VISION-BASED RELOCATION OF OBJECTS USING A MOBILE ROBOT

Ahmed S. Khusheef, Ganesh Kothapalli, and Majid Tolouei-Rad

School of Engineering, Edith Cowan University, Western Australia
Email: ahmed_shany@yahoo.com

(Received November 2012 and Accepted May 2013)

This study proposes a vision-based methodology for searching and relocation of target objects using a hexapod mobile robot. Such robot needs enhanced systems for navigation and vision-based object recognition. The navigation system is important for generating a path that covers the entire environment and for locating the position of the robot within that environment. Vision system is also important for undertaking the exploration task; it involves using a vision sensor and employing an optimal object recognition technique. This paper explains the development of such systems and presents a strategy for constructing the sensory platform to enable the robot to navigate and explore its environment. It also explains the challenges that arise from the viewpoint of vision sensor to the objects and suggests some relevant solutions. The experimental results demonstrate that it is possible to implement a navigation system within a minimum number of sensors if they are properly positioned in the robot’s body. The Experiments proved that the methodology used and that the codes developed made the robot capable of performing its task of finding, approaching and relocating objects as planned.

1. INTRODUCTION

There are many types of manufacturing operations and environments for which mobile robots can be used to search, find and relocate objects [1]. Novel challenges exist in developing a control system that helps a robot to navigate and search its environment. These include constructing an optimal navigation system that enables the mobile robot to search the whole searched area as the target might reside with equal probability at any location. As the robot is performing a visual search, choosing an exploration strategy and vision-based object recognition techniques is also difficult. One aspect is the object detection process which is challenging because the robot needs to navigate and place the object in the view field. The robot also requires a vision system that employs some image analysis techniques which are sensitive to environmental conditions such as lighting, texture and background color.

The exploration path has to be carefully planned to cover the robot’s entire environment while taking account of the visibility of the target and optimizing both navigation time and collision avoidance [2]. It must help the robot approach and observe the target efficiently through optimal object recognition methods; typically using vision sensors supported by image processing techniques. Relevant strategies differ depending on whether the environment is static (static obstacles) or dynamic (static and dynamic obstacles) [3]. Both categories can be subdivided into unknown and known environments; in the latter, information is provided on the location of obstacles before motion commences. Across the various environments, there are many navigation algorithms that deal with the robot’s navigation problem [4]. These algorithms are classified into global and local planning [5]. The former plans the robot’s path from the start to the goal by searching a graph that represents a map of the global environment (see [6-8]). Typically, the global planning methods have three drawbacks; expensive to compute, complex to construct, and difficult to obtain an accurate graph model.

Research on achieving exploration paths for mobile search robots has generally relied on global navigation planning. For instance, Fukazawa, et al. [9] proposed a points-distribution, path-generation algorithm in which the robot is given a set of points
that completely cover the environment. The robot in that study sought the shortest path that covered all of these points and kept looking for an object while it moved along the path and then relocated the detected object. The authors assumed that the robot had a complete map of the environment. They also argued that three types of the path planning algorithms can cover the entire environment in exploration applications involving the random walk, the spiral path and the zigzag path. The authors considered that the random walk cannot guarantee accomplishment of the exploration task. The other two techniques generate the exploration path by joining line segments arranged in the environment. Clearly, the computational cost for creating a path increases with the total number of line segments.

Another study proposed an efficient approach for modeling the search path by minimizing the expected time required to find the target [10]. In that work, the mobile robot was assumed to be equipped with efficient sensors, the environment containing the object was completely known, and the motion strategy enabled the robot to find the target quickly. The known built environment in [11] was divided into a set of regions for the robot that was used to search for multiple targets. The robot’s task was to discover the sequence of motions that reduced expected time to find the targets. However, the authors in [10,11] did not describe how the robot recognized and discovered the objects. Furthermore, these studies were simulations and did not involve a robot.

