Computers and Electrical Engineering 74 (2019) 1–21

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

New approach to enhancing the performance of cloud-based vision system of mobile robots☆

Mahmoud Badawy[a,*], Hisham Khalifa[a], Hesham Arafat[b]

[a] Computers Engineering and Control systems Department Faculty of Engineering Mansoura University, Egypt
[b] Mechanical Design and Production and graduated from a higher studies diploma in Automatic control - Faculty of Engineering Mansoura University, Egypt

ARTICLE INFO

Article history:
Received 4 March 2018
Revised 15 December 2018
Accepted 3 January 2019

Keywords:
3D point cloud
Cloud computing
Cloud robotics
Computer vision
Computation offloading
Mobile robot
Real-time networking
Stereo vision

ABSTRACT

Mobile robots require real-time performance, high computation power, and a shared computing environment. Although cloud computing offers computation power, it may adversely affect real-time performance owing to network lag. The main objective of this study is to allow a mobile robot vision system to reliably achieve real-time constraints using cloud computing. A human cloud mobile robot architecture is proposed as well as a data flow mechanism organized on both the mobile robot and the cloud server sides. Two algorithms are proposed: (i) A real-time image clustering algorithm, applied on the mobile robot side, and (ii) A modified growing neural gas algorithm, applied on the cloud server side. The experimental results demonstrate that there is a 25% to 45% enhancement in the total response time, depending on the communication bandwidth and image resolution. Moreover, better performance in terms of data size, path planning time, and accuracy is demonstrated over other state-of-the-art techniques.

1. Introduction

Mobile robots affect everyday life, as they can replace humans in several activities, such as material handling, building construction and demolition, or even planting and harvesting.

Mobile robots are divided into two main categories: (i) Teleoperated mobile robots and (ii) Autonomous mobile robots. Teleoperated robots can be controlled through a wireless or wired communication system, e.g., Internet, radio connection, direct cable, or satellite. Autonomous robots can perform tasks such as house cleaning, planting and harvesting, searching and rescuing, and constructing and demolishing with minimal user intervention. Therefore, to fully use their capabilities, mobile robots should be controlled with the highest degree of autonomy. To this end, various architectures have been investigated. Robotic vision has played a major role in making mobile robot navigation safer, but its computational complexity limits autonomy. One solution is to provide the robot with high computational power; however, this requires more hardware resources, which may increase size and cost, as well as larger capacity batteries.

The term “Cloud Robotics” was introduced by Kuffner [1] to support “Remotely Brained” robots with parallel computation and big data sharing over the internet. Cloud robotics exploits cloud technologies to gain elastic computation, storage, and

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Guanglong Du.

* Corresponding author.
E-mail address: engbadawy@mans.edu.eg (M. Badawy).

https://doi.org/10.1016/j.compeleceng.2019.01.001
0045-7906/

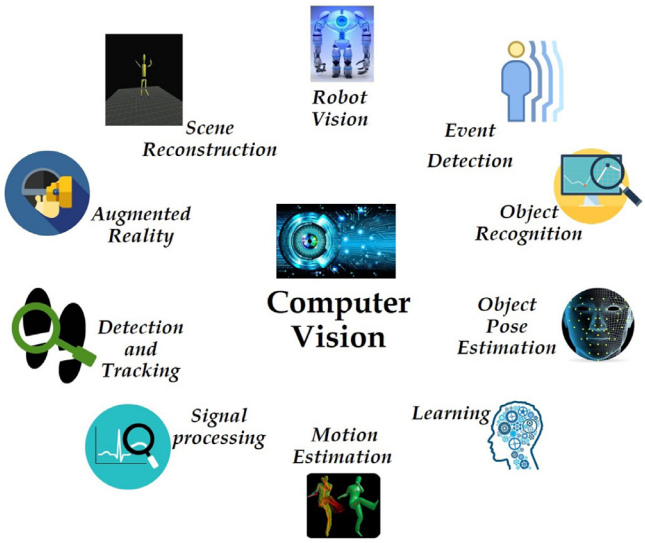


Fig. 1. Computer vision.

robotic services, and to share data from other agents (robots, machines, smart objects, or humans). Users can allocate robots to tasks through internet networks. By using cloud computing technologies, the robot's overall cost is greatly reduced. Thus, it is possible to construct lightweight, low cost, and smart robots with long-life batteries because computational load is reduced. Those robots have an intelligent "brain" in the cloud. The "brain" is provided with a knowledge base, vast information, learning models, data processing, 3D maps, and models.

Lightweight tasks use a small part of cloud processing, and thus resources are available for other purposes. In commercial clouds, users pay only for the power they use, and heavy tasks consume more cloud computing power. Therefore, this situation may be beneficial for robots that do not have high computational power. The concept of using cloud services for some tasks that, for various reasons, should not be performed by the robot itself is known as "robotic brain". This concept is not new, but it is now feasible owing to cloud technology. Scalability is a major issue in mobile robot control systems. A single user should be able to control multiple robots simultaneously, which may be physically distant from each other and the user. Cloud computing can handle different nodes as long as they have a reliable connection.

In this study, a human cloud mobile robot (HCMR) architecture is proposed to enable single or multiple users to control multiple autonomous mobile robots simultaneously. Recent research in mobile robot control has been concerned with two major problems: (i) the computational power that a mobile robot requires, and (ii) the scalability issue that prevents multiple mobile robots from being added to the system. Cloud computing technology can be used in robotic systems to solve these problems by exploiting its high computational power and scalability. Real-time networking may degrade system performance; thus, a cloud vision data flow mechanism is used. Accordingly, two algorithms are proposed: (i) a real-time image clustering algorithm (RT-IC), applied on the mobile robot side, and (ii) a modified growing neural gas (MGNG), applied on the cloud server side.

The paper is organized as follows: In Section 2, related work is reviewed. In Section 3, the main problems and the objective of this study are presented. In Section 4, the proposed HCMR architecture and algorithms are described. In Section 5, the procedure for solving the main problems is detailed, and the key algorithms are elaborated. In Section 6, the experiments are presented and the results are analyzed. In Section 7, the paper is concluded.

2. Literature review

Recently, there has been extensive research on creating a new generation of robotic systems, particularly with regard to mobile robot autonomy. A number of studies indicate that more autonomy requires more computational power to achieve complex tasks, particularly for real-time applications. Robotic vision and computer graphics are among the most complex tasks that mobile robots can perform. Computer vision uses artificial intelligence to extract information from images. As shown in Fig. 1, computer vision includes a wide range of research areas. Scene reconstruction involves computing a 3D model of the scene from two or more images. Such 3D models could range from subsets of a 3D point cloud (3DPC) to 3D surface models generated by sophisticated methods. Therefore, various algorithms have been proposed to extract multiple images into 3D models or subsets of a raw point cloud.

In computer vision, 3D reconstruction using multiple images has been extensively studied. This resulted in improved performance and quality of 3D model generation. Such 3D models may include, e.g., a real-time scene of a large city generated by thousands of images. 3D reconstruction is an automatic generation process of 3D representations of objects that are

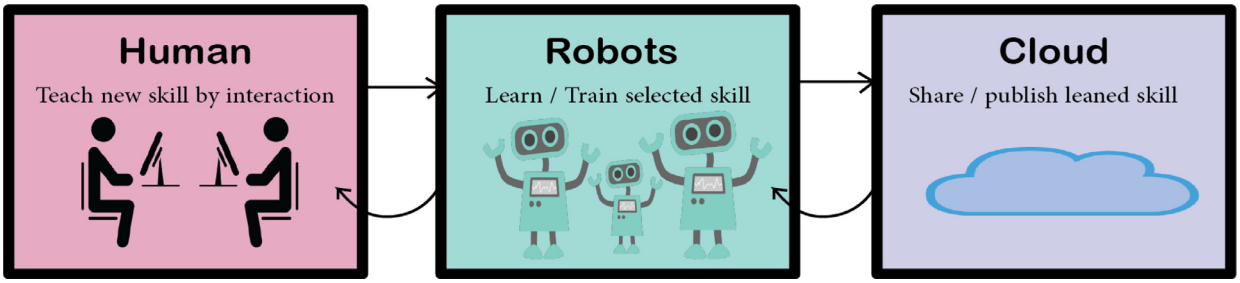


Fig. 2. Knowledge exchange and cooperation with human architecture.

difficult to model. Such models are used in graphics applications for indoor and outdoor scenes, where certain improvements are required to handle the large amounts of data and the uncontrolled environmental parameters that commonly influence outdoor scenes.

Mobile robot navigation has become safer owing to robotic vision. However, the computation complexity of robotic vision limits autonomy and reduces efficiency. One solution is to provide mobile robots with high computational power, but this would require more hardware that might increase size and cost. It would also require larger battery capacity.

Several studies on controlling mobile robots have appeared, and various systems and architectures have been proposed to allow a single user to control multiple robots. In [2], human interaction with robotic systems was studied, and controlling multi-robots by using natural language (NL) was investigated. This type of control is suitable for some applications such as service robots and automated wheelchairs for disabled people; it allows only one task to be executed at a time. Podevijn et al. [3] presented a method for controlling subgroups of swarms using vision-based arm gestures. This method is suitable for outdoor applications where mobile robots are used in wide areas such as agricultural and construction areas. Other gesture-based control methods [4] depended on the processing of neural signals generated by arm gestures. Such architectures require the user to be near the robot so that the signals can be received.

Research in mobile robots is also closely connected with research in human-robot interaction. Part of this research has led to interesting outreach activities, such as the Personal Rover Project [5], in which systems that can be used at home, school, or local science museums were developed. Other studies presented human-robot interaction techniques based on graphical user interfaces. An AJAX-based graphical user interface for a mobile robotic system with multiple sensors (an ultrasonic array, a thermal sensor, and a video streaming system) was proposed in [6] to obtain environmental information with path planning and obstacle avoidance capability. This system assumes that the environment is static.

Kakoty et al. [7] investigated mobile robot navigation and proposed a scheme, inspired by human pedestrian behavior, for robot navigation assuming unknown dynamic environment. In this method, robots constructed grid maps using onboard sonar sensors to maintain a safe direction and distance to avoid collisions with obstacles.

Ochiai et al. [8] proposed a touch panel interface to remotely control a robotic system that used simultaneous localization and mapping (SLAM) and motion planning to achieve autonomy.

A system of service robot groups that can share skills and cooperate with groups of distant human agents was proposed in [9]. Connection to the cloud was used as a knowledge repository for the robotic system. Consequently, distant groups of robots exchange learned skills with each other and adapt to cooperation situations with human agents, as shown in Fig. 2. In [10], an integration between the cyber world and the physical world was also presented to realize the idea of “Robot Cloud”, thus combining the power of robotics and cloud computing.

Ayanian et al. [11] presented a GUI-based control method, namely, an iOS application that enables the user to move a group of unmanned aerial vehicles (UAVs) through a selection box “prism” in a virtual environment. The destination coordinates are sent to the server through a wireless connection, and subsequently the server transmits them to the UAV controllers for processing. However, this architecture lacks scalability. Moreover, they introduced a motion control algorithm that is well-suited to modern tablet and smartphone interfaces, but that system also lacks scalability.

