Pathfinder: A Fog Assisted Vision-Based System for Optimal Path Selection of Service Robots

Niloy Irtisam*, Riad Ahmed†, Mohammad Moniruzzaman Akash‡, Raiyaan Abdullah§, Sujan Sarker¶, Sejuti Rahman‖, Lafifa Jamal**
Department of Robotics and Mechatronics Engineering
University of Dhaka, Dhaka-1000, Bangladesh
*niloyirtisam@hotmail.com, †riadrmedu@gmail.com, ‡akashmoniruzzaman@gmail.com,
§raiyaanabdullah@gmail.com, ¶sujan@du.ac.bd, ‖sejuti@gmail.com, **lafifa@du.ac.bd

***Abstract*—Service delivery application involving robots relies on the success of the navigation process. To ensure maximum performance, the navigation process should generate optimal path avoiding collisions with dynamic obstacles. Selecting such a path by analyzing the dynamic environment condition while putting minimal overhead on robots is a challenging problem in service robot navigation. In this work, we develop a Fog assisted service robot navigation system that puts most of the computing tasks to a central Fog server. The server employs a vision-based monitoring system to locate the robots and obstacles. The optimal path is generated by analyzing the available information and the robots are instructed accordingly. We implement a test-bed system to evaluate the performance of the proposed system. The results depict that the proposed fog-based system can achieve as low as 23.81% of average distance covered, 22.05% of average service delivery time, and 22.72% of average energy consumption compared to the without fog systems.**

***Contribution*—A fog-based system for service robot navigation is proposed that uses computer vision to generate optimal obstacle free path.**

***Index Terms*—Service Robot, Path Selection, Path Planning, Fog Robotics, Robot Vision, Internet of Things**

I. Introduction

Proliferation of mechanical, electronic and computing technologies has resulted in the development of multi-functional robots which are being widely used in various sectors such as manufacturing and service industries [1]. Computer vision and artificial intelligence have enabled these robots to automatically sense the environment, generate an optimal plan and gain experience from their previous actions. As a result the demand for service robots are increasing day by day [2]. In most cases, multiple robots are used to make their usage feasible. These robots might work independently or work together as a swarm of robots to achieve a predefined goal. In case of service robots, optimization of the navigation process is an important consideration. The level of automation and the performance of the system depends highly on the navigation process. For service robot navigation, one of the crucial problems is to find the optimal path for a given source and destination pair. The problem is further intensified due to the dynamic nature of the environment with multiple static and moving obstacles. To cope with this problem, service robots are required to have important capabilities such as optimal path planning, object detection as well as an awareness of the surroundings [3]. These capabilities facilitate the robots to navigate using the optimal path while avoiding collisions in order to ensure optimal performance.

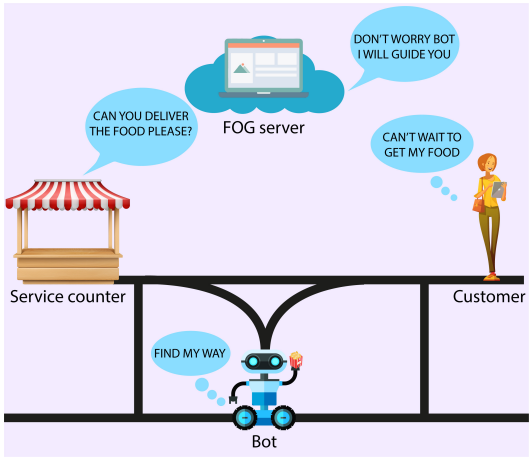


Fig. 1. A fog assisted system for service robots

Equipped with high quality camera sensors and through the use of AI based algorithms [4], today's service robots are capable of sensing obstacles and updating their plan [5]. However, this introduces computational overhead of the robots, and their battery power drains rapidly as they are required to drive high energy consuming sensors. Moreover, in case of any fault event, for example malfunctioning of the camera sensors, the performance of the robot falls drastically. Fog robotics [6], [7] involves offloading computing tasks to a central server, known as Fog Server, from the robots to mitigate this problem. The server is computationally more powerful than the robots, which are mostly of low intelligence. Most computations take place in the server and simple instructions are provided to the robots. As all robots are in communication with the server, information between robots can be easily exchanged without the need for additional sensors or communication systems in-between individual robots. A single intelligent server, controlling multiple robots of low intelligence robots, is also more cost effective than developing multiple powerful bots with high intelligence. Moreover, as all computations are being

978-1-7281-9331-1/20/$31.00 ©2020 IEEE

performed on a single server, this simplifies the networking requirements and decreases computational complexity [8].

