electronics

MDPI

Article

Development of Cloud Autonomous System for Enhancing the Performance of Robots' Path

Kaushlendra Sharma [1,2,†], Rajesh Doriya [1,†], Sameer Shastri [3,†], Turki Aljrees [4,†], Kamred Udham Singh [5,6,*,†], Saroj Kumar Pandey [7,*,†], Teekam Singh [8,†], Jitendra Kumar Samriya [9,†] and Ankit Kumar [7,*]

1 Department of Information Technology, National Institute of Technology Raipur, Raipur 492010, India
2 Department of Computer Science & Engineering, Indian Institute of Information Technology, Nagpur 440006, India
3 Department of Computer Science and Engineering, Bhilai Institute of Technology Durg, Durg 491001, India
4 College of Computer Science and Engineering, University of Hafr Al-Batin, Hafar Al-Batin 39524, Saudi Arabia
5 Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan
6 School of Computing, Graphic Era Hill University, Dehradun 248002, India
7 Department of Computer Engineering & Applications, GLA University, Mathura 281406, India
8 School of Computer Science, University of Petroleum and Energy Studies, Dehradun 248007, India
9 Department of Computer Science & Engineering, National Institute of Technology, Delhi 160058, India
* Correspondence: 11004033@gs.ncku.edu.tw (K.U.S.); saroj.pandey@gla.ac.in (S.K.P.); kumar.ankit@gla.ac.in (A.K.)
† These authors contributed equally to this work.

Abstract: With the development of computer technology and artificial intelligence (AI), service robots are widely used in our daily life. At the same time, the manufacturing cost of the robots is too expensive for almost all small companies. The greatest technical limitations are the design of the service robot and the resource sharing of the robot groups. Path planning for robots is one of the issues playing an important role in every application of service robots. Path optimization, fast computation, and minimum computation time are required in all applications. This paper aims to propose the Google Cloud Computing Platform and Amazon Web Service (AWS) platforms for robot path planning. The aim is to identify the effect and impact of using a cloud computing platform for service robots. The cloud approach shifts the computation load from robots to the cloud server. Three different path-planning algorithms were considered to find the path for robots using the Google Cloud Computing Platform, while with AWS, three different types of environments, namely dense, moderate, and sparse, were selected to run the path-planning algorithms for robots. The paper presents the comparison and analysis of the results carried out for robot path planning using cloud services with that of the traditional approach. The proposed approach of using a cloud platform performs better in this case. The time factor is crucially diagnosed and presented in the paper. The major advantage derived from this experiment is that as the size of the environment increases, the respective relative delay decreases. This proves that increasing the scale of work can be beneficial by using cloud platforms. The result obtained using the proposed methodology proves that using cloud platforms improves the efficiency of path planning. The result reveals that using the cloud computing platform for service robots can change the entire perspective of using service robots in the future. The main advantage is that with the increase in the scale of services, the system remains stable, while the traditional system starts deteriorating in terms of performance.

Keywords: cloud robotics; cloud computing; path planning; service robots; path time

Citation: Sharma, K.; Doriya, R.; Shastri, S.; Aljrees, T.; Singh, K.U.; Pandey, S.K.; Singh, T.; Samriya, J.K.; Kumar, A. Development of Cloud Autonomous System for Enhancing the Performance of Robots' Path. *Electronics* **2023**, *12*, 683. https://doi.org/10.3390/electronics12030683

Academic Editor: Antonio Brogi

Received: 22 December 2022
Revised: 15 January 2023
Accepted: 17 January 2023
Published: 29 January 2023

1. Introduction

The pace of growing technology has redefined the meaning and concept of traditional approaches. Nowadays, computing is considered one of the important resources and is

being utilized to strengthen the traditional approach to solving problems. Robot path planning is one such area where the requirement of high computation power is common sense these days. Robots are deployed in a dynamic environment, and a group of robots are employed in many places to accomplish different tasks. Handling such environments requires a smart approach that needs high computation. This paper attempts to implement the concept of conflict-based pathfinding while exploiting the advantages of cloud computing. One of the most sought-after and practical implementations of path-finding algorithms is to guide intelligent agents such as robots and smart vehicles (both virtual and real) along a path that is most suitable for their objective. The requirement is that smart agents perform tasks, and therefore, they need to be able to reach the location where the work is to be accomplished. There are a plethora of path-finding algorithms, and a considerable number of studies and implementations have been carried out on this topic. Yet, the inherent issue with finding the best paths remains the same [1]; that is, as the environments increase in size and become more complex in nature, the task of finding paths starts to become more resource-intensive. No matter how optimized and efficient and implementation is, the physical limitations are always there. To compute a solution, computational resources are required [2,3].

Another prospect of this endeavor, in practical scenarios, is that almost always, multiple intelligent agents would be traversing the environment while dealing with static as well as dynamic obstacles. Naturally, if there are more agents and complex environments, more resources are required to find a solution. There are many ways to approach this problem, the most obvious of which is to give each agent enough computational capabilities and resources to find its own path in a given environment. While this can be a viable option, it might not be suitable for practical applications, not everywhere at least. Independent agents also pose another layer of complexity when it comes to cooperation. So, this paper attempts to provide a viable option for scenarios in which multiple agents have to traverse the same environment in cooperation with each other. Instead of giving each agent the freedom to move and decide as they choose, in this study, we sought to implement a centralized system to give them two necessary instructions that are needed to traverse any given environment in the most efficient way possible.

The centralized system, if implemented, has to accomplish the work of all the agents, and that means it needs computational capabilities and resources that are on par with all the agents at the very least. The central system has to communicate with all the agents and has to analyze the environment as well. This is where cloud computing comes in; instead of setting up a local centralized system to guide the intelligent agents, cloud-based services can be utilized to provide a viable and acceptable solution. After the rather daunting amount of progress that has been made when it comes to cloud services, they are more relevant than ever. Offloading complex and/or intensive tasks to a system that has a very large amount of computational resources at its disposal is logical, especially after the massive increase in both the bandwidth and availability of high-speed internet throughout the world. This evolution of networking and cloud-based services is only going to improve from the current state. That has some very real and plausible implications for the future of computing. This paper is an effort to capitalize on those very implications [4].

2. Related Work

This section deals with the research conducted on cloud-based robots. It is required to make robots compatible to integrate with companion computers with ease. This is becoming the mainstream demand these days [5]. The concept of Roboearth proposed by Waibel et al. is one of the most cited works in this field [6]. In the last few years, the use of a cloud computing platform for robotic applications has garnered the attention of researchers but not enough to explore all its possibilities and scope. Here, in this section, some of the studies are discussed in detail. Table 1 gives a short outline of some of the prominent research conducted in recent times [6,7].

There is a high need for service robots that can intelligently and reliably support people with their activities because of the difficulties that Lam et al. identified for service robots in unstructured environments, such as the inability to properly interact and cope with unstructured surroundings. As a rule, robots are on the diminutive side, and their processing capacity is not up to tackling complex computational jobs. Because of its potential to facilitate resource sharing, parallel computation, and cost reduction, a cloud-based robot is a possible solution to the issues with tiny robots. The proposed system architecture for cloud-based path-planning systems consists of three layers: a server layer, whose job it is to make it easier for robots to share structured and massive datasets; a cloud engine layer, which provides a lightweight virtual machine called a container in which authorized programs can execute using rapyuta; and a client robot layer, which is made up of the collection of physical robots that use PaaS in the cloud. Path planner uses multithreading to make efficient use of available computing power. We can determine whether it is preferable to run a node in the cloud or on a robot based on three criteria: the availability of local processing resources; the nature of the node; and the capacity of the surrounding network [2,8].