Another study proposed a method that allows multiple users to direct a group of mobile robots; however, each user can control only one robot [12]. Other studies investigated multiple robot control using user interfaces inspired by video games [13]. The proposed HCMR architecture is an extension of that by Karulf et al. [13]. That study relates to the present study in the following respects: (i) Exploiting cloud computing technology in mobile robot control systems to enhance scalability, (ii) Allowing heterogeneous mobile robots to be controlled by the user, (iii) Allowing user devices with different platforms to access the system and control mobile robots, and (iv) Using cloud computing to rapidly solve complex problems such as path planning and computer vision.

Mobile robot vision has been widely studied, and it involves several issues and challenges. Fig. 3 can be found in [14]. It shows the classification of mobile robot challenges based on the 5 W categories (why, what for, what with, how, where) [14]. In the present study, two more categories (processing method, real-time requirements) and their child elements were added to the taxonomy.

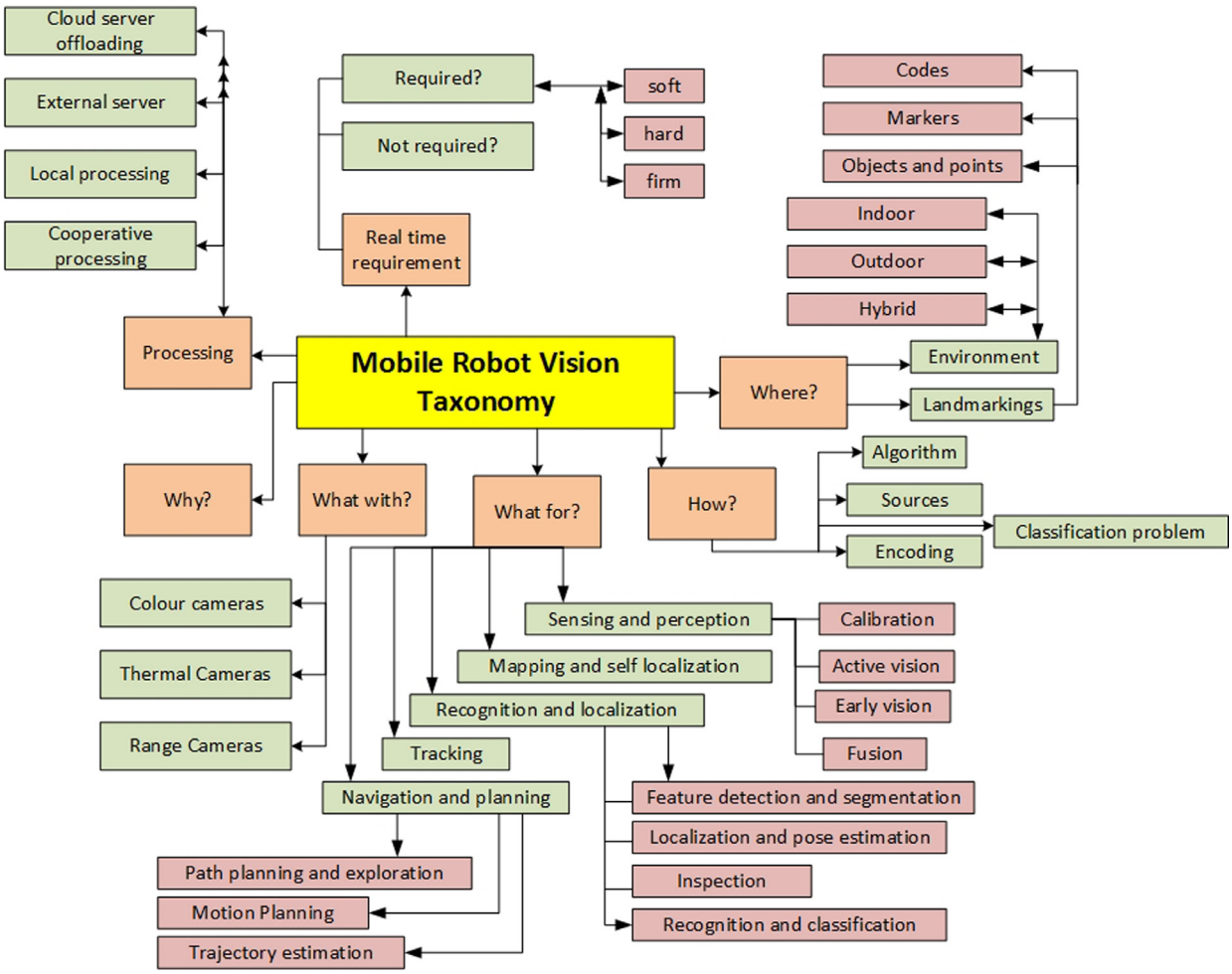


Fig. 3. Mobile robot vision taxonomy.

Recently, research on mobile robot vision has not considered the issue of real-time requirements. Object detection based on high-resolution scale-invariant feature transforms can be expedited by transmitting on-board preprocessed image information instead of raw image data to the external servers [15]; however, the cloud computing paradigm is not fully exploited. A cloud implementation of SLAM was studied in [16], where both computation offloading and collaborative work were combined to fuse information from several mobile robots. That study used a 640 × 480 pixel RGBD camera. Nimmagadda et al. proposed minimizing the total execution time of object tracking [17] through offloading decision-making for object recognition with different bandwidths, background complexities, and database sizes. In [18], a standard UDP transport protocol for transmitting large-volume images over an Ethernet network was proposed. Cloud image processing facilitated stability control, but ingenious hold action occurred in the actuation signals owing to Ethernet protocol delays. In [19], a mobile robot was used to compare the performance of on-board execution and the performance of distributed offloading.

3. Problem statement and solution plan

As seen in Section 2, numerous architectures have been proposed that can support a wide range of applications; however, most of them lack scalability and miss the issue of real-time constraints. Mobile robots are required to react under real-time constraints, which cannot be achieved by hardware with limited computational power. This issue has not been adequately addressed in the literature.

The use of cloud computing technology can ensure scalability. The proposed HCMR architecture can overcome the problem of computational power by allowing mobile robots to offload heavy computation to the cloud server. Therefore, mobile robots can start reasoning. However, another problem now arises, namely, real-time networking. Salmeron-Garcia et al. [20] applied cloud computing to construct a cloud-based 3DPC extractor for stereo images, as shown in Fig. 4. The purpose was a dynamically scalable solution that could be applied to near real-time scenarios. Their system faced several challenges,

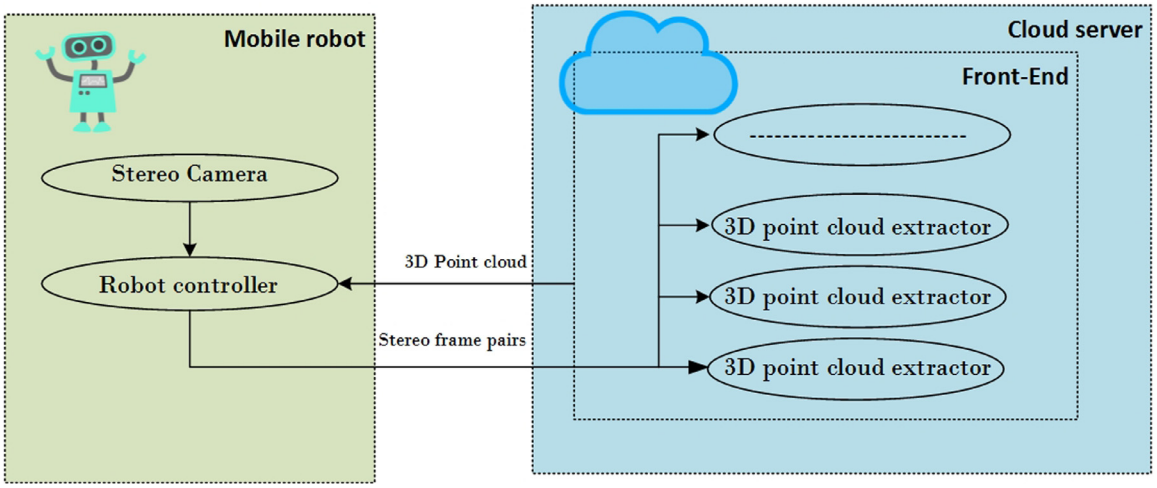


Fig. 4. Block diagram of the 3DPC extraction platform [20].

(a) Image captured by the left camera of the mobile robot

(b) Image captured by the right camera of the mobile robot

(c) 3D Map generated from the images by the stereo camera mounted on the mobile robot

Fig. 5. Various images showing a 3DPC extracted from a pair of 2D images.

such as stability, deadline conformance, and communication limitations. Fig. 5 shows a pair of images of a corridor and the 3DPC map reconstructed from them.

In the offloading algorithm presented in [20], each mobile robot decides whether to perform 3DPC extraction locally or to offload that task remotely to the cloud, according to Eqs. (1)–(3). The proposed HCMR architecture is more complicated than that in [20], as it takes into consideration that each user should be updated with the robot's physical state, surrounding environment, and progress in the allocated task. Thus, the present study considers offloading the stereo images to the cloud and letting the cloud server extract the 3DPC and other mapping data.

$$t_{local} = \frac{N_1}{IPS} \tag{1}$$

$$t_{remote} = \frac{N_1}{IPS.S} + \frac{D}{BW} \tag{2}$$

The process will be executed locally in its entirety if

$$\frac{N_1}{D} > \frac{IPS}{BW}, \tag{3}$$

where N_1, *IPS*, *D*, *BW*, and *S* denote computer instructions (inst), number of instructions per second (inst/s), amount of data sent and received (bit), network bandwidth (bit/s), and speed-up on the cloud server, respectively.

In Salmeron-Garcia et al. [20], mobile robots require high communication bandwidth so that a large amount of data may be transferred for each frame of the stereo camera. Although images with high resolution have large sizes and thus consume a correspondingly large percentage of the connection bandwidth, there is a growing need to use them for extracting 3DPCs instead of lower resolution images. High-resolution images will facilitate the reduction of 3DPC noise and enhance mapping accuracy. For each pair of images extracted to 3DPC, the mobile robot receives a large-size 3DPC. Then, the mobile robot starts reasoning. Transferring a large amount of data for each frame may reduce scalability and degrade the real-time performance of the system. Another good solution for robust visual SLAM using cloud services is Robust-Secure-Elastic PlatForm