Robot navigation in dynamic environment requires continuous monitoring of the surroundings of the robots. A computationally powerful Fog server can facilitate this monitoring system by extracting useful information such as the position of the robots, obstacles and anomalies in the path using a robust vision based system. Such a system is most effective in known environments like food courts, shopping malls, libraries, etc., where the structure of the environment is known and service robots are required to continuously move from one service point to another in order to perform a specific set of tasks. Being motivated by the above analysis, we propose a fog assisted navigation system named *Pathfinder* which uses computer vision-based algorithms to monitor the environment conditions and finds the optimal path to ensure obstacle-free navigation for the service robots. The *Pathfinder* system is analogous to a manager-worker hierarchy where a manager takes decisions and gives orders while the workers perform as instructed. The main contribution of this paper is itemized as follows:

- A fog assisted navigation system is developed for service robots where most of the computational and monitoring tasks are performed by the server.
- The environment is represented using a graph data structure with a technique to incorporate obstacles along the navigation path and the optimal path selection problem is formulated as a 0-1 Integer Linear Programming (ILP) problem.
- The optimal navigation path for the service robot is determined using Dijkstra and an improved A* algorithm.
- A testbed is implemented and the performances of the proposed system is analyzed.

The rest of the paper is organized as follows. In Section II, we explore related research in this field. Subsequently, we present a system model and assumptions for the proposed system in Section III and the functionalities of the proposed system is also presented in detail, followed by the performance evaluation in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

There have been numerous efforts to provide insight into the planning of autonomous mobile robots in different types of environments. A comparison among various mobile robot path planning strategies is done in [9]. The strategies have been categorised into Classical and Heuristic Methods which are further sub-categorised into Analytical, Enumerative, Evolutionary and Meta-Heuristic Methods. The Analytical Methods are the most complicated and are not suitable for practical usage. Enumerative Methods require a huge search space whereas the Evolutionary Methods do not perform well when the search space is too large. The Meta-heuristic methods overcome these problems while providing remarkable results. In [10], authors proposed a Multi-Sphere Scheme (MSS) based on both combinatorial and gradient-based optimization techniques considering an environment with fixed obstacles. Here, Nonlinear Programming Optimization Solver was used in tandem with MSS to resolve the intersections of two object. However, in case of dynamic obstacles, this method did not work as intended. Authors in [11] consider four problems of static obstacles and provide a mathematical analysis. The problems are: local minima trap situations, closely spaced obstacles having no paths between them, obstacles creating oscillations and oscillation in tapered paths. However, they do not consider dynamic obstacles that may appear and disappear from the environment. In [12], robot's kinematic and kinodynamic constraints are considered for time-dependent obstacle avoidance. Compared to conventional local planning, the approach yields smooth transitions and shorter velocity profiles. Although a cloud server has been used, most real time path planning takes place in the mobile robots. The robots are equipped with sensors in order to perceive its surroundings. The system thus have a partial map of the environment, i.e., the local environment of the mobile robots. Without a complete map of the environment, it is difficult to generate the optimal path when the environment is dynamic with multiple obstacles. Inspired by the parasitic nature of the cuckoo authors in [13] used cuckoo optimization algorithm to optimize the distances by the upper and lower bound of each reference point. It involves moving in a straight line towards the goal while avoiding obstacles. However, for crowded environments, the algorithm does not guarantee a complete collision free path. To find the optimal path between two locations in any indoor terrain map, a multi-layer dictionary based representation of the map is proposed in [14] to reduce the storage space required for the Dijkstra algorithm. Authors also introduced the degree of difficulty for the junctions to further improve the output. In [15], an improved A* algorithm named Weight Convergence Accelerated A* (WCAA*) is proposed for local path planning of service robots where the weight for heuristics is introduced. The proposed algorithm ensures real-time performance requirements of the service robots by significantly reducing the searching time. However, none of the above works considers the appearance of dynamic obstacles while the service robot is navigating along the path.

Although a number of works have been done in optimal path selection for service robot navigation, most of them work with the partial view of the environment and thus they can not handle the dynamic obstacle objects that appear in the navigation path. Moreover, they put most of the computational overhead on the robots. The aforementioned observations led us to develop a central Fog based system which assists the service robots in selecting optimal path for serving requests while coping with dynamic environment conditions.