In their paper, Wee et al. proposed challenges in networked robots, the architecture of cloud robotics, and applications of cloud robotics. Initially, the trend was to use onboard networked robotics, which faces many constraints such as resource constraints, meaning that the efficiency of the network robot is bounded by size, configuration, power supply, motion code, and functioning condition. Another challenge was information and learning constraints, which means that the information of the network robot is bounded by processing power, storage capacity, and the number and type of sensors it carries. The next challenge discussed was communication constraints, i.e., for communication networks, robots provide proactive routing, which requires huge estimation and memory resources in route detection and maintenance process, as well as ad hoc routing, which takes high latency. In this paper, we divided the architecture of cloud robotics into three subsystems: middleware subsystem, background task subsystem, and control subsystem. The middleware subsystem deals with robots and protocols of the network to make them compatible with each other. The goal of the background task subsystem is to maintain packages and libraries for programming languages. The control subsystem is used to control networking, storage, and computation. This study also reveals some technical challenges in cloud robotics such as computational challenges, communication challenges, and security challenges [9].

The review of the literature that we performed before the advent of our project helped in shaping the contours of our project in a more meaningful way. An overview of all the relevant literature on our project is an essential part of our documentation. Gualong hu and Wee Peng Tey, in their paper entitled "Cloud Robotics: Architecture, Challenges and Applications" proposed a cloud robotic architecture to address the problems faced by current standalone and network hardware robots and also the applications that can be useful for the cloud robotic approach. They proposed a software platform that uses the Hadoop cluster to reduce the latency and overhead time of normal robots. Table 1 depicts some of the notable contributions in recent times in the field of cloud robotics. The table presents a small description of papers based on the issues of cloud robotics.

Table 1. Different cloud computing platforms used for robot-based applications in recent times.

Author	Year	Application/Issues Addressed	Cloud Technology	Parameters	R.I	Ref
Mohd et al.	2019	Multi-Robot system with different team sizes for the fire searching model is performed. A comparison of the traditional model HDec- POSMDPs with the cloud-based model is presented with different performance metrics.	Google Cloud	Meantime, Number of Turns, Average Energy Consumption of robots.	236	[10]
Hao et al.	2018	This research paper deals with the semantic map building for intelligent service tasks and has used both private and public cloud for the experiment.	Cloud Stack	Operating Efficiency and average execution time of the algorithm	06	[11]
Turnbul et al.	2013	This paper discusses the cloud infrastructure to develop the formation control on a multi-robot system in order to run robotic applications. A robot with minimal hardware was constructed to work within the control of the cloud.	Private Cloud	Success Ratio and Space Complexity	65	[12]
Duo et al.	2016	This paper proposes a vehicular cloud computing testbed with mobile robotics for testing various network protocol behavior to measure message throughput using a middleware server.	Rabbit MQ	Message Communication and No. of Movement	17	[13]
Zhihui et al.	2017	This paper proposes a new environment for robots to monitor different service request distribution using a cloud server and also to monitor the scheduling policy and robot center solution.	Cloud Stack	Scheduling Policy, Request Distribution	57	[14]
Long et al.	2020	Robot-mind-centered service system is proposed to exploit the features of conventional robotics systems using a cloud platform. It also used the techniques of AI and 5G technology, Multifunctional intelligent clothing.	Private Cloud	Intelligence of the follow-me Robot-Mind	02	[9]
Liu et al.	2019	Autonomous navigation problem of robots is taken into consideration to address the issue. Private cloud architecture is proposed to measure the parameters of navigation and some reinforcement learning methods for the robots in the cloud. The paper also proposes a knowledge fusion algorithm for upgrading a shared model deployed on the cloud.	Private Cloud	Performance of transfer learning approaches and transfer Reinforcement Learning	43	[15]
Sandeep et al.	2019	This paper deals with the issue of the Robot offloading problem (sensing task) using cloud architecture to improve the accuracy and to minimize the cost of cloud communication using deep reinforcement learning.	Private Cloud	Sensing task, Vision task performance and Accuracy	18	[16]

3. Proposed Architecture

The term "cloud robotics" was introduced in 2010 by James Kuffner, an ex-employee of Google. Afterward, it started to raise interest among researchers [12]. Recently, Goldberg et al., a prominent name in the field of cloud robotics, described the security issues and privacy concerns present in cloud robotics in their research paper. This paper proposes to solve the robot path-planning problem using the Google Cloud Computing Platform. The continuous development of cloud technologies has spawned the concept of cloud-based robots. Google Cloud is one of the big players in the current market in terms of cloud-based

service providers. Google's Cloud Robotics Core is also available nowadays. This is an open-source platform that provides the infrastructure essential to building and running robotic solutions for business automation [17]. The first step required to set up the platform for robots is to create a Google account for it, i.e., a Google cloud account. The next step is to select the VM instance of a particular size required for the project. Once this is achieved, the setting up of the Python environment is required. The next step is to implement the path-planning algorithms on the cloud computing platform to observe the outcome of the parameters.

We proposed a solution to set up a cloud platform and run the path-planning algorithms on the cloud. Figure 1 represents the working sequence of the path-planning algorithm for finding an optimized path for robots. Figure 2 represent the Basic architecture of AWS Platform for creating an instance. Figure 3 shows the basic structure of the Google Cloud Computing Platform, which was taken as a reference to set up the cloud computing platform to run the path-planning program for robots. Figure 4 represents the block diagram of the proposed architecture to run the program and all the requirements of packages and algorithms to execute robotic path planning using the Google Cloud Computing Platform. Figure 5 shows the working flow of path-planning algorithms in the cloud environment. The flowcharts explain the requirements of packages and their sequence of execution indicating how the algorithm is implemented for execution.

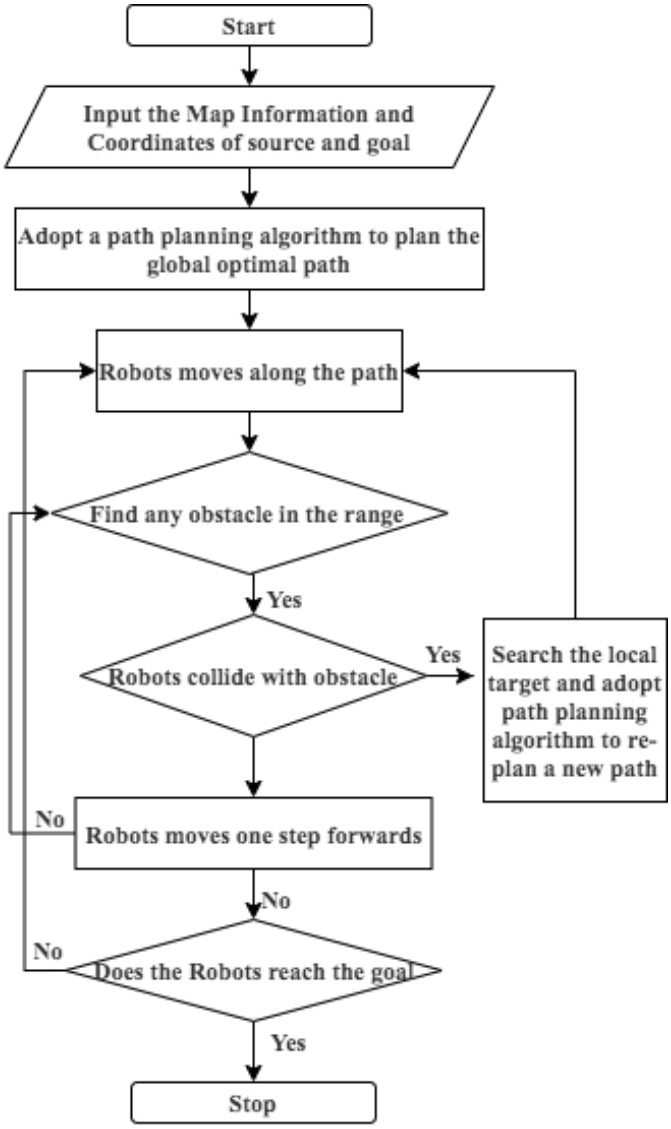


Figure 1. Proposed model as flow of path-planning algorithm.