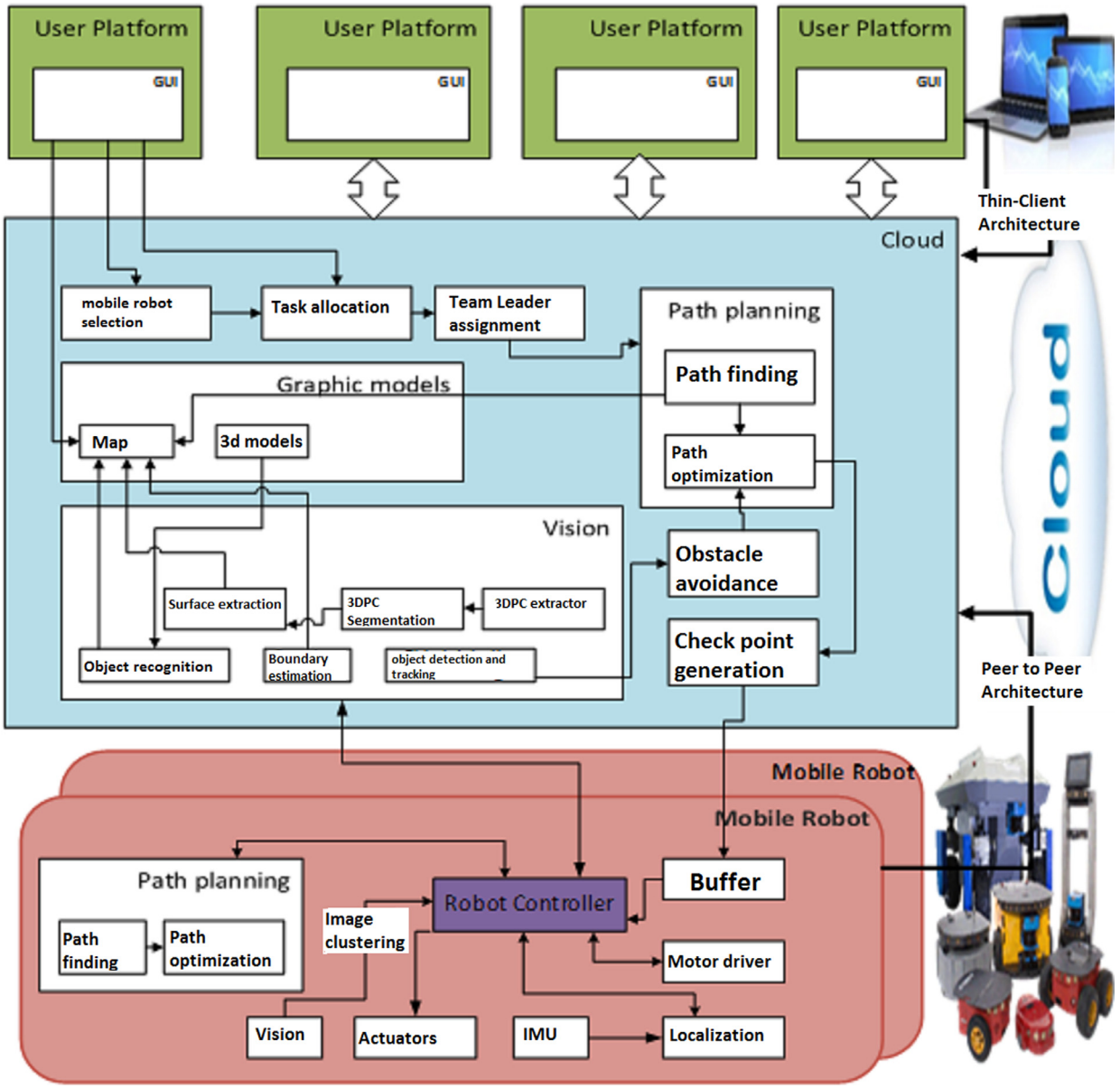


Fig. 6. Conceptual design of the proposed HCMR architecture.

(RSE-PF) proposed in [21], which is capable of reducing network latency by compressing messages before transmission and by adopting the HTTP of posting files to data servers instead of encoding them to JSON strings. The spatially hashed (SH) volume data structure technique proposed in [22] is a solution for large-scale real-time 3D reconstruction on mobile devices with noisy data and limited computational and memory resources, in which RGB-D datasets are used for 3D reconstruction. In the present study, RES-PF [21] and the technique in [20] are considered in the experiments, where lower connection bandwidth is also investigated.

4. Proposed human-cloud-mobile-robot architecture

The main objective of the proposed HCMR architecture is to enable a single or multiple users to control multiple mobile robots simultaneously. A general overview is first presented. The block structure of the proposed architecture, as shown in Fig. 6, consists of three main layers: (i) The user interface layer, (ii) the cloud server layer, and (iii) the mobile robot layer. HCMR can be implemented in a wide range of applications, such as automated urban works, agriculture, search and rescue, army and defense, manufacturing, and home automation.

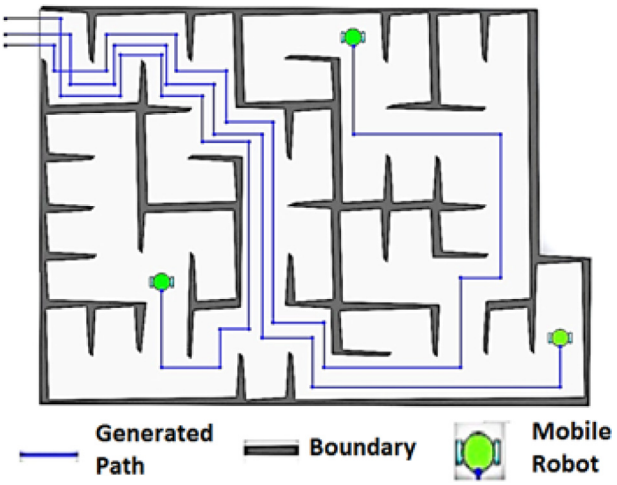


Fig. 7. Selected mobile robots moving simultaneously to the specified target through the generated paths.

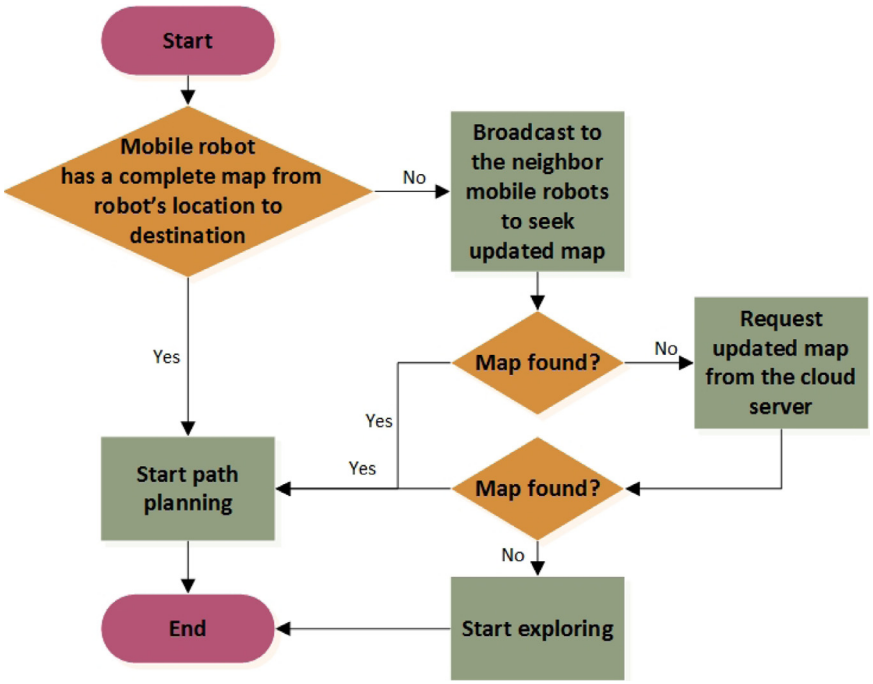


Fig. 8. Robot map query system.

A computer device, tablet, or other platform is used to access cloud applications to preview the rendered location of objects, obstacles, and mobile robots in the virtual map, constructed in the cloud, through a graphical user interface. Each user is connected to the cloud server by thin client architecture. The user platform is provided with an application that can access the virtual map constructed on the cloud server to preview, edit, remove, or add objects in certain locations on the map (e.g. boundaries) so that mobile robots can handle them. The mobile robots are connected to the cloud server using a P2P architecture.

The proposed architecture enables users to select certain mobile robots and then assign a high-level task to them. For example, to move a robot from point A to point B, to handle an object, or to simultaneously perform a cooperative action (e.g. cooperative grasping, object manipulation, formation, foraging, or flocking). Fig. 7 shows a group of mobile robots selected by the user to move to a certain point on the map.

Each user can control the mobile robots by moving them to a certain point (destination point) on the virtual map. The robots may have insufficient or expired information about the surrounding environment; thus, they follow a map query procedure, as shown in Fig. 8, to start path planning. First, a robot seeks an updated map in its local memory. If there is no

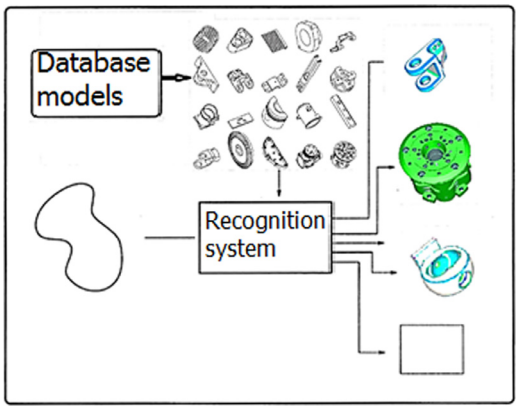


Fig. 9. Object recognition system for service robots.

updated map from the source to the destination, the robot attempts to acquire an updated map from neighboring robots. If this fails, the robot requests the updated map from the cloud server. If this also fails, it starts searching and exploring the environment until it reaches the destination.

In the task allocation module, the high-level task is broken down to lower level subtasks using task planning algorithms. To achieve the allocated task, the robots have to navigate to the target location where the task is to be performed. This is achieved by the pathfinding module, where several paths to the target are explored. For better performance, the HCMR architecture calculates and chooses the shortest path from the source to the target by applying a path optimization algorithm. To attain some degree of autonomy, the robots should be provided with a path planning sublayer so that they can navigate in the surrounding environment in case the connection with the cloud server is slowed down or lost, and in case a group of robots should cooperate in a certain task or move in unison.

If planning should take place in the cloud, the optimal path would break down into a group of straight lines whose endpoints are called checkpoints. These checkpoints are sent to a robot in the form of a sequence of relative polar coordinates (r, θ). A buffer is used to receive the sequence of instructions and checkpoints from the cloud. The buffer sends the checkpoints to the motor driver individually, where they are translated into signals to be sent to the motors. After arriving at the target location, the buffer sends the instructions to the actuators to perform the required task.

Obstacle avoidance is the task of satisfying some control objective subject to non-intersection or non-collision position constraints. It is an active topic of research [23,24] owing to the growing need for easy navigation in urban, industrial, or even military environments. Normally, obstacle avoidance differs from path planning in that it is usually implemented as a reactive control law, whereas path planning involves the pre-computation of an obstacle-free path, along which the robot will be guided by a controller.

To control a group of mobile robots, it is necessary to assign a group leader, so that in case a robot loses connection to the cloud, it may continue receiving instructions through the leader. The leader is a robot with a reliable connection to the cloud, and with the most effective communication with the other robots in the group. It is chosen by applying a leader selection algorithm.

Mobile robot localization is achieved by odometry. The inertial measurement unit (IMU) is an electronic device that measures a mobile robot's orientation, velocity, and gravitational force by means of accelerometers and gyroscopes. IMU sends the measured velocity and direction to the localization module. The localization module, in turn, gives feedback to the motor driver and to the map in the cloud so that the robot's location is re-rendered on the map.

There are several sensing devices that mobile robots can use to describe the surrounding environment, such as, Lidars, stereo cameras, RGBDs, thermal cameras, and ultrasonic sensors. In this study, it is assumed that mobile robots in the proposed architecture will use a stereo vision system. Stereo cameras mounted on mobile robots are used to capture real-time stereo images. The onboard controller sends these stereo images to the cloud for object detection and tracking, object recognition, and boundary estimation. The graphic models module in the cloud server layer is provided with a massive library of 3D models of daily-life, industrial, or any other special objects. The captured images are compared with them to perform the object recognition task, as shown in Fig. 9. In the next section, the vision module in the cloud layer will be explained in detail.

