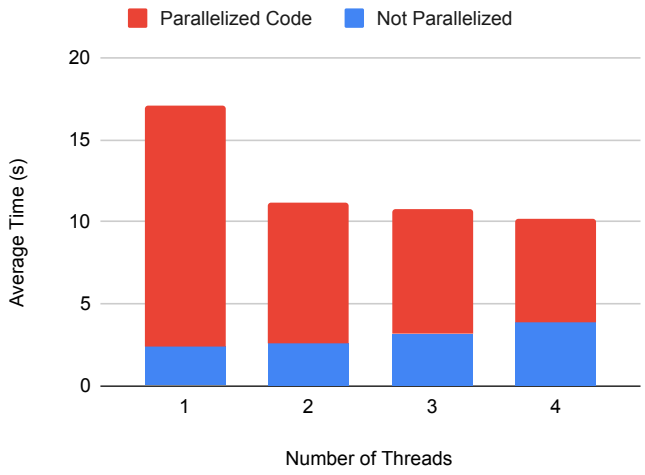


Fig. 5. Processing time, in seconds, by the number of threads, for the Intel i3-1005G1 processor.

other parts of the code, responsible for reading new frames, format conversions and displaying the image on the screen. This amount of code was not the focus of parallelization in this study, being considered the serial part of the code.

In all the tested platforms, a significant gain was noticed with the parallelization of the irradiance algorithm.

The time spent on the sequential portion of code is approximately constant (for each processor) and independent of the number of threads. A small increase in processing time was observed for the i3-1005G1 processor, when using 3 and 4 threads (processor has only 2 physical cores, with support for 4 threads) in Fig 5. The processing portions related to the sequential part of the code remained similar on the two Intel platforms, even though they are of different generations and using storage units with different technologies. However, on the Raspberry the time required to process the sequential part of the code was substantially longer than on Intel platforms, as expected due the constructions differences of each type of platform.

Since the sequential portion of code takes a constant amount of time, the more threads are used, the greater the impact of the unparallelized code, representing 35.51% of total time for the Intel i7-4710MQ processors, 38.36% for the i3-1005G1 processor and 39.22% for the Raspberry. So, some optimizations in the serial portion of code can also represent great improvements in the algorithm performance as a whole and should be explored in the future.

Analyzing the case of Raspberry PI 2, in Figure 4, which is the cheapest platform with the most limited resources among the tested platforms, it took 279.5 seconds to process the 746 frames of the depth map, using one thread, and 105.2 seconds using the four available threads. In other words, the platform is capable of processing 2.7 depth maps per second, using one thread, and 7.1 depth maps per second using all four available threads. In this way, it can be considered that the Raspberry

PI 2 platform has computational capacity much higher than the stipulated minimum (5.04 per minute), being eligible to be shipped on the low cost robotic prototype for the purpose of controlling the robot.

Fig 6 shows speedup, that is, the performance gain, for each of the platforms, considering the total processing time. The speedup is calculated dividing the single thread time by the time spent by $n$ threads, where $n$ are the number of threads [15]. The parallelization of the code brought performance gains, in all tested platforms.

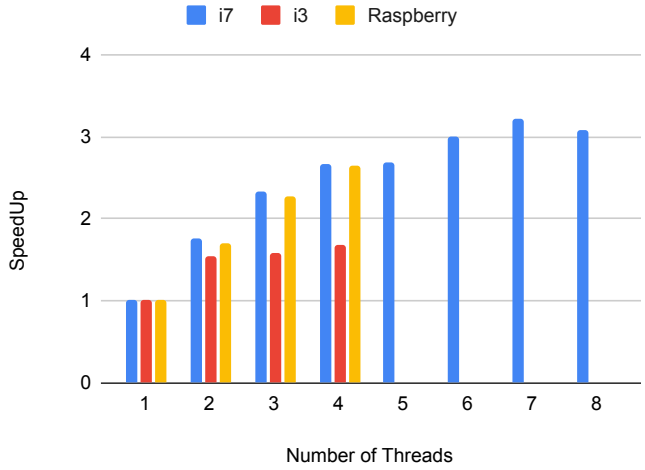


Fig. 6. Speedup for each of the tested platforms, considering the total processing time.

## VII. Conclusions

Considering the COVID-19 pandemic caused by the SARS-CoV-2 virus, researching ways to eliminate this virus is of paramount importance.

In this paper, a performance study of three target computational platforms which could be used in the development of low-cost mobile robots for disinfection purposes is presented.

Despite the significant differences in the total processing time between the Intel and ARM platforms, the experiments showed that all the tested platforms have the ability to process irradiance in the environment at run time. The Raspberry PI 2, which is the cheapest platform and the most resources limited, was able to perform 7.1 depth maps per second (frames per second). So, all platforms analyzed can be considered eligible to compose a low-cost robot.