In creating the VM Instance in the proposed work, we considered the following configuration: The configuration of the virtual machine instance created for our project is 8, 23 CPU capacity with 30 GB. After creating the VM instance, the firewall settings were configured, and after installation, all the dependencies were completed. The next step was to create a Python environment; setting up the Python environment was achieved by installing all the required packages and launching Jupyter notebook where the path-planning algorithms were implemented. The proposed architecture at the cloud level provides essential packages to run robot path planning in the cloud environment. Only the data and commands that are necessary for the direct execution of given tasks are delivered to the robots as the users of the cloud services. Furthermore, as described, the proposed architecture is of modular type and can be adjusted and modified for other scenarios as well. The pseudocode is presented in Algorithm 1, which shows the working of the networking module for both the client and server in the cloud setup. The next section will discuss in detail the various important reasons to implement cloud computing platforms for robot path planning as well as some of the key points of the path-planning problem that should be precisely considered while deriving the path from its source to destination. The path should be collision-free and optimized for traversal.

To validate the results with more justification, AWS [18,19] is another cloud platform that was utilized here to perform the A* star algorithm in three different reference maps. Three variants of each environment were considered for testing the code. The solution utilized EC2 instances from Amazon Web Services [20] to do the heavy lifting. The idea is that most of the complex and resource-intensive tasks would be performed by the EC2 instances, as they are scalable and cost-effective at the same time. Figure 2 presents the basic architecture of AWS for creating EC2 instances using the lambda function.

The agents, instead of finding their own path, will be waiting for directions from the client. The client will first parse the information about the environment to the EC2 instance. Then, the client will fetch and relay the starting as well as the goal positions of the respective agents to the EC2 instance as well. The EC2 instance will process the data received from the client to find those paths that are not conflicting. At this stage, the client will act purely as a relay, i.e., the direction is just sent to the respective client that is provided by the EC2 instance until they have reached their destination. It will notify the EC2 instance about the same, and the connection will end, thus concluding a successful traversal through the directions received. An EC2 instance will act like a server, constantly running and waiting for a client to request a connection. If a client connects, the requests will be fulfilled, and the connection will end, allowing another (or the same) client to connect again. With this method, the system can handle more than one local system of mobile agents.

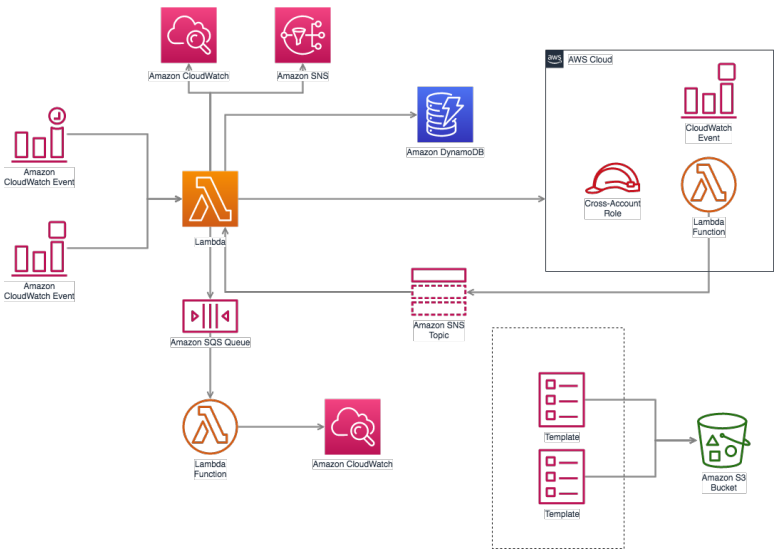


Figure 2. Basic architecture of AWS Platform for creating an instance.

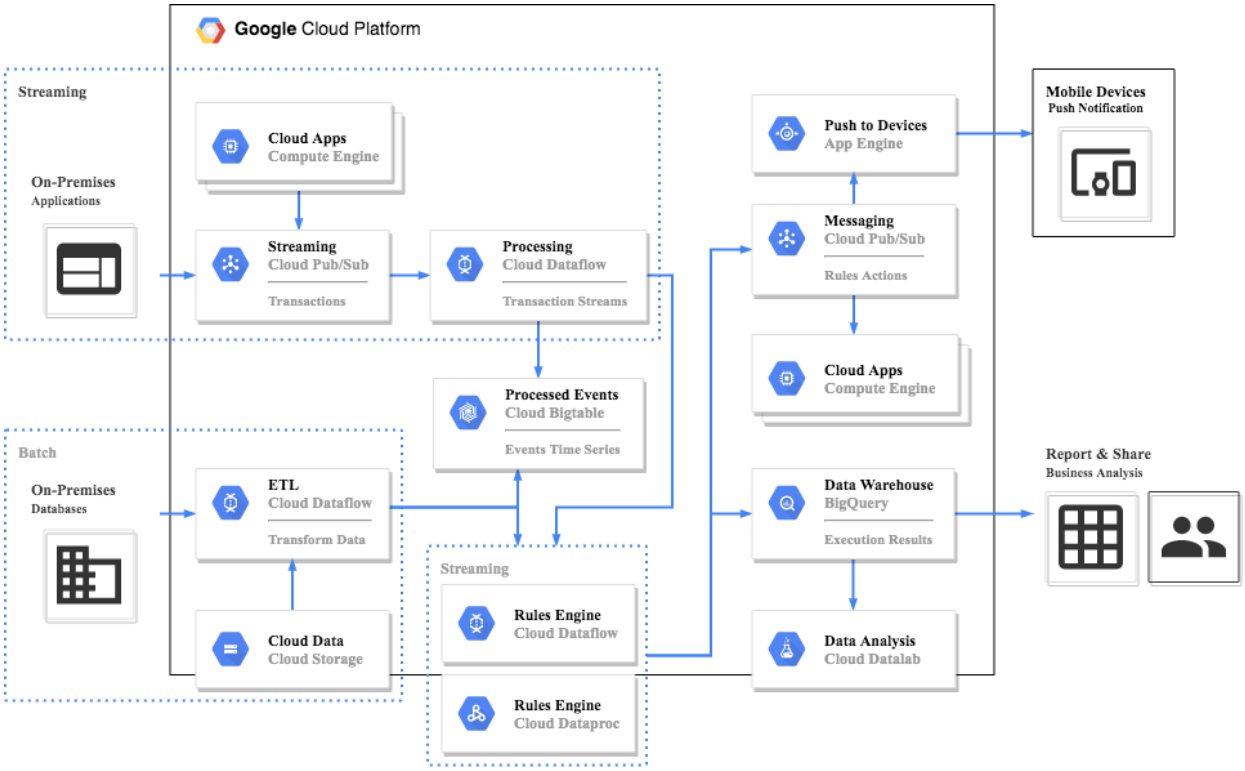


Figure 3. Basic architecture of Google Cloud Computing Platform for mounting applications and programs on top of it.