III. Proposed System

In this section, we first discuss the entities, different functional modules and workflow of the proposed system in detail. Later we unfold the functionality of the different modules, formulate the obstacle free optimal path selection problem and provide the solution of the problem.

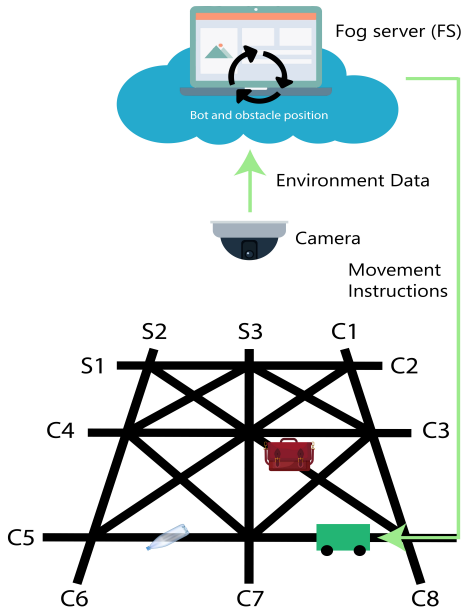


Fig. 2. The system model

A. Entities

The system model is shown in Fig. 2. It consists of a central fog server (FS) which controls a service robot (SR). We consider a service environment where there are n customers $(C_1, C_2, C_3, ..., C_n)$ and m service points $(S_1, S_2, S_3, ..., S_m)$. Here, $C_i (i = 1, 2, 3, ..., n)$ and $S_j (j = 1, 2, 3, ..., m)$ denotes the location of a customer and a service point, respectively. An overhead camera connected to the server continuously monitors the environment i.e the location of the SR and external objects. Each customer C_i sends request for a service to the FS and based on this request the FS generates an optimal path analyzing the environment dynamics. The FS then generates the navigation instructions and sends them to the SR for serving the request. Here we assume that the service provided by the system is only limited to delivery services and a SR can serve only one request at a time. Movement of the SR is also restricted to a predefined navigation track and we consider dynamically appearing objects.

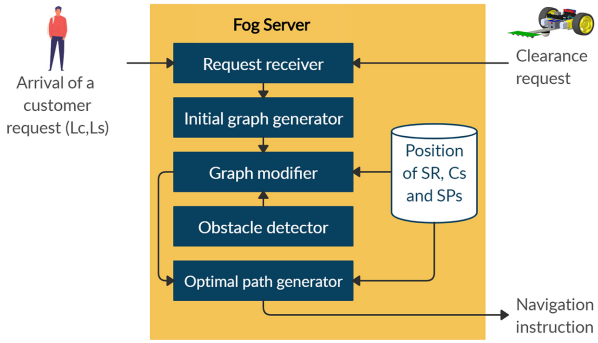


Fig. 3. Workflow of the proposed system

B. Workflow of the proposed system

Fig. 3 depicts the workflow of the proposed model. A customer posts a service request, (C_i, S_j) to the service receiver module of the FS via the user interface. The FS initially generates a graph of the navigation track and stores it in the back-end database. The database also keeps track of the current location of the SR, location of the customers and service points. The dynamic environment includes obstacles appearing in the path of the SR. The obstacle detector module detects the objects which obstruct the path using the vision system. The graph modifier module then modifies the graph accordingly. The optimal path generator module generates an optimal path for the SR based on the its current location and environment information.