Some researchers have tried to avoid constructing a comprehensive environmental map. Tovar’s [12] robot used critical events in on-line sensor measurements, such as crossing lines, to construct a minimal representation that provided a sensor feedback motion strategy. The authors introduced a visibility tree, which represents simply-connected planner environments, to dynamic encode enough information for generating optimal paths. Another study [13] presented a guide tracking method in which the mobile robot is provided with a trail from a starting point to the target location. The benefit of a trail is that the mobile robot reaches the target location with little autonomous navigation skills. However, the trail needs to be shaped prior to the robot’s navigation process. Most of the algorithms described above assume that the environment is known before the navigation takes place. The authors also did not explain the object recognition systems that were employed through the search process.

Conversely, the local navigation algorithms use information from the sensors for commands that control the robot’s motion in every control cycle without constructing a global map. The potential field algorithm [14] and The Bug algorithms [15] are well-known examples of the local navigation methods. These algorithms assume that the mobile robot has detailed knowledge of the start and target locations, and directions so that it can find an optimal path between these two locations and avoid obstacles. Generally, these methods are easy to construct and optimal for real-time applications.

In indoor environments, there are many aspects that can be used as a reference for the robot’s motion, such as walls and doors. For example, Zhichao [16] explained an algorithm that detects door features, such as color, texture and intensity edges from an image. The extracted door information was used as a landmark for indoor mobile robot navigation. Murali’s [17] robot always performed straight line navigation in the center of a corridor by keeping a ceiling light in the middle of the image but this greatly restricted its motion.

The walls are particularly important when designing a navigation system that enables the mobile robot to work autonomously in different indoor environments. Pieces of furniture on the floor of the environment may obstruct access to the target object and need to be considered when designing the navigation system. In this study, the robot uses a local method similar to the Bug formulations [15]. It employs its sensor (vision and range) system to obtain the local environmental information. However, the robot has extra information about its surrounding terrain such as: walls defined the environment; and all obstacles are located close to the walls.

The mobile robot’s object recognition tools must deal with 3D models in real-time. Specifically, the robot requires the ability to detect the target object from different sides, distances and rotation angles. The appearance of an object in an image varies from one viewpoint to the next. Variations due to environmental conditions, target characteristics and sensor efficiency also complicate the object recognition. Lighting, texture and background color are the major relevant environmental conditions while the key target characteristics involve texture and contrast features. Various methods can be used in order to detect the object in the image such as color segmentation [18], template matching [19] and speeded up robust features (SURF) [20]; however, they are severely limited by their need for training data and sophisticated algorithms [19].

Visual tracking is a crucial research area because it is involved in many robot applications such as navigation and visual surveillance [21]. It consists of capturing an image by a camera, detecting a goal object in the image by image processing and guiding the robot automatically to track the detected object [22]. For indoor robot navigation, tracking is widely used for service robots [23]. For example, the service robot used by Abdellatif [23] tracked by following a colored target. Color segmentation was applied to recognize the object and then the target’s location was determined. A camera with three range sensors was used to detect obstacles and target distances. The camera and range sensors outputs were used as inputs.
for a controller, which enabled the mobile robot to follow the object while avoiding obstacles. Abdellatif’s work was limited to using a single color for target detection. Furthermore, there was no option available to the robot if the object was not detected in the current view.

This paper presents a vision-based technology for a robot that executes the visual-search for a target object. If it is found, the robot will implement visual tracking to approach the target object before grasping it. When this is completed, the robot will again execute the visual search; however, this time, it will search for the target location where the robot will release the object. If the robot returns to the start location, which is defined by a visual landmark, without finding the target object, the search process has been accomplished. First, the measurement module is described. This is followed by explanation of the navigation system. Finally, the experimental results are presented.