The parallelization of the algorithm proved to be efficient in the problem addressed, resulting in a speedup of up to 3.2X, for the Intel i7-4710MQ, 2.7X for the Raspberry PI 2 and 1.7X for the Intel i3-1005G1.

Considering the need to process other algorithms together with this (location, construction of 3D maps and trajectory generation, among others), in order to constitute a functional autonomous robot, it is also necessary to evaluate the performance of the other algorithms to determine if each platforms have sufficient computing capacity for the desired application. This analysis will be performed in future works.

## References

[1] Fatima Amanat and Florian Krammer. Sars-cov-2 vaccines: Status report. *Immunity*, 52(4):583–589, 2020.

[2] Catur Hilman AHB Baskoro, Hendri Maja Saputra, Midriem Mirdanies, Vita Susanti, Muhamad Fahrur Radzi, and Rizal Imam Abdul Aziz. An autonomous mobile robot platform for medical purpose. In *2020 International Conference on Sustainable Energy Engineering and Application (ICSEEA)*, pages 41–44. IEEE, 2020.

[3] Aladin Begić. Application of service robots for disinfection in medical institutions. In *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*, pages 1056–1065. Springer, 2017.

[4] Marcel Bentancor and Sabina Vidal. Programmable and low-cost ultraviolet room disinfection device. *HardwareX*, 4:e00046, 2018.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Pacharawan Chanprakon, Tapparat Sae-Oung, Treesukon Treebupachatsakul, Pimkhuan Hannanta-Anan, and Wibool Piyawattanametha. An ultra-violet sterilization robot for disinfection. In *2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, pages 1–4. IEEE, 2019.

[7] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55, 1998.

[8] Magda Diab-El Schahawi, Walter Zingg, Margreet Vos, Hilary Humphreys, Lorena Lopez-Cerero, Astrid Fueszl, Jean Ralph Zahar, and Elisabeth Presterl. Ultraviolet disinfection robots to improve hospital cleaning: Real promise or just a gimmick? *Antimicrobial Resistance & Infection Control*, 10(1):1–3, 2021.

[9] Nathan D. Grubaugh, Emma B. Hodcroft, Joseph R. Fauver, Alexandra L. Phelan, and Muge Cevik. Public health actions to control new sars-cov-2 variants. *Cell*, 2021.

[10] Moez Guettari, Ines Gharbi, and Samir Hamza. Uvc disinfection robot. *Environmental Science and Pollution Research*, pages 1–6, 2020.

[11] Iltefat H. Hamzavi, Alexis B. Lyons, Indermeet Kohli, Shanthi Narla, Angela Parks-Miller, Joel M. Gelfand, Henry W. Lim, and David M. Ozog. Ultraviolet germicidal irradiation: Possible method for respirator disinfection to facilitate reuse during the covid-19 pandemic. *Journal of the American Academy of Dermatology*, 82(6):1511–1512, 2020.

[12] Christiane Silke Heilingloh, Ulrich Wilhelm Aufderhorst, Leonie Schipper, Ulf Dittmer, Oliver Witzke, Dongliang Yang, Xin Zheng, Kathrin Sutter, Mirko Trilling, Mira Alt, Eike Steinmann, and Adalbert Krawczyk. Susceptibility of sars-cov-2 to uv irradiation. *American Journal of Infection Control*, 48(10):1273–1275, 2020.

[13] Wladyslaw Kowalski. *Ultraviolet germicidal irradiation handbook: UVGI for air and surface disinfection*. Springer science & business media, 2010.

[14] Shanthi Narla, Alexis B. Lyons, Indermeet Kohli, Angeli E. Torres, Angela Parks-Miller, David M. Ozog, Iltefat H. Hamzavi, and Henry W. Lim. The importance of the minimum dosage necessary for uvc decontamination of n95 respirators during the covid-19 pandemic. *Photodermatology, Photoimmunology & Photomedicine*, 36(4):324–325, 2020.

[15] Peter Pacheco. *An introduction to parallel programming*. Elsevier, 2011.

[16] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 139:1–20, 2015.

[17] Luca Tiseni, Domenico Chiaradia, Massimiliano Gabardi, Massimiliano Solazzi, Daniele Leonardis, and Antonio Frisoli. Uv-c mobile robots with optimized path planning: Algorithm design and on-field measurements to improve surface disinfection against sars-cov-2. *IEEE Robotics & Automation Magazine*, 28(1):59–70, 2021.

[18] Birju Vachhani. *Autonomous mobile robot navigation with GPU acceleration for unmanned UV-C based decontamination applications*. PhD thesis, Rutgers University-School of Graduate Studies, 2019.

[19] Guang-Zhong Yang, Bradley J Nelson, Robin R Murphy, Howie Choset, Henrik Christensen, Steven H Collins, Paolo Dario, Ken Goldberg, Koji Ikuta, Neil Jacobstein, et al. Combating covid-19—the role of robotics in managing public health and infectious diseases, 2020.