C. Communication Protocol

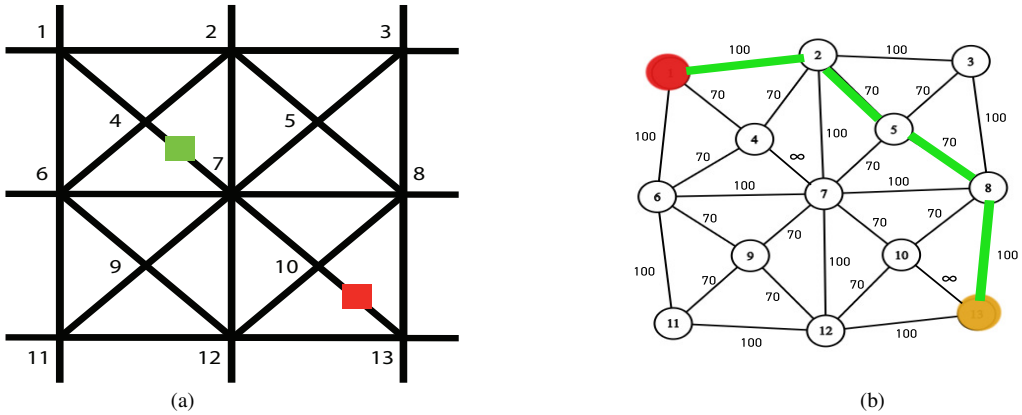
Communication between the service robot and the Fog server takes place via WiFi. We consider a simple communication protocol between the SR and the FS where, after determining the optimal path, the server sends the movement instructions to the SR. The FS sends movement instructions to the SR to move onto the next node from its current node. The FS continuously checks for the obstacles along the path, updates the graph and generates the optimal path accordingly. Whenever the SR reaches a node it asks for clearance from the server. The FS then checks for any changes in the previously generated path. If the path is unchanged, the server sends the next movement instruction for the current path. Otherwise, the updated optimal path is communicated with the SR. This process is repeated until the SR reaches its destination. The SR only understands and executes simple instructions such as ”go forward”, ”turn left”, ”rotate”, etc., received from the server.

D. Graph Generation from the Environment Image

At first the environment pathway is converted into a graph data structure $G(V, E)$, where V denotes the set of nodes and E denotes the set of edges. From the image of the navigation track, the points of intersections of the lines as well as the end points of the lines are detected, which are defined as the nodes of the graph. K-means clustering is used in order to determine the centre point of each node [16]. Afterwards the connectivity between the nodes i and j is determined and the weight w_{ij} is assigned as the euclidean distance between the centre of corresponding nodes. Thus the image to graph generator module converts the navigation track into a weighted graph which contains all possible paths in the environment and serves as an input to the path selection problem.

E. Obstacle Detection

When a clearance request arrives at the FS, it determines the coordinates of the SR from the video feed data and its position is mapped to the corresponding node of the graph. The position of external obstacles are also determined using algorithms proposed in [17] and their coordinates are used to determine which pair of nodes the obstacles lie between. After detecting all of the obstacles (if any), the FS constructs a set $E' \subseteq E$ of edges that contains obstacles.



(a) (b)

Fig. 4. An illustrative example of optimal path selection method for the SR: (a) Given navigation track (Node 1 and 13 are the source and destination, respectively) having two objects obstructing the paths $(4, 7)$ and $(10, 13)$ (b) Generated graph: $W_{4,7} = \infty$ and $W_{10,13} = \infty$ due to the presence of obstacles, and other edge weights are set as the Euclidean distance between corresponding nodes. The resulted optimal path: $1 \rightarrow 2 \rightarrow 5 \rightarrow 8 \rightarrow 13$

F. Problem formulation

The optimal path selection for the SR from a service point s to a target customer location t can be formulated as a 0-1 integer linear programming (ILP) problem as follows:

$$minimize: \sum_{(i,j)\in E} w_{ij} \times x_{ij} \tag{1}$$

such that

$$x_{ij} \in \{0, 1\} \tag{2}$$

$$\forall i, \sum_{j} w_{ij} - \sum_{j} w_{ji} = \begin{cases} 1, & if \quad i = s \\ -1, & if \quad i = t \\ 0, & otherwise \end{cases} \tag{3}$$

$$(i, j) \notin E' \tag{4}$$

Given a weight vector, w_{ij}, equation 1 finds the shortest obstacle free path from s to t for the SR. Here, x_{ij} is a binary variable whose value is 1 if edge (i, j) is in the shortest path and otherwise 0.

Constraint 3 ensures that all the vertices excepts s and t must have the same number of incoming and outgoing edges to be included in the path from s to t. Constraint 4 is the obstacle avoidance constraint which indicates that any edge (i,j) containing obstacles can not be a part of the shortest path.