2. MEASUREMENT MODULE

2.1 Sensor Configuration

The hexapod has been described in [1]. Figure 1 illustrates the movement control system of the hexapod. The robot starts to read and process the sensor data using the main controller to obtain the environmental information. This information is then sent to the motion planning and navigation algorithm. Motion planning divides the total displacement that the robot wants to achieve into smaller sections. The maximum distance of one walking step is 7 cm and thus, each section can be equal to or less than this displacement (in this study, the maximum step size is limited to 5 cm to reduce the probability of legs colliding with each other). The navigation algorithm uses the environmental information within the robot’s motion and determines the robot’s state and location in the environment. Accordingly, the robot calculates the desired total displacement or steering angle needed. The inverse kinematic [24] is used to calculate the desired joints’ angles to manage the movement of the robot. The microcontroller translates the desired joints’ angles into the PWM signals that control the operation of the servo motors.

The robot uses six range sensors and a camera to obtain environmental information for navigation purposes. The two range sensors connected in front of the robot are used to estimate the distance between the robot and the walls or obstacles directly in the robot’s path. They are also used to control the robot when it approaches the target object. On each side of the robot there are two sensors for:
- maintaining a desired distance from a wall;
- correcting errors that come from the robot’s legs or wheels slipping on the floor;
- detecting the boundaries of obstacles to enable navigation around them; and
- detecting the nearest wall when the robot starts to navigate.

The camera enables the robot to detect the target object and identify the final location. The robot is also equipped with seven analogue force sensors; one sensor in each leg and in the gripper. The leg sensors enable the robot to walk in the rough terrain that the robot is designed to cope with while the gripper sensor allows the robot to control the force that is applied to the target when it is grasped and relocated.

2.2 Vision System

As mention before, the robot executes the visual-search for a target object. If it is found, the robot will implement visual tracking to approach the target object before grasping it. When this is completed, the robot will again execute the visual search; however, this time, it will search for the target location where the robot will release the object.

2.2.1 Object Detection by Template Matching

When the robot searches for the object, there are three problems regarding the object’s invariant features:
scaling, rotation and the 3D models of the object (the camera’s viewing angles). These can be solved in a template matching technique by using various template sizes with different possible rotations and object sides [19]. Template matching can be performed by using color or grey scale images; both the template and the original image must have the same format. In the former, the calculation is made for each channel in the image while the latter is carried out in only one channel. Thus, template matching using the grey scale will be faster than carried out in only one channel. Thus, template matching technique by using various camera’s viewing angles). These can be solved in a template matching technique by using various template sizes with different possible rotations and object sides [19]. Template matching can be performed by using color or grey scale images; both the template and the original image must have the same format. In the former, the calculation is made for each channel in the image while the latter is carried out in only one channel. Thus, template matching using the grey scale will be faster than template matching that is explained in [19] was adapted and used to match the 3D object in real-time image I:

1. Using 3D model (object), create 2D object projection templates \( T(w_i, h_i, f_i, d_i) \), where \( w_i, h_i \) is the template dimension in location \( i \), \( f_i \) is the object’s side views \( (f = 1 \text{ to } 4) \) and \( d_i \) is the object distance from the robot (60 to 150 cm in 30 cm steps).
2. Convert the template T and the captured image I to grey scale, if necessary.
3. Find the best match R in I for T using template matching algorithms.
4. Find the center of T in the image and send it to the robot.

In step 1, an infinite number of template images can be created from the 3D object with different distances from the robot. However, if all poses and side projections of the 3D object are taken into consideration, it will be computationally expensive [19]. Therefore, only the four side 2D projections of the object are considered, together with the distance between the robot and the object (60 to 150 cm in 30 cm steps).

The best match correlation can be found, which can then be compared with the correlation threshold. The final step (4) is executed if the best match value is equal or greater than the threshold. In this step, the center of the perfect matching area is determined in the original image. The center of the area represents the desired object’s center; it is \( C(x + w/2, y + h/2) \).

Figure 2 shows the results of detecting and recognizing two objects by using color images; the red rectangles represent the objects that are found in the real-time images.