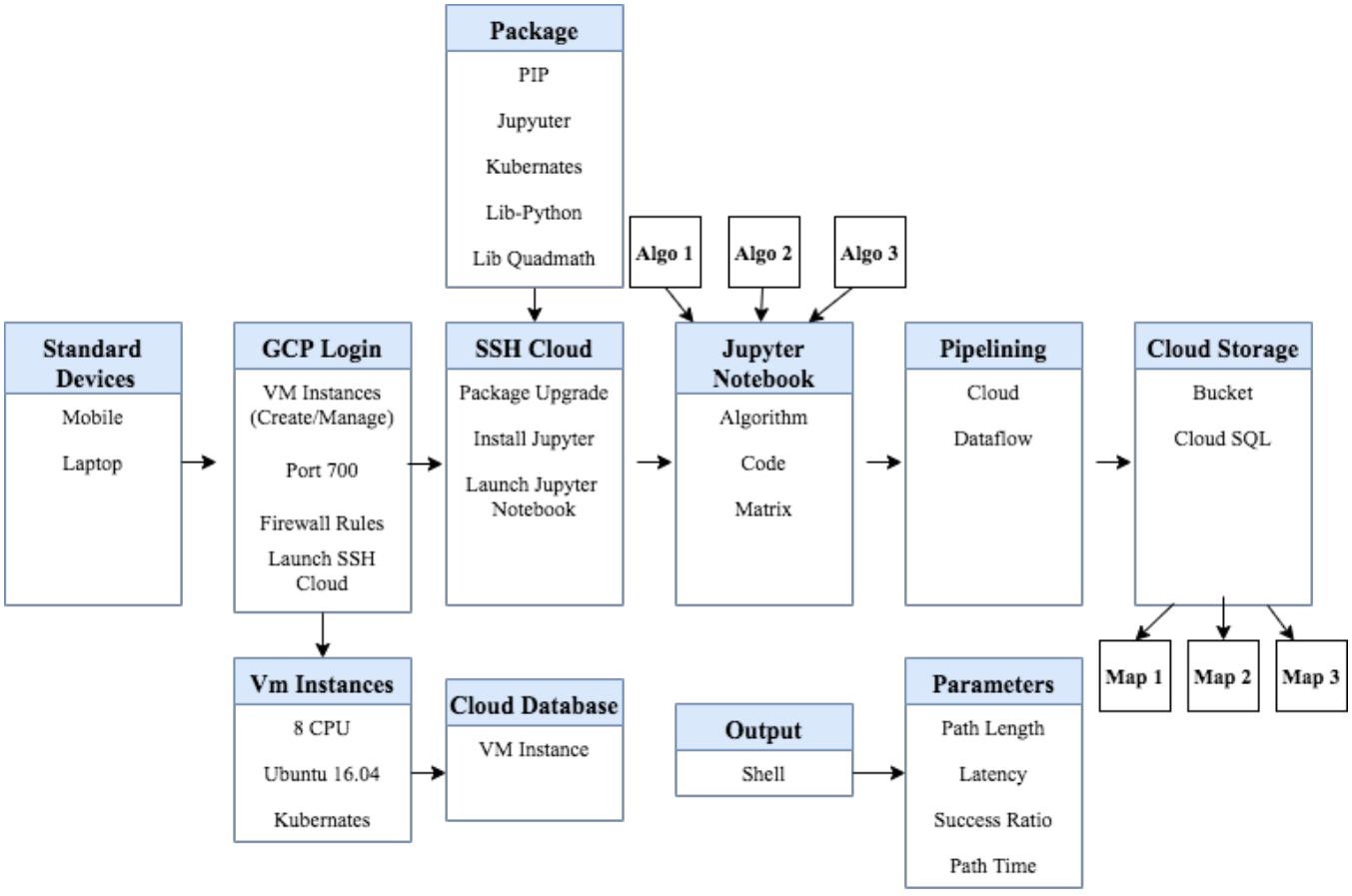


Figure 4. Proposed architecture for robot path planning using Google Cloud Computing Platform.

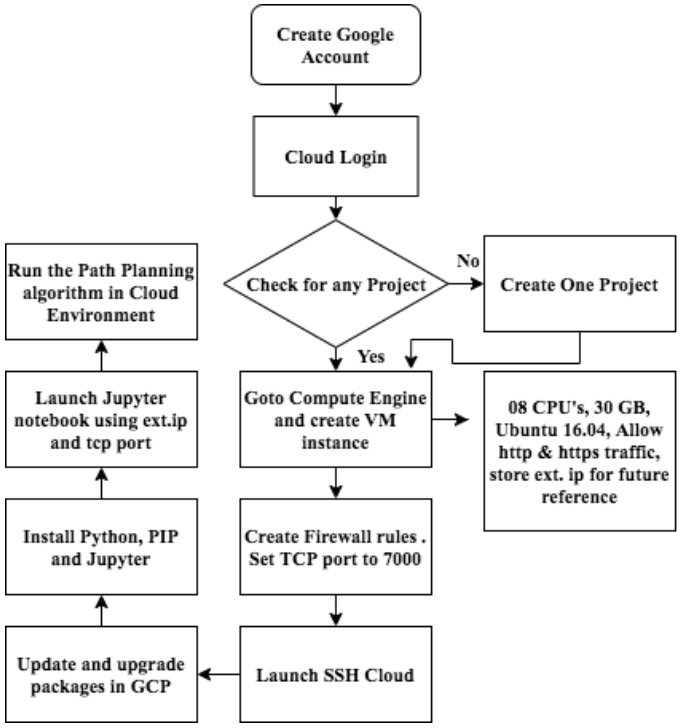


Figure 5. Setting up of cloud environment.

Algorithm 1 Networking Module for Cloud Server

Get user input for the coordinates
For i in range (*no of agents*):
  sCoord = (*sX*, *sY*)
  gCoord = (*gX*, *gY*)
Initialise the client
client = *client* ('15.206.191.18, 4005, 1024', *notify* = *True*)
Send the data & receive notification from server
  client.send (*Map*)
  client.send (*Start*)
  client.send (*Goal*)
  sendingTimeEnd = *currentTime*()
agent[]
Create a List of Agents that are all on their starting positions
Command agents to move according to the directions received
move=1
data =*client.receive*()
if data == True:
  break
else :
  move+ = 1
For i in range (*no of agents*):
  agents[*i*].*move*(*data*[*i*])
  agentTimeEnd=*currentTime*()
End Agent Time

4. Robot Path Planning Using Cloud Computing Platform

The concept of cloud-based robots started early in 2010 when the problem with standalone robots came into the picture; the capacity of a single hardware robot was insufficient, and gathering information took a long time and caused delays, which was also the case of networked hardware robots in which robots were interconnected and shared

information; if one robot runs out of service, then the information contained in that robot that had to be shared cannot be forwarded to that particular robot [12]. Then came the concept and use of cloud-based robots, where all the necessary information is kept in the cloud itself in a centralized way, and it is not required by the robot to store each and every piece of information for performing a particular task [21]. The use of cloud robotics was recently developed, where there are centralized storage and control from where the other robots take the required message and perform that particular task. The importance of using cloud computing for robot path planning can be understood with the help of the following points:

Service robots are specifically made for a particular type of environment. When building service robots, two factors are more important: The first is the hardware requirement, and the other one is the software requirement. The hardware part mostly resembles all types of service robots. The differentiation is made based on the programming embedded in it for accomplishing a specific task. The limitation of this process is that physical robots are not portable and flexible to be used by other applications or in some other required place, though the hardware requirements do not change much; the only feature that becomes a hurdle is the software part. As already a specific type of programming is already performed and embedded into it, to make it more usable and operational, there is a need to physically reprogram it, which may require handling it manually and can increase the entire cost of applying robots. This problem can be easily resolved by introducing cloud computing technologies for robots. As there is no limitation of space or computational power at the cloud server, once the robots connect to the cloud server, they may obtain the required program instantly from the cloud server and can operate in the environment accordingly.

Considering the problem of robot path planning, a service robot can instantly access the best result-oriented algorithms for any specific environment (Maze, Random Obstacles, or Maze Maps) from the cloud server's repository and can optimize the solution to a greater level. Even the applied intelligence at the cloud server can suggest or compute by applying the best technique for the robots at the server itself, which is not possible when a robot is physically localized in any environment.