5. Data flow mechanism

To achieve real-time networking and the best utilization of cloud resources, the response time should be reduced when computational tasks are offloaded to the cloud server. The response time can be reduced through (i) simplification of the computational tasks by discarding redundant stereo image clusters, (ii) workload sharing between the mobile robot and the cloud server or neighboring robots, (iii) parallelization of the work between cloud computational nodes, (iv) reducing

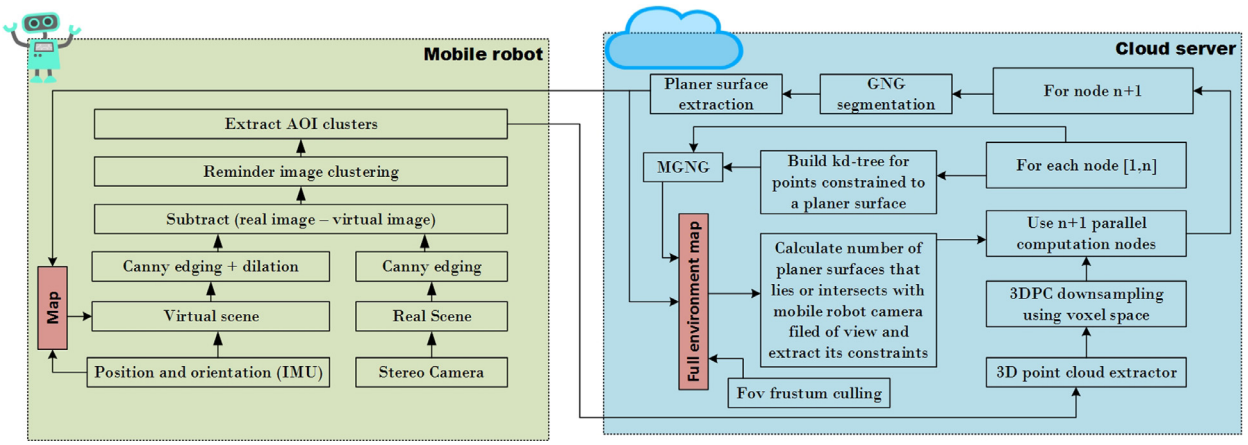
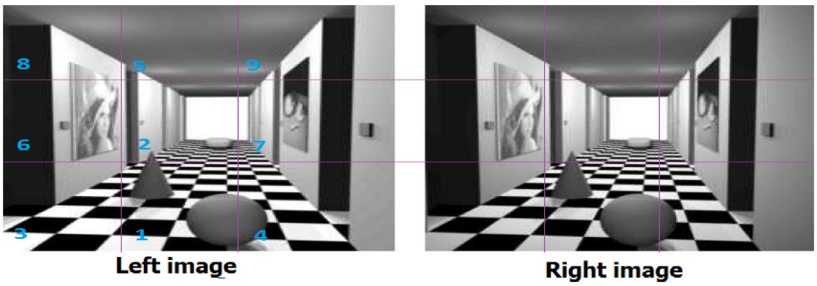
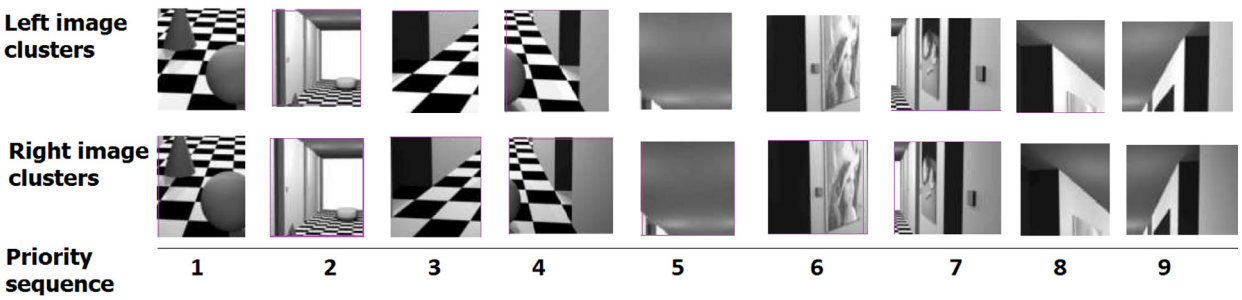


Fig. 10. Cloud-vision data flow mechanism.



(a) Image pairs



(b) Corresponding clustered images in order of priority

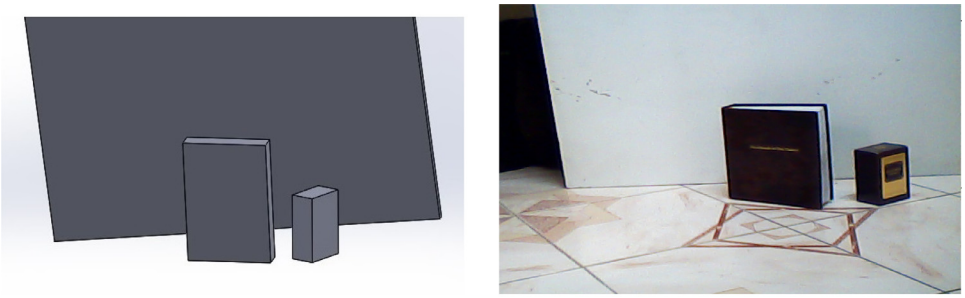
Fig. 11. Various images illustrating image pair clustering.

the amount of data to be sent and received between the mobile robot and the cloud server, and (v) assigning priorities to computational tasks, thus enabling the mobile robot to better handle real-time constraints in the surrounding environment. A robot's reasoning will be enhanced if part of the results arrives from the cloud server earlier while other parts are still being processed in the cloud, as shown in Fig. 16.

Tasks (i)–(v) above can be achieved using the data flow mechanism shown in Fig. 10. In the next sections, there will be a detailed description of the two proposed algorithms, namely, RT-IC, applied on the mobile robot side, and MGNG, applied on the cloud server side. The navigation procedure in the surrounding environment will also be described.

5.1. RT-IC algorithm

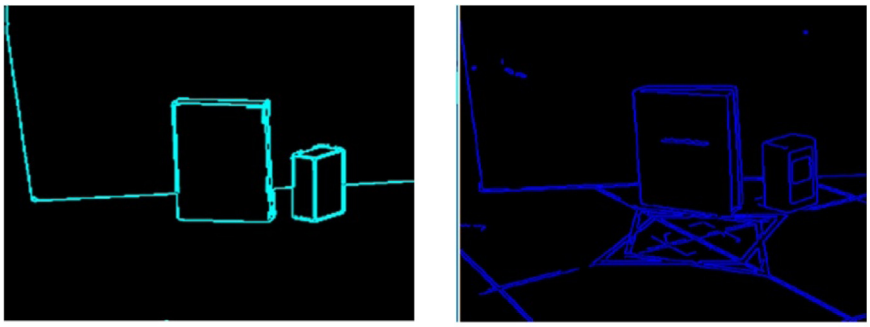
During the exploration process, the mobile robot captures stereo images of the surrounding environment. The camera frame rate is directly proportional to the robot's speed. The image pairs are clustered, and then priorities are assigned to each cluster (the middle lower clusters have higher priority because they are considered to be the nearest objects to the robot. Detecting those near objects will enable the robot to efficiently meet deadlines in obstacle avoidance and path re-planning calculations). As shown in Fig. 11, the top priority is cluster number 1, which is the nearest to the robot.



(a) Virtual image created from a viewport on robot's map

(b) Real scene image captured by the mobile robot's right camera

Fig. 12. Real captured scene with its corresponding virtual scene.



(a) Canny edged virtual image created from a viewport on robot's map

(b) Canny edged real scene image captured by the mobile robot's right camera

Fig. 13. Canny edged real scene with its corresponding canny edged virtual scene.

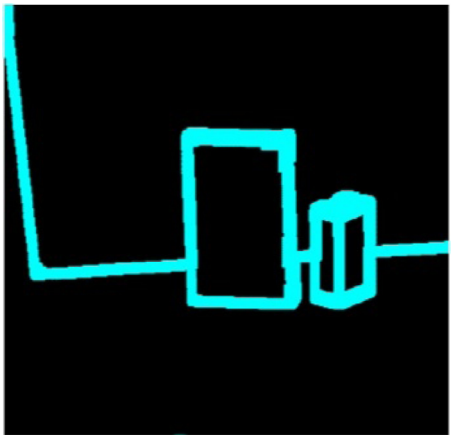


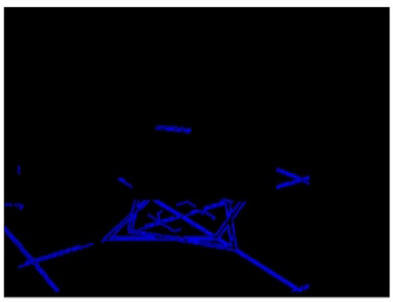
Fig. 14. Dilated canned image of a virtual scene.

As Fig. 12 shows, a virtual scene is obtained by creating a viewport on the map that is stored in the robot's memory. Viewport location and orientation are obtained from the robot's IMU.

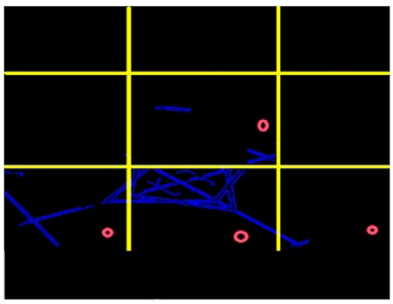
Canny edging is an image processing technique used for edge detection. It is applied to real and virtual scene images, as shown in Fig. 13.

Dilation is an image processing technique for binary images that gradually enlarges region boundaries. By applying this technique to canny edged images, edge thickening is achieved. Here, it is applied to canny edged virtual images. This step is important because it ensures avoiding inaccurate congruence between real and virtual images during the subtraction process. Fig. 14 shows the dilated canned image of a virtual scene.

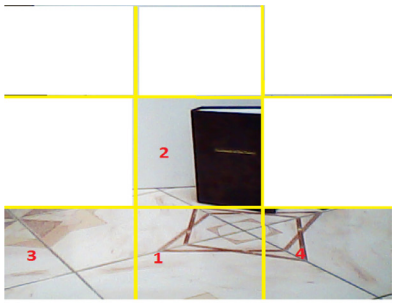
In the subtracting module, the canny edged real scene images are subtracted from the dilated canny edged virtual scene images. Each pixel in the real image is subtracted from each pixel in the virtual image. The result is then clustered into



(a) Image resulting from subtraction process



(b) Four clusters with red circles are AOI, as they contain non zero pixels



(c) Clusters corresponding clusters to AOI that will be offloaded with priority number

Fig. 15. Various images showing the discarding of redundant clusters.