### 2.2.2 Detecting Object by SURF

The SURF [25] has been successfully adapted, implemented and tested. Figure 3 illustrates the results of detecting and recognizing two objects by using this method. First, the template of an object to be found is taken; then the interesting points and descriptors of the template image are extracted and calculated. The small circles, which represent the SURF interesting points, can be clearly seen in Figures 3B and D. Next, the interesting points and descriptors of the environment (in the real time image) are extracted and determined. Then, the matching process is undertaken by comparing the interesting points and descriptors of both the template and the real-time image. The results of the detection process are shown Figures 3A and C; the green rectangles represent the objects that are found in the real-time images. The number of interesting points, which is the same in both images, is determined. If the number of matching interesting points is equal or greater than a threshold number, then and the average of their horizontal center coordinates in the captured image is calculated and fed to the controller.

#### 2.2.3 The Landmark Design

In order to identify the delivery location in an environment, a mobile robot needs to observe characteristics of this location. Fast determination of the location’s features by the camera is crucial in real-time navigation. Therefore, a cylindrical shaped artificial landmark that has two solid colors (green and blue) was used (see Figure 4A). The main advantage of using the cylindrical shaped landmark is that it appears the same from any side direction. The two-color landmark pattern was chosen because it is less likely to be confused with the background environment and can also provide a more accurate detection process.

Figure 2. Template matching results using a color images; original images are on the left and template images are on the right.

In step 2, both the template and the captured image can be used either as a grey scale or as a color image. The size of the template is \( T(w, h) \) and the captured image is \( I(W, H) \). The size of the resulting image \( R(x, y) \) that holds the correlation number is \( R(W - w + 1, H - h + 1) \). In step 3, template matching algorithms move (by sliding) the patch of \( T(w, h) \) through the \( I(W, H) \), one pixel at a time (left to right, up and down). At each location, the algorithms compare the data of \( T \) to the data of the particular area of I and store the comparative result in R. The algorithms also calculate how “good” or “bad” the match (correlation) is in that location. The existing OpenCV libraries are employed to implement the method.

Figure 3 shows the results of detecting and recognizing two objects by using color images; the red rectangles represent the objects that are found in the real-time images.
2.2.4 Segmentation by RGB

The RGB color segmentation was achieved by using the original 24-bit image directly to segment the desired color. Four rules were used for the task of object detection based on color:

1- \{rl \leq r(u, v) \leq rh\} means that the primary color components (red) should be between the maximum (rh) and the minimum (rl) threshold values.

2- \{gl \leq g(u, v) \leq gh\} means that the primary color components (green) should be between the maximum (gh) and the minimum (gl) threshold values.

3- \{bl \leq b(u, v) \leq bh\} means that the primary color components (blue) should be between the maximum (bh) and the minimum (bl) threshold values.

4- Regarding the object color, choose the absolute of one of these forms: \|r(i,j) - g(i,j)\|, \|r(i,j) - b(i,j)\| \text{ OR } \|b(i,j) - g(i,j)\| \leq \text{Threshold differences}. For instance, if the red object is to be detected, this value will be the greatest absolute difference value between the green and the blue abs \|b(i,j) - g(i,j)\|.

In this method, the segmentation code reads and compares the intensity weight value of each pixel in the input image using the above four rules. Then, the results of the comparison are combined using the AND logical operation. If the pixels satisfy these rules, then the pixels are considered to be the object color. In this case, these pixels are given the maximum value (255). Otherwise, those pixels that do not satisfy the four rules are given the minimum intensity (0). This process converts the color image to a binary image. The maximum and minimum threshold values of the object color weights are determined by using MS Paint. Then, these values are used with the above four rules to write the segmentation code that determines whether each image pixel either follows the object color or not.

2.2.5 Segmentation by HSI

In HSI color space, each color with all its status from darkest to brightest is assigned by a particular period of Hue values. The amount of the original color that is mixed with white color is specified by a Saturation value and the brightness of the color is assigned by Intensity values. Color segmentation based on HSI is done to utilize the object’s color content in the input.
image. The segmentation process is achieved using the following steps:

1. Determine the object color interval of the HSI color space;
2. Convert the image contents from RGB color space to HSI color space; and
3. Apply the segmentation method.