The other issue is the use or requirement of space. A physical robot can have a limited memory as per the requirement of the operation. Any change to be made requires physical interruption to it. This is where cloud computing comes in and plays an important role. The robot can access the space more or less as per the requirement. Space availability at the cloud server is scalable and can store any volume of data for the robots. This flexible use policy has multiple advantages; for instance, all the executions of robots can be stored and monitored at the cloud end. Several intelligent techniques can be applied to it to perform much better in the next iteration. It may also help in finding a suitable technique based on the past performance of robots, which can help in achieving optimized results. In the case of multiple robots, there is no need to traverse the environment multiple times by each robot. All the robots can share their traversal with the cloud, which can be coordinated and executed in a much better way by using cloud computing.

The coordination of robots in a multi-robot environment is one of the important and crucial tasks that need to be handled with precision. This could be more deliberately handled with the help of a cloud computing platform. There are several types of coordination such as centralized coordination, incremental coordination, and distributed coordination, and sometimes, a hybrid approach is considered the best option. Again, applying the best techniques needs a clear understanding of the current scenario, some past experience, and intelligent mechanisms to apply the best technique in the time of need. These requirements make it complicated to provide an optimum solution for the run time. The only issue that becomes a big hurdle is the availability of time, space, and computational power. The presence of a cloud computing platform can handle this issue with ease.

Another prospect of this endeavor, in practical scenarios, is that almost always, multiple intelligent agents would be traversing the environment while dealing with static and dynamic obstacles. Naturally, if there are more agents and complex environments, more

resources are required to find a solution. There are many ways to approach this problem, the most obvious of which is to give each agent enough computational capabilities and resources to find its own path in a given environment. While this can be a viable option, it might not be suitable for practical application, not everywhere at least. Independent agents also present another layer of complexity when it comes to cooperation. Thus, this study sought to provide a viable option for such scenarios where multiple agents have to traverse the same environment in cooperation with each other. Instead of giving each agent the freedom to move and decide as they choose, in this project, we implemented a centralized system to give them the necessary instructions that are needed to traverse any given environment in the most efficient way possible. The centralized system, if implemented, has to accomplish the work of all the agents. This means that it needs computational capabilities and resources that are on par with all the agents at the very least. The central system has to communicate with all the agents and analyze the environment as well. This is where cloud computing comes in; instead of setting up a local centralized system to guide the intelligent agents, cloud-based services can be utilized to provide a viable and acceptable solution.

The goal of cloud robotics is to have complex algorithms run in the cloud while more basic robots operate on the ground. Robots' computational loads are greatly lightened when data processing is moved to high-performance cloud servers [22]. Each additional robot added into the system would have to independently replicate the experiences and learning of its predecessors if not for the cloud-connected network. However, when robots are networked together in the cloud, previously learned information may be recycled [23]. There are some important concepts in robot path planning that are mandatory to consider such as path generation, tracking control, and obstacle detection shown in Table 2 and in Algorithm 2. The robot's linear velocity v_p is controlled by satisfying Equation (1). ω_p is the angular velocity given by Equation (2). Equation (3) defines the nearby obstacle definition for the robot. Equation (4) defines the assigned velocity for robots [24,25]. Equation (5) defines the formula for finding the shortest path shown in Algorithm 3.

$$v_p = \begin{cases} \frac{G_n - P_n}{D_{bf}} V_{max}, |G_n - P_n| \leq D_{bf}, \\ V_{max}, |G_n - P_n| > D_{bf}, \end{cases} \tag{1}$$

$$\omega_p = \begin{cases} \frac{\Delta\theta}{D_{bf}} \omega_{max}, |\Delta\theta| \leq \theta_{bf}, \\ \frac{\Delta\theta}{|\Delta\theta|} \omega_{max}, |\Delta\theta| > \theta_{bf} \end{cases} \tag{2}$$

$$\boldsymbol{d_o} = \sum_{1} \left[(1 - \frac{|\boldsymbol{d_i}|}{D_{safe}})^{\alpha} (\frac{\boldsymbol{d_i}}{|\boldsymbol{d_i}|}) \right] \tag{3}$$

$$\omega_o = \begin{cases} \frac{V_p}{|X_o|}, \theta \leq 90 \\ \frac{-V_p}{|X_o|}, \theta > 90 \end{cases} \tag{4}$$

$$lv = \sqrt{((x[i] - x[i-1])^2 + (y[i] - y[i-1])^2)} \tag{5}$$

Table 2. Definition of the variables taken into the equations.

Variables	Definition
v_p	*Controlling linear velocity of robots*
G_n	*Target for the nth robot*
P_n	*Current position of the nth robot*
D_{bf}	*Parameter for the buffer distance*
V_{max}	*Maximum speed of the robot*
ω_p	*Robots angular velocity*
$\Delta\theta$	*Angle between current & next position*
θ_{bf}	*Next buffer angle*
ω_{max}	*Maximum angular velocity*
$\boldsymbol{d_o}$	*Obstacle Vector with distance & direction*
D_{safe}	*Safe distance between robot & obstacle*
$\boldsymbol{d_i}$	*Vector for the obstacle*
ω_o	*Assigned angular velocity*
lv	*Path length of robot*
$\|X_o\|$	*Distance between robot & next point*

Algorithm 2 A* Algorithm for finding path

Input : *A graph*
Output : *Path between start and goal nodes*
repeat
  pick n_{best} *from O such that* $f(n_{best} \leq) f(n), \vee_n \in O$
  Remove n_{best} *from O and add to C*
If n_{best} =q_{goal}, *EXIT*
  Expandn_{best} *for all* $xStar(n_{best})$ *thatare not in C*
If $x \in O$ *then*
  add x to O
else if $g(n_{best}) + c(n_{best}.x) < g(x)$ *then*
  update x'*s backpointer to point to* n_{best}
end if
untill O is empty

Algorithm 3 PFA Algorithm for finding path

Input : *A graph*
Output : *Path between start and goal nodes*
$t = 0$, $x_c(0) = x_{start}$, $Flag = 0$, *Calculate the potential function*
while Next decision is not goal do
  if $Flag = 0$ *then*
    Change the cell weight and treat the cells as occupied by an obstacle
    Update the potential of each cell
  end
end
if the UAV is trapped in a cell or visit the same cells multiple times then
  if $Flag = 1$,
    search for the trapping point
      if $Flag = 0$
  end
$x_{x+1} \leftarrow$ *cell nearby with lowest potential function*
$t \leftarrow t + 1$
end