G. Optimal Path Selection

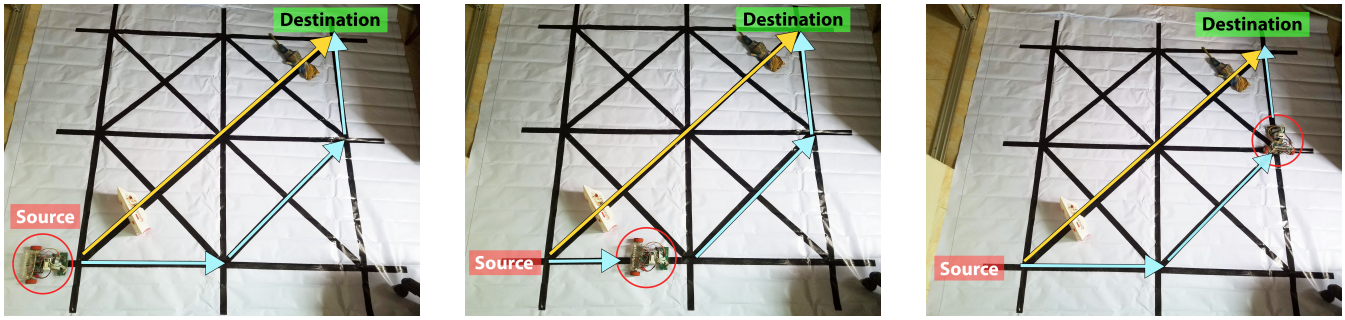
The optimal path selection problem of equation 1 is a 0-1 ILP problem which can be solved by the FS using an available ILP solver. However, in this work the Dijkstra algorithm was used on the pre-processed graph as it is a popular algorithm for finding the shortest path from a source node to all other nodes. In pre-processing phase, the weight of the edges were updated as $w_{ij} = \infty$ if $(i, j) \in E'$ in order to account for obstacles. In short, the Dijkstra algorithm starts with a source node and its neighbour with the shortest distance is selected and visited. Afterwards, the neighbour with the shortest distance starting from the initial node i.e., the shortest total distance starting from the initial node to the neighbouring node via the second node is picked and that neighbouring node is then visited. This process is repeated and in this way the shortest path from the initial node to every other node in the graph is determined. Here, we use Dijkstra algorithm with fibonacci heap whose computational complexity is $O(E + V \times logV)$ [18]. We consider a typical service environment (e.g., a restaurant floor) for which the generated graph has moderate number of nodes thus Dijkstra algorithm finds the shortest path with acceptable computational delay. However, while handling a large scale and complex environment we may end up with a large graph for which the Dijkstra algorithm suffers from higher execution delay. To reduce this delay for the large graphs we use weight convergence accelerated A* (WCAA*) algorithm [15] which is an improved version of A* algorithm. Fig. 4 illustrates the generated optimal path using the proposed approach for a given example scenario.

IV. Performance Evaluation

In this section, we develop a fog assisted and vision based simulation test bed to analyze the performance of the proposed system. We implement the proposed fog version of two algorithms: Dijkstra and Weight Convergence Accelerated A* (WCAA*) and compare their performance with two without-fog approaches: without-fog (Dijkstra) [14] and without-fog (A*) [15]. In both of the without-fog approaches, the SR itself senses the environment, detects obstacles and calculates the shortest path. For the WCAA* algorithm the value of the weight of the heuristics is 2 [15].

A. Simulation Environment

In order to simulate the system, a 7 $\times$ 7 feet gridmap similar to fig. 4 with intersecting straight lines and diagonal lines is used to replicate a real world scenario. The black lines denote the paths along which the SR will travel. The points of intersection of the lines and the end points of the lines denote the destination nodes. A Xiaomi Redmi Note 3 with a 16MP camera was used as the overhead camera.The overhead camera was mounted 8 feet above the centre of



(a) (b) (c)

Fig. 5. An illustrative example of testbed simulation (The yellow line denotes the shortest path which is not usable due to the presence of obstacles. The blue lines denote the path followed by the SR): (a) The SR gets the obstacle free shortest path calculated by the FS (b) The SR reaches the next node following the obstacle free path (c) As the environment is unchanged, the previous path is followed

the area. For larger environments, multiple cameras might be stitched together to obtain a full view of the environment. The camera communicated with the server via WiFi. A TP-Link Archer C50 WiFi router is used to create the WiFi network. All computations take place on the Fog server which has been built on the python Django framework. The server ran on a laptop (Intel Core i5 6300HQ processor, 8gb DDR3L RAM and 256gb SSD storage). Communication between the SR and the FS takes place via WiFi. A user interface was developed for the front-end of the server where a user can input the request. A simple service robot is constructed which uses a differential drive system consisting of two 12V 300rpm DC motors. L293D motor driver is used for operating the motors. NodeMCU is used as the microcontroller due to its WiFi capabilities and an 1100mAh battery is used to power the SR.