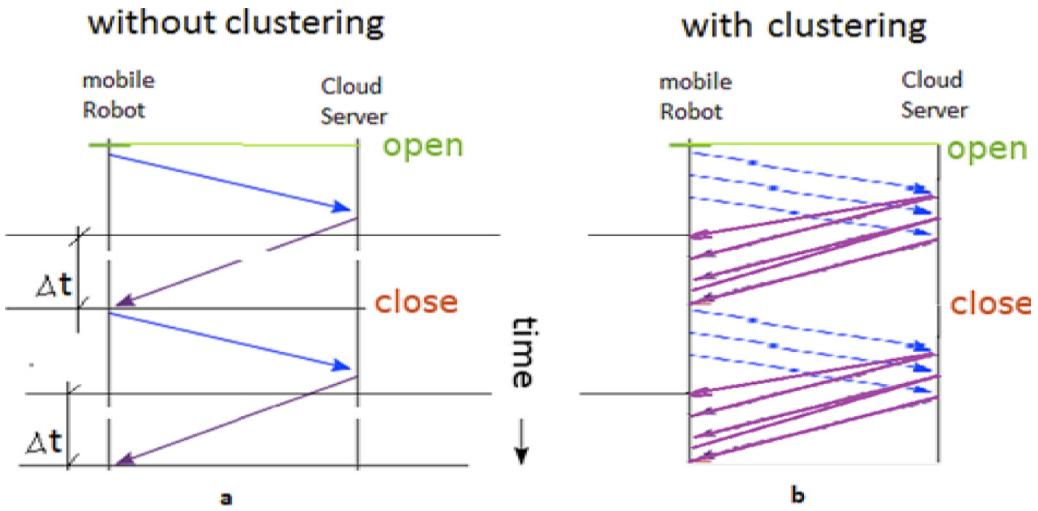


Fig. 16. (a) Traditional offloading method, (b) Clustering offloading method.

three rows and three columns of image clusters. If an image cluster contains non zero pixels, it is considered an area of interest (AoI). Each AoI is then offloaded to the cloud server for surface extraction, and the cloud server map is updated accordingly. Fig. 15 shows the image resulting from the subtraction process, and the AoI clusters. It is clear that textured areas in the real images are not subtracted because 3D reconstructed surfaces are not textured so that they can be offloaded. This problem will be the focus of future research.

Fig. 16 shows the difference in the time analysis of offloading processes with and without clustering. By using the clustering technique, the robot will receive the most important part of the result (the nearest surface segments) earlier by $\triangle t$, thus allowing the reasoning process to start before the complete result is received, as shown in Algorithm 1 (RT-IC).

Algorithm 1 RT-IC Algorithm.

```
PairImage= CapturePairImage
Counter =1
while Counter ≠ 9 do
    ClusterImage= getClusterImage_{prioriyNumber=Counter}
    if ClusterImage= AOI then
        Offload(ClusterImage, MobileRobotID, position, orientation, FrameNo, ClusterPriority)
    end if
    Counter = Counter +1
end while
```

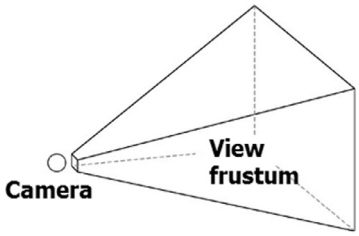


Fig. 17. FOV frustum of the mobile robot.

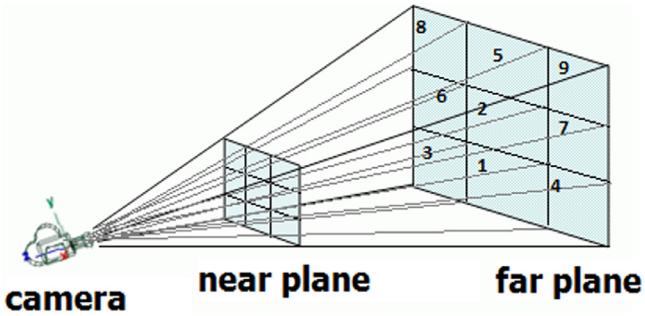


Fig. 18. Main FOV frustum is partitioned into nine frustums.

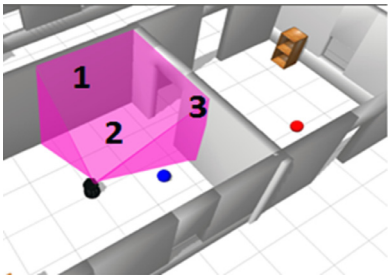


Fig. 19. Cloud server detects the planar surfaces located in the FOV of robot's stereo camera.

5.2. MGNG algorithm

A 3D point-model usually contains thousands of points. 3DPCs are collections of scanned points. Assuming that stereo image extraction produces a 3DPC containing 100 K points, each point will contain coordinates and color: four bytes for each (XYZ) float coordinate and one byte for each (RGB) color ingredient. This will require each robot to have a connection bandwidth of 15 MB/s, which is considered excessively high for real-time networking. 3DPCs may not even contain any topological information. However, most of the topological information can be deduced by applying suitable algorithms. Thus, a surface extraction technique is used after the application of 3D point extraction to the stereo clustered images for the following reasons:

1. To reduce the amount of data transferred from the cloud server to each robot.
2. To achieve mapping accuracy by providing robots with a set of surfaces that contain sufficient topological information.
3. To simplify path planning calculations in the local robot controller, as it is highly efficient to estimate the optimal path using a set of surfaces rather than searching each time in a 3DPC.

The mobile robot sends the position and orientation of the stereo image to the cloud server. From the position and orientation values, the cloud server can simulate the field of view (FOV) frustum of the robot at the current frame on the full map stored on the cloud server. Fig. 17 shows the FOV frustum of a camera. As the image pairs are clustered into nine clusters, the FOV frustum must be partitioned into nine frustums as well. Fig. 18 shows the partitioned main frustum. From the map stored on the cloud server, the number (n) of planar surfaces that lie inside or intersects with the FOV of the robot's camera can be calculated. Then the cloud server obtains the equations of the planar surfaces to be used in the region growing layer. Subsequently, the dedicated frustum partition is used to perform a FOV culling by deleting all surfaces that fully lie inside the frustum and to trim the inside parts of the intersecting surfaces. The point cloud is sampled by using a voxel grid to reduce the total processing cost and decrease the number of noisy points.

Accordingly, the cloud server can parallelize the computation by executing (n+1) computational nodes. Each node from *1* to *n* is responsible for updating each planar surface structure by using the extracted 3DPC. One additional computation node is responsible for non-planar surface extraction. For each point *p* in the 3DPC, if it approximately satisfies the equation of any of n planar surfaces, it will be moved to the computational node corresponding to this planar surface. Once the point *p* is moved, it will be inserted in a k–d tree. All points in the same computation node are extracted to separate planar surfaces using the MGNG algorithm discussed in Section 5.2.1. These surfaces are then returned to update the robot's map and the server's full map. After the entire 3DPC has been searched, there will be a set of remaining unconstrained points $P_{UnConst}$ that do not belong to any of the planar surfaces. These points will be inserted in a *k–d* tree, and the GNG algorithm (discussed in Section 5.2.2) will be applied to extract planar and non-planar surfaces. The result will be returned to the robot's map and to the full map on the cloud server. Each computation node has now a set of points, where segmentation and surface extraction are performed.

5.2.1. Planar-constrained computation nodes

For computational nodes that have planar point cloud constraint, the procedure and notation used in GNG are shown in Algorithm 2, where c_{st} is the edge joining s and t, c_{sn} is the edge joining s and its neighboring node, e_w and e_n are

Algorithm 2 MGNG algorithm.

Create two randomly positioned nodes w_s, w_t in R^3 and connect them with a zero age edge with error = 0
while Stopping criteria are not met **do**
  Generate random input data $\bar{X}$
  Locate the two nodes s (nearest) and t (2nd nearest) nearest to $\bar{X}$
  if $c_{st} = 0$ **then**
    $c_{st} = 1$
  end if
  $error_s = error_s + ||\bar{w_s} - \bar{X}||^2$
  $w_s = w_s + ew(\bar{X} - \bar{w_s})$
  $w_n = w_n + en(\bar{X} - \bar{w_n})$
  $Age_{c_{s,n}} = Age_{c_{s,n}} + 1, \forall n \in toNeighbors$
  if $c_{s,n} = 0$ **then**
    $c_{s,n} = 1$
  else
    $Age_{c_{s,n}} = 0$
  end if
  if $Age_c > Age_{\max}, \forall$edge c$\in C$ **then**
    $c = 0$
    if $n_{ccount} = 0, \forall$node n$\in N$ **then**
      Remove n
    end if
  end if
  if $Step_{number}/\lambda = Integer$ **then**
    Insert node N_{new} to $Node_{list}$
    $\bar{W}_{N_{new}} = \frac{\bar{W}_{N_{Nearest}} + \bar{W}_{N_{2ndNearest}}}{2}$
    $\bar{error}_{N_{new}} = \bar{error}_{N_{nearest}}$
    $\bar{error}_{N_{nearest}} = \alpha * \bar{error}_{N_{nearest}}$
    $\bar{error}_{N_{2ndNearest}} = \alpha * \bar{error}_{N_{2ndNearest}}$
  end if
end while
while Not all nodes are inside a boundary **do**
  $Node_{BoundaryList_0} = Node_{list\,leftMost}$
  $Currentnode = Node_{BoundaryList_0}$
  while $CurrentNode \neq Node_{BoundaryList_0}$ OR $1stLoop$ **do**
    $Node_{BoundaryList_{NodeNumber}} = Node_{MaxAngle_{CurrentNode,+Y}} \forall node \in currentNodeNeighbors$
    $currentnode = Node_{BoundaryList_n}$
    n=n+1
  end while
end while
while Stopping criteria are not met **do**
  $\bar{W}_{NodeBoundary} = \bar{W}_{outsideKneighbor} \forall BoundaryNode \in BoundaryList$
end while

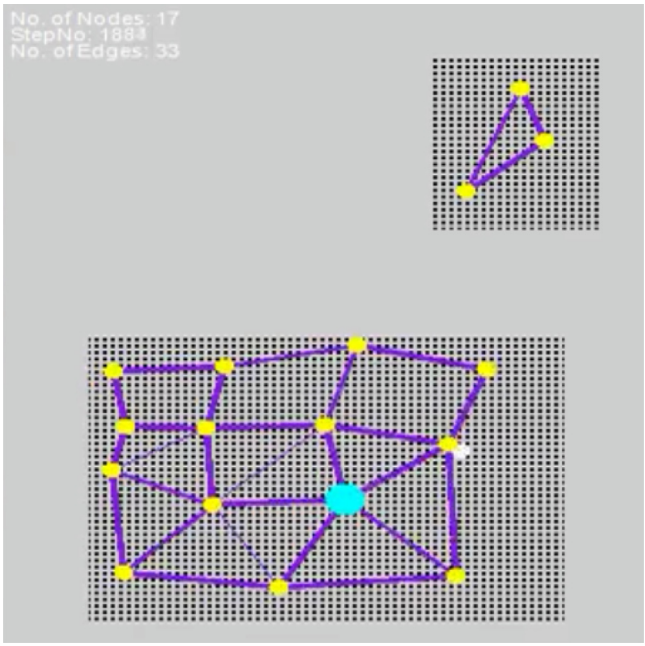


Fig. 20. Planar point cloud is segmented into two topologies.

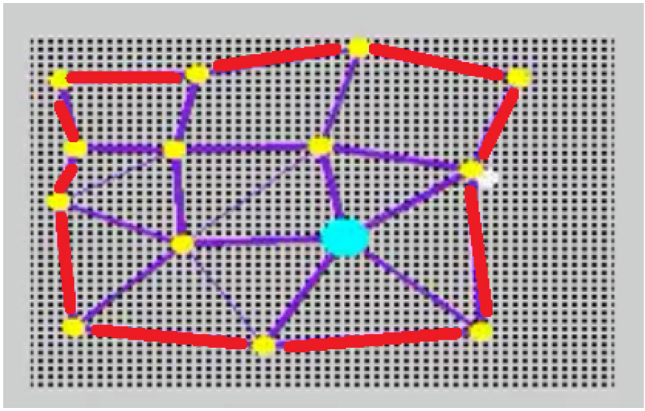


Fig. 21. Contour edges extracted marked in red color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

factors in the interval [0,1], vs is the position vector of the s node, and w_n is the position vector of a node neighboring s. Thereby, the topological structure of the 3DPC can be learned by the MGNG algorithm after the computational node learns the topological structure of the planar point cloud, as shown in Fig. 20. The outer contour of each segment is extracted as shown in Fig. 21. This will facilitate the application of the region growing algorithm and reduce the amount of data that will be sent to the mobile robot, thereby reducing bandwidth consumption.