5. Results and Discussion

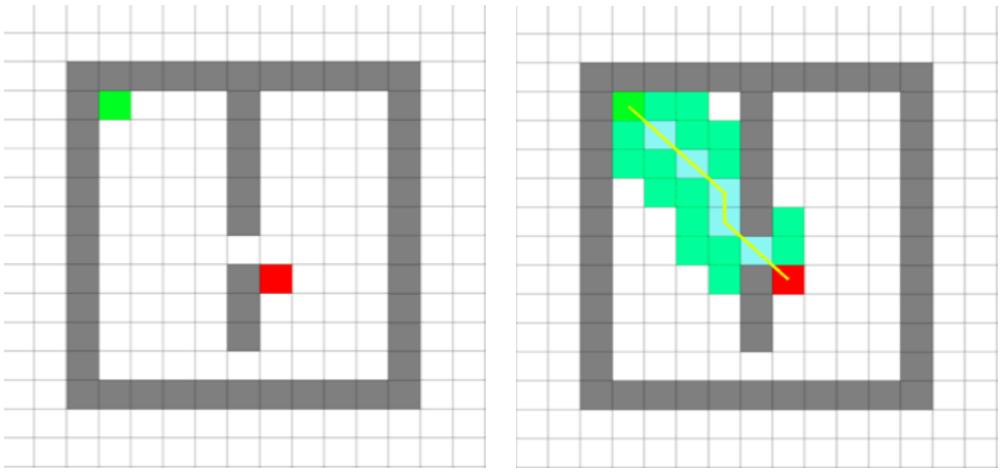
This paper presents the results in two parts. In the first part, we implemented the A* star algorithm and observed the time taken using the algorithm. The first step in doing so

was to choose a cloud service [26]. Here, we chose Google Cloud Platform (GCP) for this purpose, as they offered a free tier and an interactive console. The next step was setting up a virtual machine (VM) instance in the cloud platform. Following the interactive guide on Qwiklabs, it becomes easy to set up a VM instance and run SSH into a VM instance through the local machine without having to go to a web-based cloud console. The next step was to set up an n1-standard-2 machine having 2-CPU and 7.5GB RAM with Debian GNU/Linux 9 OS. The next step was benchmarking with a robot pathfinding algorithm (A* algorithm). Here, we took an A* star algorithm with a given maze in an executable Python program and timed its execution speed using the 'time' Unix tool, and recorded the execution times. The last step was to compare the local machine versus the cloud VM instance. Cloud VM instance performs better than the local machine in terms of speed and efficiency of execution of the program, as they are highly scalable and flexible in terms of CPU speed and memory, and they can be highly optimized for a specific task.

Table 3 presents the results obtained by using A* star algorithm using cloud and local machines. It is clear from the results that cloud VM instances for computing the same problem produced much better results than the operation performed using the local machine shown in Figure 6.

Table 3. Comparative study of time taken by the path planning algorithm using local machine and using Google cloud.

S.No	Local Machine (ms)	Google Cloud (ms)
1	182	93
2	203	102
3	173	87
4	152	71
5	196	84
Avg.	181	87



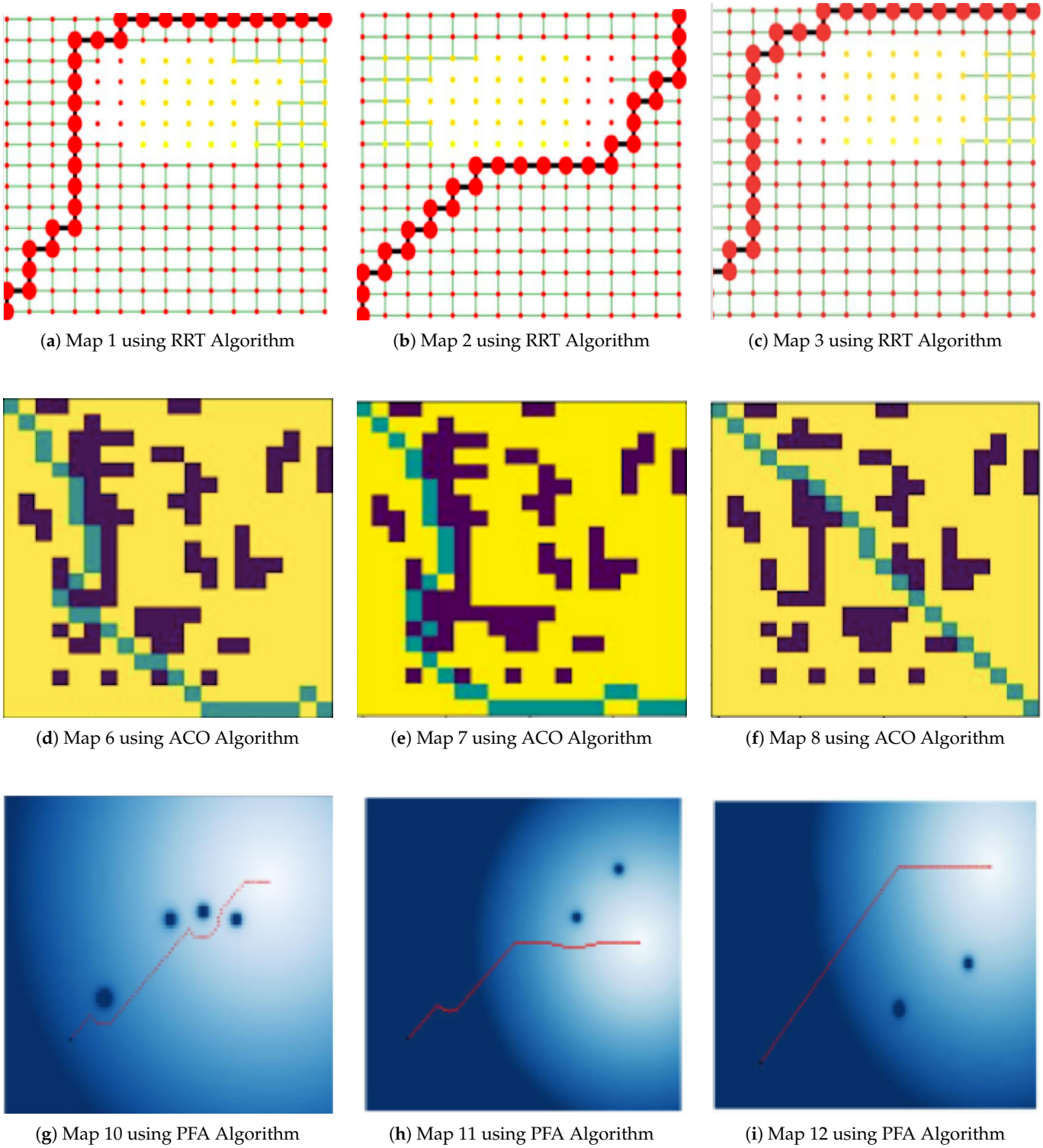
(**a**) PP using Local Machine (**b**) PP using Google Cloud

Figure 6. Output of path planning using A* Star algorithms using local machine (**a**) and Google Cloud Computing Platform (**b**).

After the successful run of the A* star algorithm for a maze map shown in Figure 6, we implemented the rapid exploring random-tree algorithm (RRT), ant colony optimization algorithm (ACO), and potential field approach (PFA) algorithm and tested with different maps. To validate the use of the cloud platform, we chose the algorithms that have their own aesthetic features, and to evaluate the performance of each algorithm shown in Figure 2, the different nature of its execution was investigated. Some key points covering their pros and cons are stated here.

Ant colony optimization (ACO) is one of the well-accepted algorithms for solving problems related to optimization. Due to the feature of having inherent parallelism and proficiency in solving travel salesman problems, it is used in many progressive applications [27,28]. The rapidly exploring random tree (RRT) algorithm is another algorithm considered in our work. Many modified versions of RRT are available for solving the problem of robot path planning. One of the important features of this algorithm is that it is quick, and no adjusting parameter is required initially. Its limitation is that its computational time is highly dynamic [29,30]. The next algorithm used in our work is the potential field approach (PFA) [31]. One of the greatest advantages of PFA is having a very fast computation time, whereas the limitation is that it produces local minima in which robots are trapped. Three different kinds of environments were tested with three different algorithms. Each individual environment was tested three times, and their averages were tabulated. The dataset to test against the system was randomly generated using scripts and/or programs so as to make sure the system can handle all kinds of scenarios. The result presented in Figure 7 shows the comparison of the time taken by the local machine with respect to the relative delay. Table 3 shows the relative delay (RD), which may be caused by network, hardware, and implementation issues following the trend of becoming less and less significant as the task at hand scales. To validate this observation, the second stage of the implementation included the execution of the algorithm in AWS using EC2 [32]. An EC2 instance acts as the brain of the system. A virtual machine instance can have any amount of computational capability and resources that are required as per the task at hand, making them a very scalable solution. To test the system, the dataset was divided into three major categories, and three different kinds of environments were tested. They were categorized based on the complexity of the environment, namely dense, moderate, and sparse environments. For each category, three sizes were tested as well: small, medium, and large environments in each category. Each individual environment was tested three times, and their averages were tabulated. The dataset to test against the system was randomly generated using scripts and/or programs so as to make sure the system can handle all kinds of scenarios [33].