B. Simulation Parameters

We analyze the performance of the proposed system on the following metrics:

- Average distance travelled: the average distance travelled by the SR to serve a request.
- Average service time: the average time required to serve a request.
- Average energy consumption: the amount of energy consumed by the SR to serve a request.

C. Generated optimal path

Fig. 5 demonstrates the simulated system in operation. The optimal path in absence of any obstacles is denoted by the yellow line. However due to the presence of obstacles, the bot is unable to travel along that path. The object detection algorithm detects the presence of the obstacles and a new optimal path has been generated which is shown in blue. The bot travels along this path avoiding the obstacles.

D. Impact on average distance travelled

The impact of varying service request arrival rates on the average distance travelled by a SR to serve a request is shown in Fig. 6. From the graph it is clear that the average distance increases with the increasing request arrival rates. However, the SR in the proposed system can serve requests by covering 23.81% less distance on average. This can be accounted to the fact that the Fog server has an overhead vision system that continuously monitors the entire environment. In case of any obstacles appearing in the path, the system automatically updates the optimal path in order to minimize disruptions. On the other hand, the system without the FS cannot monitor the entire environment and its vision is only limited to what is in front of it. If any obstacle appears in its path, it has no way of knowing until it goes in front of the obstacle. Afterwards it is required to turn back and generate a new path from there on, which might also contain unknown obstacles. Note that, as the graph generated from the navigation track has fewer number of nodes, Dijkstra and WCAA* provide almost same performance in both of the fog and without fog versions. However, the Fog based versions always outperforms the without fog versions.

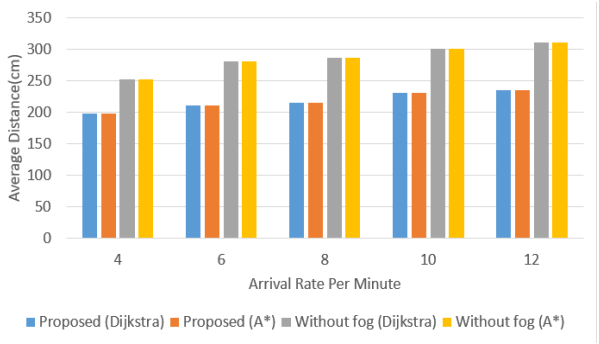


Fig. 6. Avg. distance travelled for serving requests

E. Impact on average service time

The impact of varying service request arrival rates on the average service time is shown in Fig 7. For all the studied systems, the average time increases as the service rate increases. However, our system was found to require 22.05% less time compared to the other two systems. As seen in section IV-D, the proposed system travels less distance on average to serve a request. Owing to this, requests can be served in a shorter

time allowing more requests to be completed in a limited time window.

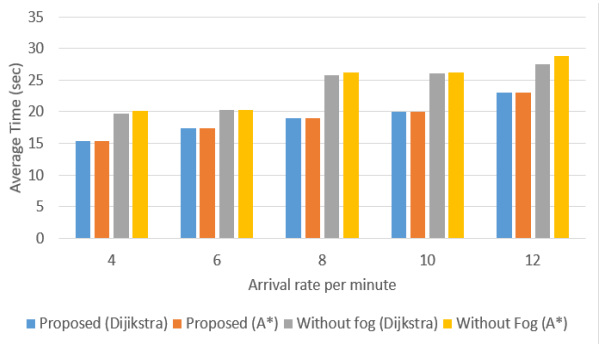


Fig. 7. Time required for serving requests

F. Impact on average energy consumption

The impact of varying service request arrival rates on the average energy consumed to serve a request is shown in Fig. 8. Energy consumption is an important economic factor as lower energy consumption ensures lower operational costs. For all four systems, with the increase in service rate it is observed that on average more energy was consumed per request. However, our system consumes 22.72% less energy on average compared to the without-fog system. As seen previously, the proposed system requires less distance and time on average to serve requests. This means that the motors and other components have to operate less compared to the other two systems. This results in less energy consumption.