After the extraction of the outer contour of each segment, its outer nodes are enforced by the MGNG algorithm to push outside and completely fit the 3DPC using the *kd-tree* arrangement, as shown in Fig. 22.

5.2.2. Non-planar constrained computation nodes

For computational nodes that have non-planar point cloud constraint, it is required that planar and non-planar surfaces be extracted. Accordingly, he GNG algorithm is applied to learn the topological structure of the 3DPC. Subsequently, segment planarity is checked. If the segment is planar, it will then follow the pipeline in Section 5.2.1. The planar segments are to be applied to GNG with planar surface extraction. Then, a segment is sent to the mobile robot, and the full map on the cloud server is updated.

The planarity check algorithm depends on estimating the normal vectors for all nodes, whereby the similarity $s_{i,j}$ between the i_{th} and j_{th} nodes is defined. Specifically, similarity can be defined using the inner product of the normal vectors between

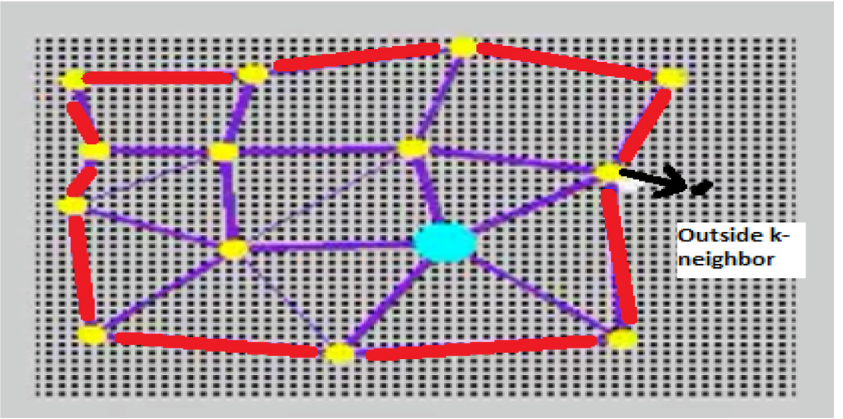


Fig. 22. Boundary node is enforced to push outside to the nearest k-neighbor point.

nodes, and equals the angle between the normal vectors by applying Eq. (4). If $s_{i,j}$ is approximately equal to 1 for all nodes in the segment, the entire segment is planar.

$$s_{i,j} = \frac{n_i.n_j}{|n_i|.|n_j|} \tag{4}$$

5.3. Mobile robot navigation

In the proposed HCMR architecture, the cloud server recognizes ground surfaces, i.e., planar surfaces on which the mobile robot can move, as those with node normal approximately parallel to the Z vector of the mobile robot, assuming the terrain is flat, and their Z levels are approximately equal to the Z level of the robot's wheels. The cloud server will not extract this surface as an outer contour of edges and vertices but as a raw nodal network generated directly from the GNG algorithm.

When the mobile robot navigates from its current point S to a destination point D, it will first reach the nearest node point $S_{nearest}$ in the nodal network of the ground surface. During its movement toward $S_{nearest}$, the robot estimates the shortest path between $S_{nearest}$ and the nearest node $D_{nearest}$ to D. Then, the robot navigates along the shortest path selected to $D_{nearest}$ and then to D, as shown in Fig. 23. In this method, the time required for path planning is greatly reduced because a simple algorithm through a nodal network is used compared to path planning through a 3DPC, which requires longer time to execute a heuristic search using a complex artificial intelligence algorithm for optimal path selection.

6. Experimental result discussion and analysis

In this section, an application of the proposed approach is presented. The proposed approach takes into consideration different conditions, image resolutions, point cloud sizes, and network bandwidths. On the mobile robot side, a Python code with the OpenCV package was used to cluster the pair images of each frame and to discard redundant clusters. On the cloud server side, the 3DPC extractor code written in Python with the OpenCV package was first used followed by the point cloud segmentation code MGNG written in C++ with OpenGL on an Intel 2.5 GHz Corei5 processor with 6 GB RAM. The proposed approach was applied to individual frames. To simulate cloud performance, the code was written so as to allow multiple computation nodes and parallel processing. The cloud simulation parameters, such as number of created cloudlets, instruction length of each cloudlet, and number of created computation nodes (virtual machines), were monitored to extract different sets of image pairs. The parameters were applied to "CloudSim" (a professional cloud simulator), and the results were found to be valid. The settings $age_{\max} = 30 = 30$, $e_w = 0.05$, and $e_n = 0.002$ were used. The experimental results were analyzed in terms of response time, data size path planning time, and accuracy. The related discussion will be presented in the next sections.

6.1. Response time analysis

To prove the efficiency of the proposed approach, the response time was used as a metric. Response time, as indicated in Eq. (5), is the total time T_t for offloading clustered images, processing, and receiving 3D surfaces for one frame. The obtained results were compared with those by other state-of-the-art techniques that do not use clustering or MGNG segmentation, such as the technique in [20] and RSE-PF [21].

$$T_t = T_0 + T_p + T_C, \tag{5}$$

where T_t, T_0, T_p, and T_c denote total time, time for offloading image pairs to the cloud server, time for surface extraction from image pairs, and time for receiving results from the cloud server, respectively.

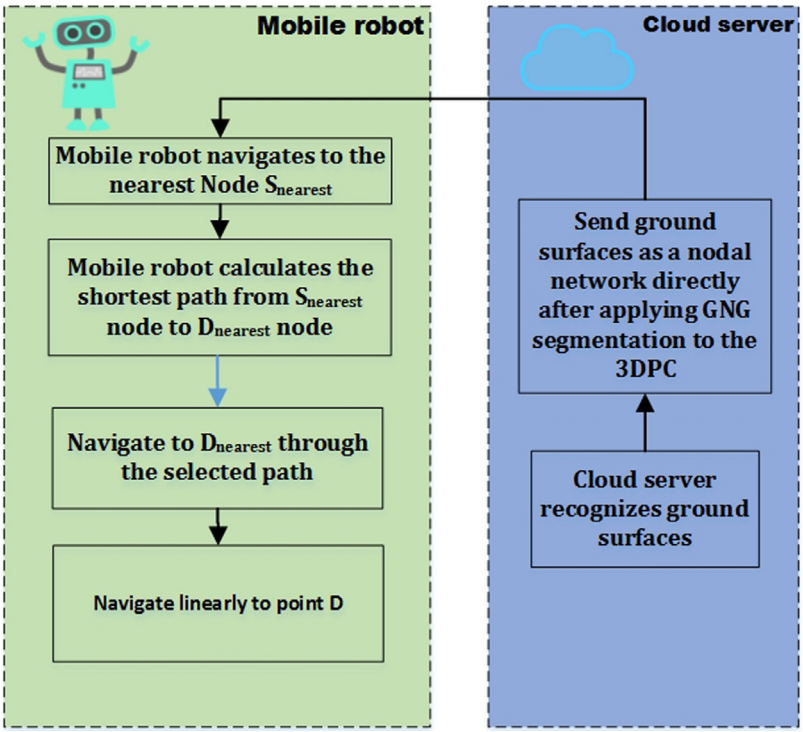


Fig. 23. Mobile robot navigation in the surrounding environment.

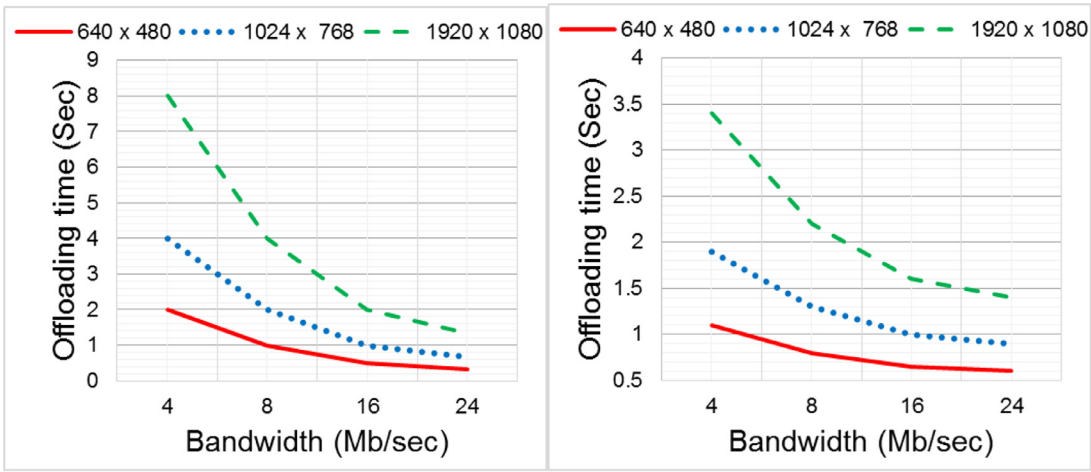
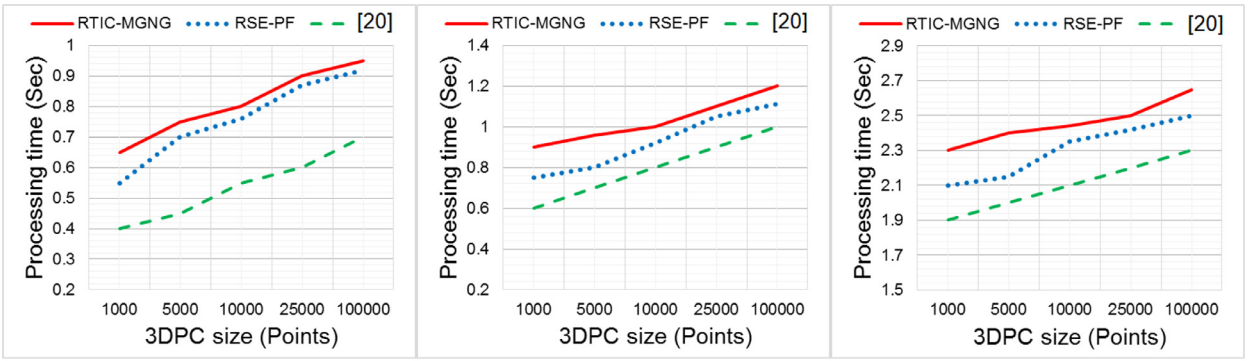


Fig. 24. Offloading time for different sizes of image pairs versus different communication bandwidths.

Fig. 24(a) shows the time required to offload a pair of high-resolution images to the cloud server. There is no major difference in t_0 between RT-IC with MGNG and the approach in [20], as in both cases, image pairs of the same size will be offloaded on the assumption that no redundant clusters are discarded; otherwise, the size of the image pairs will be reduced, and consequently t_0 for RT-IC with MGNG will also be reduced. Fig. 24 (b) shows lower offloading time for RSE-PF because it uses data compression before transmission, which may increase processing time, as will be seen later.