The object color HSI interval is computed by taking an image of the object for which the robot will search. Then, the image is sent to MS Paint in order to assign the object color’s HSI interval. OpenCV is used to convert the RGB image to an HSI image. After converting from the RGB color space to the HSI color space, the H, S and I components of each pixel in the image \( I(u, v) \) are compared with the pre-determined HSI interval, as in the equation below. Next, the results are combined using the AND logical operation to determine whether the pixel follows the object color or not. If all three values of the H, S, and I in that pixel are within the stated HSI ranges, then the pixel is considered to be following the object color. In this case, this pixel is given the maximum value (255). Otherwise, it is given the minimum intensity (0). This process will segment the image and convert it to a binary image \( B(u, v) \):

\[
B(u, v) = \begin{cases} 1 & \text{if } (H \leq H_{\text{low}}(u, v) \leq H_{\text{high}}) \text{ AND } (S \leq S_{\text{low}}(u, v) \leq S_{\text{high}}) \text{ AND } (I \leq I_{\text{low}}(u, v) \leq I_{\text{high}}) \\ 0 & \text{otherwise} \end{cases}
\]

Once the object is detected, its location relative to the robot must be determined. This is done by calculating the features of the object in the image. In this study, two of the object’s features are needed: the area and centroid of the object in the image (see sections 2.2.3 and 3.2). These are determined by using the technique proposed by [26, 27]. After the area and center coordinates of an object in the image is calculated, the resulting values are then forwarded as input signals to the robot controller. Note that in both RGB and HSI color segmentation methods, some image noise might appear in the binary image background because of the segmentation process; therefore, this noise must be removed or reduced. This was done using the morphological opening operation, which is performed by eroding the image and then dilating it (see [28]). This operator was used because it eliminates all small noises and keeps the shape and area of the target object in the image.

3. THE NAVIGATION SYSTEM

The navigation system relies on the code that controls the robot’s motion. The system gathers sensor information in real-time and translates it into control commands. In the proposed navigation system, the robot starts its motion by searching the environment in the start point with the objective of finding the target object. If the robot does not find the target, it will navigate and explore its environment during its motion. Figure 9 depicts the mobile robot navigation strategy in which the robot keeps the wall on its left when moving forward. Any obstacle that exists in the robot’s path will be considered as a structure that is similar to the wall and the robot turns right and navigates along this obstacle. While the robot navigates, it continues searching for the target and correcting its location beside the wall.

Figure 5 represents the flowchart segments of the navigation process of the hexapod mobile robot. First, the robot executes searching mode (1), which enables the robot to search 360° in the current field (see section 3.1). If the target object is found in this process, then time and effort included in the searching process will be saved. However, if the robot does not find the object, it will move to the nearest wall (or similar obstacle). The nearest wall can be found either by the vision system (as such, it is defined by a specific artificial landmark), or by the ultrasonic range sensors. When the robot reaches the pre-defined distance (1) from the wall (which is defined by the user and it must enable the robot to rotate without colliding with the walls or obstacles), it will turn 90° to the right and increase the termination counter (counter (1)), which helps the robot decide when the searching process is complete (i.e., when it has covered the whole searching area). The robot will compare the current counter value with the pre-defined value that has been decided by the user. If it is equal to or greater than the pre-defined value, the robot will terminate the navigation and the search process. Otherwise, the robot will position itself beside a wall and continue the search.