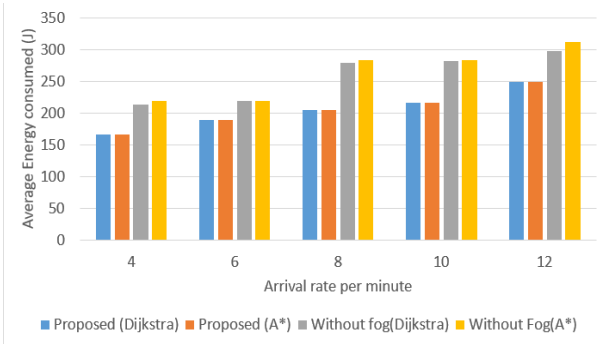


Fig. 8. Energy usage for serving requests

V. Conclusion

A vision based system for service robot navigation is proposed in this paper. The proposed system uses the concept of Fog robotics to offload computational tasks to a server in order to generate optimal paths to maximize efficiency while putting minimal overhead to the service robots. The optimal path selection problem is formulated as a 0-1 integer linear programming problem and the optimal path is calculated using two different fog based approaches and tested on a real test-bed. The results of test-bed simulation depicts that a fog assisted system can achieve satisfactory performance. The system can be further improved to generate optimal paths in environments where pathways are not specifically defined. Generating an optimal path set for multiple service robots while avoiding collision among themselves will also be an interesting future work. Performance of the proposed system on large scale environment and varying server conditions will also be evaluated.

References

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. Beck, M. Haage, M. Hansen, M. Roa, O. Sornmo, S. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kessler, and M. Danzer, "Smerobotics: Smart robots for flexible manufacturing," *IEEE Robotics & Automation Magazine*, vol. 26, pp. 78–90, 03 2019.

[2] J. Wirtz, P. Patterson, W. KUNZ, T. Gruber, V. Lu, S. Paluch, and A. Martins, "Brave new world: Service robots in the frontline," *Journal of Service Management*, vol. 29, 11 2018.

[3] S. Nurmaini and B. Tutuko, "Intelligent robotics navigation system: Problems, methods, and algorithm," *International Journal of Electrical and Computer Engineering*, vol. 7, 12 2017.

[4] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. M. Basalamah, "A survey of shortest-path algorithms," *ArXiv*, vol. abs/1705.02044, 2017.

[5] Yanrong Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, April 2004, pp. 4350–4355 Vol.5.

[6] S. L. Krishna Chand Gudi, S. Ojha, B. Johnston, J. Clark, and M. Williams, "Fog robotics for efficient, fluent and robust human-robot interaction," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Nov 2018, pp. 1–5.

[7] N. Tian, A. K. Tanwani, J. Chen, M. Ma, R. Zhang, B. Huang, K. Goldberg, and S. Sojoudi, "A fog robotic system for dynamic visual servoing," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 1982–1988.

[8] S. Sarker, M. A. Razzaque, M. M. Hassan, A. Almogren, G. Fortino, and M. Zhou, "Optimal selection of crowdsourcing workers balancing their utilities and platform profit," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8602–8614, 2019.

[9] M. N. Zafar and J. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia Computer Science*, vol. 133, pp. 141 – 152, 2018.

[10] A. Shurbevski, N. Hirosue, and H. Nagamochi, *Optimization Techniques for Robot Path Planning*, 01 2014, vol. 231, pp. 111–120.

[11] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, Oct 2000.

[12] F. G. Lopez, J. Abbenseth, C. Henkel, and S. Dörr, "A predictive online path planning and optimization approach for cooperative mobile service robot navigation in industrial applications," in *2017 European Conference on Mobile Robots (ECMR)*, Sep. 2017, pp. 1–6.

[13] S. Hosseininejad and C. Dadkhah, "Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839575, 2019.

[14] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," in *Proceedings of the 2015 2nd International Conference on Information Science and Security (ICISS)*, ser. ICISS '15. USA: IEEE Computer Society, 2015, p. 1–4.

[15] J. Cao and J. Liu, "Path planning in service robot based on improved a* algorithm," *IOP Conference Series: Earth and Environmental Science*, vol. 170, p. 042110, 07 2018.

[16] Y. Li and H. Wu, "A clustering method based on k-means algorithm," *Physics Procedia*, vol. 25, pp. 1104–1109, 12 2012.

[17] M. Chandrajit, G. R, and V. T, "Multiple objects tracking in surveillance video using color and hu moments," *Signal & Image Processing : An International Journal*, vol. 7, pp. 15–27, 06 2016.

[18] J. Sneyers, T. Schrijvers, and B. Demoen, "Dijkstra's algorithm with fibonacci heaps: An executable description in chr." in *20th Workshop on Logic Programming*, 01 2006, pp. 182–191.