In the second phase of the run, we considered the time taken by the algorithms, the sending time (ST) from the robot to the cloud, and the difference between the results of the local machine and VM instance (Diff.), where T_1 = the time taken by the cloud, T_2 = the time taken by the local machine, and the processing delay in nine different reference maps using Google Cloud and nine different maps using AWS. As evident from the results presented in Table 4 and the graph in Figure 8, the relative delay that may be caused by network, hardware, and implementation issues follows the trend of becoming less and less significant as the task at hand scales. Smaller environments have relatively larger delays, while larger maps have a relatively small delay. For each category, three sizes were tested as well, small, medium, and large environments in each category. Each individual environment was tested three times, and their averages were tabulated. The dataset to test against the system was randomly generated using scripts and/or programs so as to make sure the system can handle all kinds of scenarios. The data suggest that the performance overhead was almost constant or close to constant and that the same amount of time was always taken, irrespective of the task. The tests were fairly rudimentary, only focusing on the timings because, in the current day and age, CPU and RAM consumption does not really affect the results at all, especially because of the small scale and very basic structure of the project. There was not much strain on the hardware of any system.



(**a**) Map 1 using RRT Algorithm (**b**) Map 2 using RRT Algorithm (**c**) Map 3 using RRT Algorithm

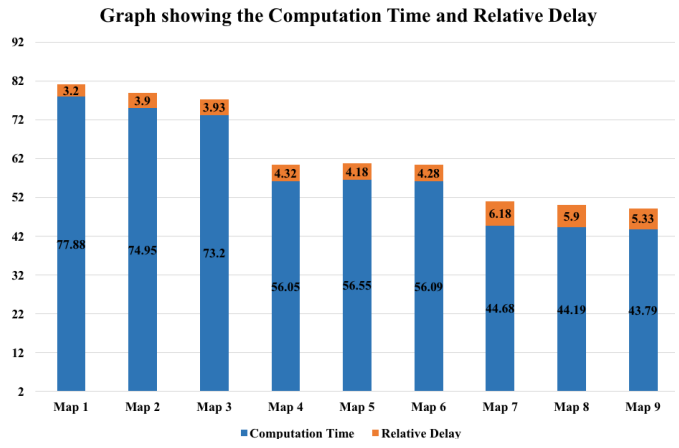
(**d**) Map 6 using ACO Algorithm (**e**) Map 7 using ACO Algorithm (**f**) Map 8 using ACO Algorithm

(**g**) Map 10 using PFA Algorithm (**h**) Map 11 using PFA Algorithm (**i**) Map 12 using PFA Algorithm

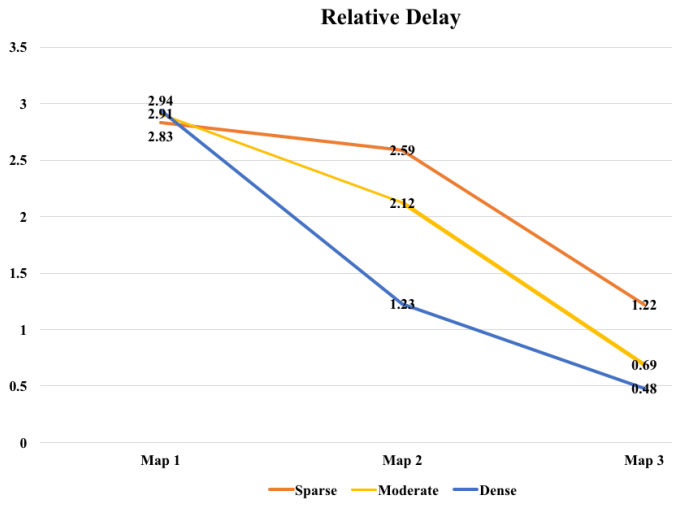
Figure 7. Output of path planning algorithms applied on different reference maps using Google Cloud Computing Platform.

Table 4. Comparison of results obtained, path length, and path time on four different reference maps using cuckoo search algorithm and proposed approach.

			Google Cloud Computing Platform			
Ref. Map	**Algo. Used**	**Sending Time**	**Time Taken by Agents**	**Expected Time Taken**	**Time Difference**	**Network Delay (%)**
Map 1	RRT	4.60	77.8888	73.8348	0.946	4.20
Map 2	RRT	4.59	74.9542	75.8492	0.895	3.90
Map 3	RRT	4.50	73.2089	73.9849	0.776	3.93
Map 4	RRT	4.61	78.0373	78.9843	0.947	3.60
Map 5	RRT	4.04	74.7899	75.6749	0.885	3.23
Map 6	ACO	3.09	6.0549	6.1469	0.092	3.32
Map 7	ACO	3.09	6.5576	6.556	0.098	3.18
Map 8	ACO	3.07	6.0982	6.1872	0.089	3.28
Map 9	ACO	3.05	5.7798	5.8728	0.093	6.65
			AWS Cloud Computing Platform			
Ref. Map	**Algo. Used**	**Sending Time**	**Time Taken by Agents**	**Expected Time taken**	**Time Difference**	**Network Delay (%)**
Map 1	A*	4.02	65.525	64.000	1.525	2.83
Map 2	A*	4.03	160.153	156.00	4.153	2.59
Map 3	A*	5.978	512.243	506.00	6.248	1.22
Map 4	A*	3.630	47.381	46.00	1.381	2.91
Map 5	A*	4.046	76.625	75.00	1.625	2.12
Map 6	A*	3.610	443.049	440.00	3.049	0.69
Map 7	A*	3.792	51.516	50.00	1.516	2.94
Map 8	A*	3.973	125.540	124.00	1.540	1.23
Map 9	A*	3.973	791.831	788.00	3.831	0.48



(**a**) Graph showing computation time and relative delay of the reference maps



(**b**) Graph showing relative delay performance of the dense, moderate, and sparse maps using AWS

Figure 8. Output of path planning using A* Star algorithms with a local machine (**a**) and Google Cloud Computing Platform (**b**).

In this work, we focused on using cloud computing platforms for robotic applications. The main advantage revealed in this study is that when the number of services increases with respect to any applications, traditional applications start to deteriorate in terms of performance, whereas cloud computing platforms show steady performance in terms of parameters. This is the main advantage over traditional approaches that we would like to highlight. The benefit of using a cloud computing platform adds an extra advantage to the applications, as it does not limit storage and computational capacity.

6. Conclusions

The evolution of networking and cloud-based services is only going to improve from the current state. That has some very real and plausible implications for the future of computing. This study aimed to capitalize on those very implications. We proposed the use of Google Cloud and AWS cloud computing platforms for robot path planning. RRT, PFA, ACO, and A* algorithms were implemented to test the proposed approach. The time taken by the cloud server was monitored and compared with the time taken by the conventional approach. The result obtained using the proposed approach showed improvement in performance. The experiment was conducted on different reference maps, and a significant improvement was observed. The time taken by the local machine was compared with the time taken by the cloud plus the time taken while transferring the data between the cloud server and the robot. This addition created a slight overhead, but as the scale of work increased, this overhead gradually decreased. This behavior shows that using the cloud for robotic applications may be more useful in dense and complex environments. There are several other benefits of using cloud computing platforms: Multiple algorithms can be implemented for different reference maps, which can be used instantly as and when required by applying any Intelligent techniques. It can also be concluded that the cloud instance performs faster than the local machine. Additionally, it serves on-demand, it is self-service, with broad network access, resource pooling, and rapid elasticity, and it provides measured service.