Fig. 25 shows the time required for image pair processing to extract the 3DPC using RT-IC and MGNG compared with RSE-PF and the technique in [20]. Different sets of image resolutions and 3DPC sizes were considered in Fig. 25 (a)–(c). It is obvious that the processing time for the MGNG algorithm is slightly longer than that for the other techniques because MGNG starts nodal network surface extraction immediately after 3DPC extraction. Moreover, the processing time was calculated without parallelization. The proposed approach, however, pipelines the cluster processing task within the same frame, thus

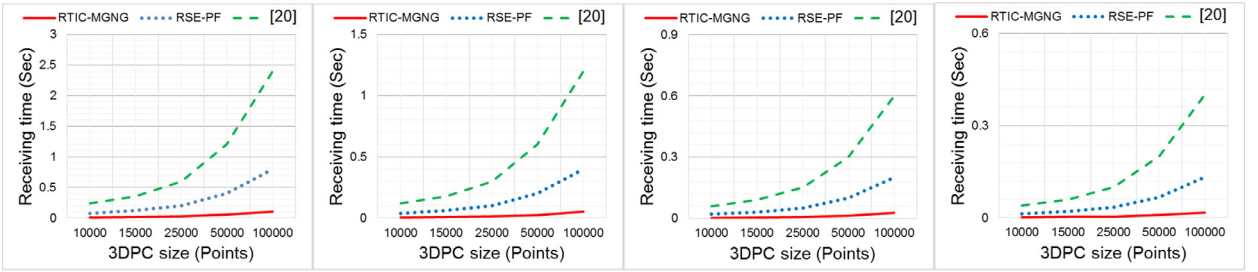


(a) 640 x 480 (b) 1024 x 768 (c) 1920 x 1080

Fig. 25. Processing time comparison between RT-IC with MGNG and other state-of-the-art techniques.

Table 1
Time analysis (s) for processing different sizes of stereo images versus different bandwidths.

Technique	RTIC_MGNG					[20]					RSE-PF				
Image_Sizes	Mb/s	t_0	t_p/cluster	t_c	t_t/**frame**	Mb/s	t_0	tp/frame	t_c	t_t/**frame**	Mb/s	t_0	t_p/frame	t_c	t_t/**frame**
640_x_480	4	2	0.08	0.006	**2.08**	4	2	0.55	0.15	**2.7**	4	0.67	0.61	0.05	**1.31**
	8	1	0.08	0.003	**1.083**	8	1	0.55	0.075	**1.625**	8	0.35	0.61	0.024	**0.94**
	16	0.5	0.08	0.0015	**0.583**	16	0.5	0.55	0.0375	**1.087**	16	0.16	0.61	0.01	**0.79**
	24	0.33	0.08	0.001	**0.413**	24	0.33	0.55	0.025	**0.905**	24	0.12	0.61	0.01	**0.73**
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
1024_x_768	4	4	0.087	0.03	**4.092**	4	4	0.6	0.3	**4.9**	4	1.35	0.77	0.12	**2.24**
	8	2	0.087	0.015	**2.09**	8	2	0.6	0.15	**2.75**	8	0.72	0.77	0.05	**1.54**
	16	1	0.087	0.01	**1.09**	16	1	0.6	0.1	**1.7**	16	0.45	0.77	0.031	**1.25**
	24	0.66	0.087	0.005	**0.749**	24	0.66	0.6	0.05	**1.31**	24	0.297	0.77	0.018	**1.01**
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
1920_x_1080	4	8	0.21	0.06	**8.284**	4	8	2.2	0.6	**10.8**	4	2.43	2.42	0.18	**5.03**
	8	4	0.21	0.03	**4.281**	8	4	2.2	0.3	**6.5**	8	1.26	2.42	0.09	**3.77**
	16	2	0.21	0.015	**2.279**	16	2	2.2	0.15	**4.35**	16	0.63	2.42	0.05	**3.1**
	24	1.33	0.21	0.01	**1.608**	24	1.33	2.2	0.1	**3.63**	24	0.39	2.42	0.03	**2.84**



(a) 4 Mb/sec (b) 8 Mb/sec (c) 16 Mb/sec (d) 24 Mb/sec

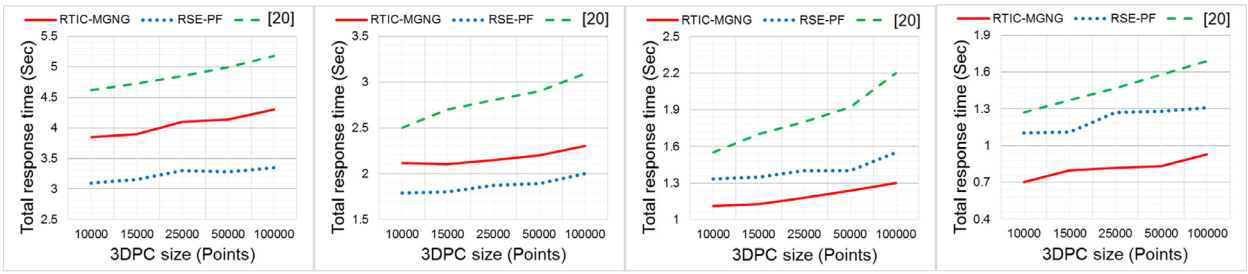
Fig. 26. Comparison between receiving time for RT-IC with MGNG and other state-of-the-art techniques.

reducing the total processing time of the entire frame on the cloud server, as shown in the total response time analysis in Fig. 27 and Table 1.

Fig. 26(a)–(d) show the time required to receive the extracted 3DPC from the cloud server for different 3DPC sizes. It can be concluded that the proposed approach is less time-consuming in receiving data compared with other techniques because the mobile robot receives a lightweight mapping dataset that consist of nodal network segments, whereas the mapping datasets in RSE-PF and the technique in [20] consist of dense 3DPCs.

Fig. 27 shows the total response time analysis for the proposed approach in comparison with other state-of-the-art techniques. Different bandwidths have been considered, indicating the advantage of the proposed approach over the technique in [20] in Fig. 27 (a)–(d), and over RSE-PF for relatively higher bandwidths (above 8 Mbit/s) in Fig. 27 (c) and (d).

Table 1 shows a time analysis for different image resolutions versus different network bandwidths to compare the response time using RT-IC with MGNG and other state-of-the-art techniques. As it is clear that the total time required to



(a) 4 Mb/sec (b) 8 Mb/sec (c) 16 Mb/sec (d) 24 Mb/sec

Fig. 27. Comparison between total response time for RT-IC with MGNG and other state-of-the-art techniques.

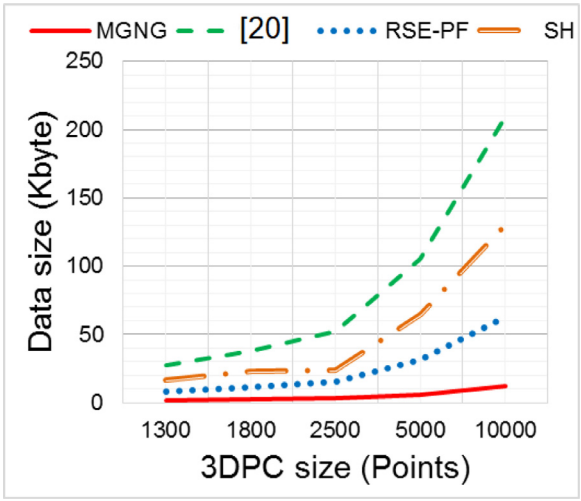


Fig. 28. Data size of generated MGNG surface maps compared with different point cloud maps generated by other techniques.

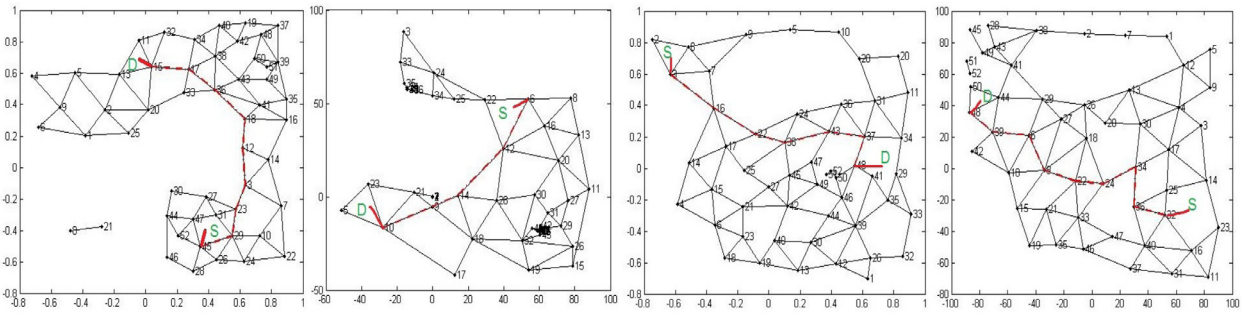
process one frame of stereo images on the cloud server using RT-IC with MGNG is less than that for other techniques in most cases, the superiority of the proposed approach is demonstrated.

6.2. Data size analysis

In this section, the amount of data received from the cloud server after 3D map reconstruction is investigated as another metric to measure the efficiency of the proposed approach. Typically, this experiment was conducted several times to measure the received data sizes for different 3DPC sets for the extracted frame and their corresponding nodal network segments that completely fit the 3DPC. Surely, the size of the received data affects not only network bandwidth usage but also performance. Mobile robots may have limited memory, power, and storage capabilities. Therefore, the data received should be minimized as much as possible. In addition, if memory usage is minimized, further calculations by the robot will be facilitated, and hence performance will be enhanced. Fig. 28 shows the advantage of using the MGNG algorithm to reduce the size of the received mapping results in comparison with other state-of-the-art techniques, namely, the techniques in [21] (RSE-PF), [20,22] (SH).

6.3. Path planning time analysis

To prove the effectiveness of the proposed approach not only in scene reconstruction but also in path planning, Dijkstra's algorithm was used to select the shortest path from source to destination in an MGNG generated nodal network simulated in MATLAB. The proposed simulated Dijkstra algorithm is referred to here as MGNG-D. The required time for path planning in nodal networks extracted from 3DPCs of different sizes was measured, as shown in Fig. 29. Fig. 30 shows the measured path planning time in nodal networks extracted from 3DPCs of different sizes. It can be seen that path planning through a pre-extracted nodal network using MGNG is less time consuming compared with those by other related state-of-the-art techniques, namely the FSE (full subsumption Euclidean) and FSW (full subsumption weighted) techniques presented in [25]. Fig. 31 shows the robot battery life using MGNG-D periodically compared with other techniques. Battery life is estimated mathematically as $BatteryLife = BatteryCapacity/[\alpha * TaskExecutionTime * LengthOfInstructions]$, where α is a factor that indicates power dissipation per instruction. This factor is estimated experimentally.



(a) 15K points in 156 ms (b) 20K points in 183 ms (c) 12K points in 146 ms (d) 18K points in 167 ms

Fig. 29. Various figures showing path planning from a source node to a destination node in a nodal network generated from different 3DPC sets.

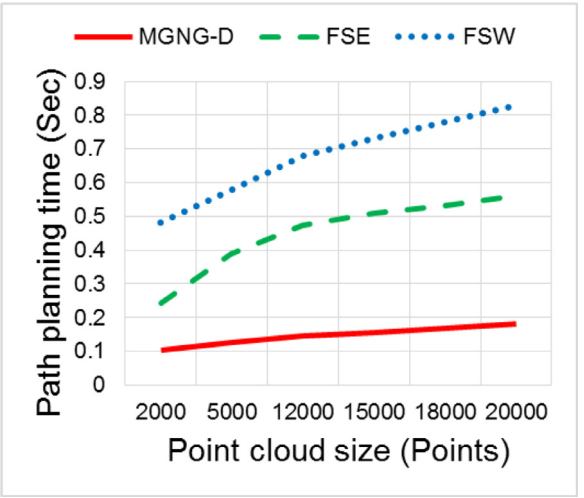


Fig. 30. Path planning time using MGNG-D and other state-of-the-art techniques for different sets of mapping data.

6.4. Accuracy analysis

The accuracy of the final mapping is constrained by several factors, such as image pair resolution, 3DPC extraction algorithm, and MGNG parameters. In the proposed MGNG algorithm, optimal parameter values were used to allow accurate mapping results. The accuracy of the proposed approach is greatly improved compared with other state-of-the-art techniques because a) it can use high-resolution image pairs for mapping with less response time compared with other techniques. As high-quality images are used, it yields better 3DPC accuracy; b) MGNG converts 3DPCs into nodal network surfaces that completely match the 3DPC shape, and thus it is robust against noisy and cluttered points. The approximated error in dimension measurements between mapping results and the real region of interest was obtained for different techniques constrained by computation offloading frequency, which is the number of times the mobile robot sends stereo cluster pairs to the cloud server, extracts mapping results and receives them back for one frame per second. In this case, image resolution is specified so as to maintain computation offloading frequency. In Fig. 32, each technique uses different image resolutions to maintain a computation offloading frequency of 2 Hz. It can be deduced that the proposed approach has the highest mapping accuracy owing to its ability to maintain higher stereo image resolution within the specified computation offloading frequency in comparison with the other state-of-the-art techniques.

7. Conclusion and future work

The HCMR architecture was proposed, whereby a single user can control a group of mobile robots simultaneously. HCMR uses cloud technology, which allows an open shared environment with dynamically scalable computing capability. Cloud computing stands short of facing the challenge of dealing with real-time applications due to its limited networking capabilities comparatively with the high data transmission needed by such applications. A data flow mechanism was proposed that overcomes this problem through simplification of the computational tasks, workload sharing, parallelization of the work between cloud computation nodes, and reduction of the amount of data to be sent and received between the mobile robot and the cloud server.

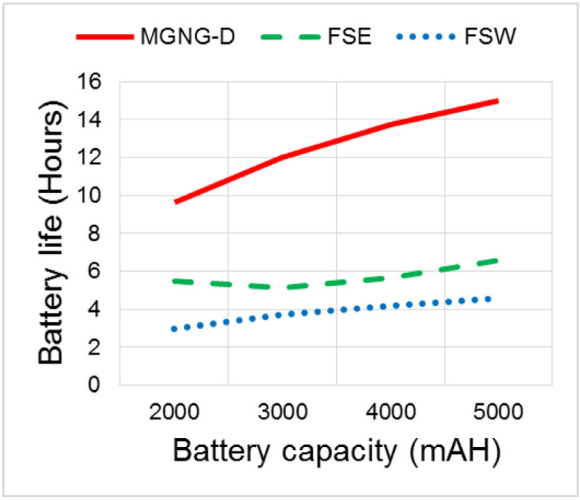


Fig. 31. Battery life in MGNG-D and other state-of-the art techniques.

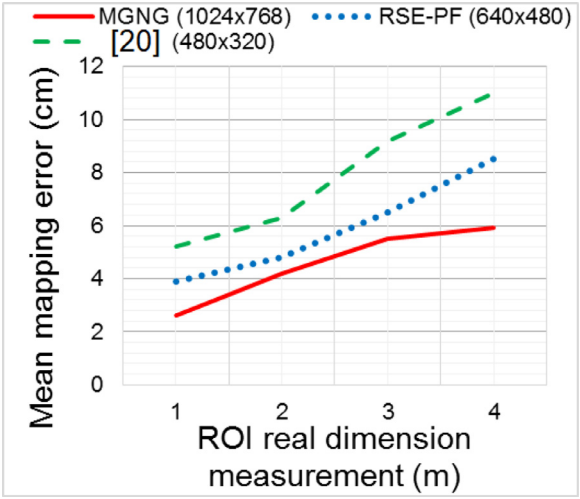


Fig. 32. Accuracy measurement of our approach compared with other state-of-the-art techniques.

Although the experimental results demonstrated the feasibility of the proposed approach, there are certain problems that should be investigated in future work. For instance, the mobile robot clustering algorithm does not eliminate redundant clusters that contain textures; furthermore, it eliminates redundant clusters and hence reduces network utilization, but it may increase processor utilization. However, critical processor utilization was not noticed. Additionally, the HCMR architecture should be extended to include edge computing technology. The data flow mechanism including the 3D point processing algorithms should be accordingly redesigned so as to allow greater reduction of the response time not only for navigation but also for task achievement. In the present study, MGNG was designed to categorize 3D points into planar-constrained points and non-planar-constrained points. Thus, the proposed MGNG algorithm should be extended so as to include more regular geometric surfaces such as cones, ellipsoids, cylinders, or even partial geometric shapes.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.compeleceng.2019.01.001.

References

[1] Kuffner J. Cloud-enabled humanoid robots. In: Proceedings of the IEEE-RAS 10th international conference on humanoid robots (Humanoids) Nashville TN, United States, Dec.; 2010.
[2] Remmersmann T, Schade U, Schlick C. Supervisory control of multi-robot systems by disaggregation and scheduling of quasi-natural language commands. In: Proceedings of the IEEE international conference on systems, man, and cybernetics (SMC). IEEE; 2012. p. 315–20.

[3] Podevijn G, O'Grady R, Nashed YS, Dorigo M. Gesturing at subswarms: towards direct human control of robot swarms. In: Proceedings of the conference towards autonomous robotic systems. Springer; 2013. p. 390–403.
[4] Shafivulla M, Rajesh V, Khan H. Semg based human computer interface for robotic wheel. In: Proceedings of the 4th international conference on intelligent human computer interaction (IHCI). IEEE; 2012. p. 1–5.
[5] Falcone E, Gockley R, Porter E, Nourbakhsh I. The personal rover project:: the comprehensive design of a domestic personal robot. Rob Auton Syst 2003;42(3–4):245–58.
[6] Paolini C, Lee GK. A web-based user interface for a mobile robotic system. In: Proceedings of the IEEE 13th international conference on information reuse and integration (IRI). IEEE; 2012. p. 45–50.
[7] Kakoty NM, Mazumdar M, Sonowal D. Mobile robot navigation in unknown dynamic environment inspired by human pedestrian behavior. In: Proceedings of the progress in advanced computing and intelligent engineering. Springer; 2019. p. 441–51.
[8] Ochiai Y, Takemura K, Ikeda A, Takamatsu J, Ogasawara T. Remote control system for multiple mobile robots using touch panel interface and autonomous mobility. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS 2014). IEEE; 2014. p. 3272–7.
[9] Quintas J, Menezes P, Dias J. Cloud robotics: towards context aware robotic networks. In: Proceedings of the international conference on robotics; 2011. p. 420–7.
[10] Du Z, He L, Chen Y, Xiao Y, Gao P, Wang T. Robot cloud: bridging the power of robotics and cloud computing. Future Generat. Comput. Syst. 2016.
[11] Ayanian N, Spielberg A, Arbesfeld M, Strauss J, Rus D. Controlling a team of robots with a single input. In: Proceedings of the IEEE international conference on robotics and automation (ICRA). IEEE; 2014. p. 1755–62.
[12] McCann E, McSheehy S, Yanco H. Multi-user multi-touch multi-robot command and control of multiple simulated robots. In: Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction. ACM; 2012. p. 413–14.
[13] Karulf E, Strother M, Dunton P, Smart WD. RIDE: A Mixed-Mode Control Interface for Mobile Robot Teams Report Number: WUCSE-2012-3. All Computer Science and Engineering Research; 2012.
[14] Martinez-Gomez J, Fernandez-Caballero A, Garcia-Varea I, Rodriguez L, Romero-Gonzalez C. A taxonomy of vision systems for ground mobile robots. Int J Adv Rob Syst 2014;11(7):111.
[15] Bistry H, Zhang J. A cloud computing approach to complex robot vision tasks using smart camera systems. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE; 2010. p. 3195–200.
[16] Riazuelo L, Civera J, Montiel J. C2 tam: a cloud framework for cooperative tracking and mapping. Rob Auton Syst 2014;62(4):401–13.
[17] Nimmagadda Y, Kumar K, Lu Y-H, Lee CG. Real-time moving object recognition and tracking using computation offloading. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE; 2010. p. 2449–55.
[18] Wu H, Lou L, Chen C-C, Hirche S, Kuhnlenz K. Cloud-based networked visual servo control. IEEE Trans Ind Electron 2013;60(2):554–66.
[19] Agostinho L, Olivi L, Feliciano G, Paolieri F, Rodrigues D, Cardozo E, et al. A cloud computing environment for supporting networked robotics applications. In: Proceedings of the IEEE ninth international conference on dependable, autonomic and secure computing (DASC). IEEE; 2011. p. 1110–16.
[20] Salmerón-García J, Iñigo-Blasco P, Díaz-del Río F, Cagigas-Muñiz D. Study of communication issues in dynamically scalable cloud-based vision systems for mobile robots. In: Robots and sensor clouds. Springer; 2016. p. 33–52.
[21] Yun P, Jiao J, Liu M. Towards a cloud robotics platform for distributed visual slam. In: Proceedings of the international conference on computer vision systems. Springer; 2017. p. 3–15.
[22] Dryanovski I, Klingensmith M, Srinivasa SS, Xiao J. Large-scale, real-time 3D scene reconstruction on a mobile device. Auton Robots 2017;41(6):1423–45.
[23] Mohammad SHA, Jeffril MA, Sariff N. Mobile robot obstacle avoidance by using fuzzy logic technique. In: Proceedings of the IEEE 3rd international conference on system engineering and technology (ICSET). IEEE; 2013. p. 331–5.
[24] Faisal M, Hedjar R, Al Sulaiman M, Al-Mutib K. Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment. Int J Adv Rob Syst 2013;10(1):37.
[25] Schirmer R, Biber P, Stachniss C. Efficient path planning in belief space for safe navigation. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE; 2017. p. 2857–63.

Mahmoud Badawy is Assistant Professor in Computers Engineering and Control systems Dept. - Faculty of Engineering - Mansoura University, Egypt. He received the B.Sc., M.Sc. and Ph.D. from Computers and Systems Engineering, Mansoura University, Egypt. His major research interests are computer networking, Distributed control systems, Database Management systems, mobile robots and cloud computing.

Hisham Khalifa is a software engineer received a Bachelor's degree in Mechanical Design and Production and graduated from a higher studies diploma in Automatic control - Faculty of Engineering Mansoura University, Egypt. His interests are in the areas of software development, mobile robots, computer vision, micro-controller programming, and Hardware design.

Hesham Arafat is a Professor - head of Computers Engineering and Control systems Dept. - Faculty of Engineering - Mansoura University, Egypt. He is a founder member of the IEEE SMC Society Technical Committee on Enterprise Information Systems (EIS). His interests are in the areas of network security, mobile agent, Network management, Search engine, pattern recognition and distributed databases.