The future course of action of this research would be to test the path-planning applications of the robot with some more parameters and complex configuration spaces. We will also try to explore the Multi-robot coordination scheme on some applications using Google Cloud.

Author Contributions: Validation, S.K.P.; Formal analysis, R.D.; Investigation, K.U.S.; Resources, T.S. and A.K.; Data curation, J.K.S. and T.A.; Writing—original draft, K.S. and S.S.; Funding acquisition, T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not Applicable.

Conflicts of Interest: All authors declare there are no conflict of interest

References

1. Abu-Amara, F.; Bensefia, A.; Mohammad, H.; Tamimi, H. Robot and virtual reality-based intervention in autism: A comprehensive review. *Int. J. Inf. Technol.* **2021**, *13*, 1879–1891. [CrossRef]
2. Lam, M.L.; Lam, K.Y. Path planning as a service PPaaS: Cloud-based robotic path planning. In Proceedings of the 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014), Bali, Indonesia, 5–10 December 2014; pp. 1839–1844.
3. Dawarka, V.; Bekaroo, G. Cloud robotics platforms: Review and comparative analysis. In Proceedings of the 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), Mon Tresor, Mauritius, 6–7 December 2018; pp. 1–6.
4. Liu, J.; Zhou, F.; Yin, L.; Wang, Y. A Novel Cloud Platform for Service Robots. *IEEE Access* **2019**, *7*, 182951–182961. [CrossRef]
5. Camargo-Forero, L.; Royo, P.; Prats, X. Towards high performance robotic computing. *Robot. Auton. Syst.* **2018**, *107*, 167–181. [CrossRef]
6. Waibel, M.; Beetz, M.; Civera, J.; d'Andrea, R.; Elfring, J.; Galvez-Lopez, D.; Häussermann, K.; Janssen, R.; Montiel, J.; Perzylo, A.; et al. Roboearth. *IEEE Robot. Autom. Mag.* **2011**, *18*, 69–82. [CrossRef]
7. Alamri, A.; Ansari, W.S.; Hassan, M.M.; Hossain, M.S.; Alelaiwi, A.; Hossain, M.A. A survey on sensor-cloud: Architecture, applications, and approaches. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 917923. [CrossRef]
8. Sharma, K.; Doriya, R. Path planning for robots: An elucidating draft. *Int. J. Intell. Robot. Appl.* **2020**, *4*, 294–307. [CrossRef]
9. Hu, L.; Jiang, Y.; Wang, F.; Hwang, K.; Hossain, M.S.; Muhammad, G. Follow me Robot-Mind: Cloud brain based personalized robot service with migration. *Future Gener. Comput. Syst.* **2020**, *107*, 324–332. [CrossRef]
10. Mohamed, K.; Elshenawy, A.; Harb, H. Comparison of Traditional and Cloud-based models for Multi-robot Exploration and Fire Searching. In Proceedings of the 1'st International Conference on Information Technology (IEEE/ITMUSTCONF), Jeju-si, Republic of Korea, 16–18 October 2019.
11. Wu, H.; Wu, X.; Ma, Q.; Tian, G. Cloud robot: Semantic map building for intelligent service task. *Appl. Intell.* **2019**, *49*, 319–334. [CrossRef]

12. Turnbull, L.; Samanta, B. Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure. In Proceedings of the 2013 Proceedings of IEEE Southeastcon, Jacksonville, FL, USA, 4–7 April 2013; pp. 1–4.
13. Lu, D.; Li, Z.; Huang, D.; Lu, X.; Deng, Y.; Chowdhary, A.; Li, B. VC-bots: A vehicular cloud computing testbed with mobile robots. In Proceedings of the First International Workshop on Internet of Vehicles and Vehicles of Internet, Paderborn, Germany, 4–8 July 2016; pp. 31–36.
14. Du, Z.; He, L.; Chen, Y.; Xiao, Y.; Gao, P.; Wang, T. Robot cloud: Bridging the power of robotics and cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 337–348. [CrossRef]
15. Liu, B.; Wang, L.; Liu, M. Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4555–4562. [CrossRef]
16. Chinchali, S.; Sharma, A.; Harrison, J.; Elhafsi, A.; Kang, D.; Pergament, E.; Cidon, E.; Katti, S.; Pavone, M. Network offloading policies for cloud robotics: A learning-based approach. *arXiv* **2019**, arXiv:1902.05703.
17. Crow, S. Google Cloud Robotics Platform coming to developers in 2019. *Robot Rep.* **2018**, *1*, 01–04.
18. Sparrow, R. Robots and respect: Assessing the case against autonomous weapon systems. *Ethics Int. Aff.* **2016**, *30*, 93–116. [CrossRef]
19. Koubâa, A. Service-Oriented Computing in Robotic. In *Encyclopedia of Robotics*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–12.
20. Huang, M.H.; Rust, R.T. Engaged to a Robot? The Role of AI in Service. *J. Serv. Res.* **2020**, *1* 1094670520902266. [CrossRef]
21. Saini, M.; Sharma, K.; Doriya, R. An empirical analysis of cloud based robotics: Challenges and applications. *Int. J. Inf. Technol.* **2022**, *14*, 801–810. [CrossRef]
22. Wan, J.; Tang, S.; Yan, H.; Li, D.; Wang, S.; Vasilakos, A.V. Cloud robotics: Current status and open issues. *IEEE Access* **2016**, *4*, 2797–2807. [CrossRef]
23. Saravanan, K. Cloud robotics: Robot rides on the cloud–architecture, applications, and challenges. In *Robotic Systems: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2020; pp. 2027–2040.
24. Song, K.T.; Sun, Y.X. Coordinating multiple mobile robots for obstacle avoidance using cloud computing. *Asian J. Control* **2021**, *23*, 1225–1236. [CrossRef]
25. Song, K.T.; Chiu, Y.H.; Kang, L.R.; Song, S.H.; Yang, C.A.; Lu, P.C.; Ou, S.Q. Navigation control design of a mobile robot by integrating obstacle avoidance and LiDAR SLAM. In Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018; pp. 1833–1838.
26. Hurbungs, V.; Bassoo, V.; Fowdur, T. Fog and edge computing: Concepts, tools and focus areas. *Int. J. Inf. Technol.* **2021**, *13*, 511–522. [CrossRef]
27. Chen, X.; Kong, Y.; Fang, X.; Wu, Q. A fast two-stage ACO algorithm for robotic path planning. *Neural Comput. Appl.* **2013**, *22*, 313–319. [CrossRef]
28. Rashid, R.; Perumal, N.; Elamvazuthi, I.; Tageldeen, M.K.; Khan, M.A.; Parasuraman, S. Mobile robot path planning using Ant Colony Optimization. In Proceedings of the 2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), Ipoh, Malaysia, 25–27 September 2016; pp. 1-6.
29. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), Philadelphia, PA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
30. Wei, K.; Ren, B. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* **2018**, *18*, 571. [CrossRef]
31. Hwang, Y.K.; Ahuja, N. A potential field approach to path planning. *IEEE Trans. Robot. Autom.* **1992**, *8*, 23–32. [CrossRef]
32. Singh, A. Security concerns and countermeasures in cloud computing: A qualitative analysis. *Int. J. Inf. Technol.* **2019**, *11*, 683–690.
33. Choi, Y.; Choi, M.; Oh, M.; Kim, S. Service robots in hotels: Understanding the service quality perceptions of human–robot interaction. *J. Hosp. Mark. Manag.* **2020**, *29*, 613–635. [CrossRef]