The locating process is initiated by reading the front ultrasonic range sensors’ information. If the distance between the robot and the wall or the obstacle is equal to or less than the maximum pre-defined value, this means there is not enough free space for the robot to navigate and correct its location beside the wall. In this case, the robot executes searching mode (3), which enables the robot to find out that either the encountered obstacle is the starting point or not. If so, the robot terminates its motion; otherwise, it turns 90° to the right. If there is enough space to correct the location, the robot will read the information from both left-side range sensors. If both measurements are equal to or less than the maximum pre-defined value (2) of the distance interval (which specifies the largest distance from the adjacent wall and it also specified by the user), the robot will locate itself beside the wall (see section 3.4) and then move a specific distance before resuming the search process. The hexapod robot walks six steps (30 cm) and then stops to update the sensors’ information. A counter (2) is added to the program to specify the distance that the robot walks before searching for the target object (performing searching mode (2)) as will be explained in section 4.
Vision-Based Relocation of Objects Using a Mobile Robot

Approach the pre-defined wall

Turn right 90° and add one into the counter (1)

Is the value of counter (1) equal to terminator?

Yes

Stop navigation

No

Searching mode (4)

Read the front range sensor's information

Searching mode (3)

Is the distance from the next wall or the obstacle ≤ the pre-defined distance (1)?

Yes

B

D

No

Read the side range sensors' information

Searching mode (2)

Are both measurements ≤ the pre-defined distance (2)?

Yes

Locate the robot beside the wall

Walk specific distance and add one into the counter (2)

Is the value of counter (2) equal to pre-defined value?

No

A

B

Yes

Initiate counter (2)
However, if both left-side range sensor measurements are greater than the maximum pre-defined distance value (2), this will mean that the robot moves beside the obstacle, having reached the boundaries of this obstacle. In this case, the robot turns 90° to the left in order to keep the obstacle on its left side. Then, it executes searching mode (5), which enables the robot to finds out that either it encounters entrance (door) or not (see section 3.1). If so, the robot turns 90° to the right and moves beside that entrance. Otherwise, it keeps walking and reading the sensor measurements until it detects the boundaries of the obstacle again. This enables the robot to navigate around the obstacle. In this process, the robot keeps searching the environment for the target object as explained before. However, if one of the left-side range sensor measurements is greater than the maximum pre-defined distance value (2), the robot seeks to detect the boundaries of the obstacle and therefore, it walks a specific distance and then checks the sensors’ information.
The robot navigation process continues until either the robot finds the target, or the counter value reaches the pre-defined termination value. The robot also terminates the searching process if the start point is encountered. If the robot finds the target object, it will move to approach it, then grasp and relocate it to a new location.

### 3.1 Searching Mode

Searching mode means that the robot uses its vision system to find a target that is a known object in an unknown 3D environment; this task requires the robot to control the camera’s action. This process includes two stages: ‘where to look next’ and ‘where to move next’ [29]. In this study, both robots are equipped with cameras that do not have zoom capabilities. Therefore, the target’s size in the image depends on its distance from the robots.

In the first stage, the robot stops its movement and fixes the camera in a current position (viewpoint) for a length of time sufficient for acquiring and processing the image and updating the robot’s knowledge about the environment. If the target object is not detected in the current viewpoint, the robot executes the second stage in which it moves the camera to the next optimal viewpoint. The next viewpoint should be reachable with high probability of detecting object. It should also bring other hidden search areas into the camera’s view [29].

The selection of the next viewpoint includes choosing the position and direction of the camera (the new sight angle) relative to the previous viewpoint of the camera. The view angle of the vision sensor plays an important role in the search process because it decides the area that will be searched each time. It also decides the maximum pan angle that must be used to rotate the camera each time. For instance, the blue object that is shown in Figure 6A is not detected and recognized by the vision system because portions of the object area are hidden from the camera’s view. In this case, if the camera is rotated by an angle that is the same as the sensor’s sight angle, the same problem will continue to appear, as shown in Figure 6B. Moreover, if the pan angle is greater than the sensor’s vision angle, even the object that is on the area between the new field (cyan area) and the previous one (yellow area) will be ignored, as shown in Figure 6C. Therefore, in order to solve this problem the camera is rotated by pan angles that are less than the sensor’s view angle, as shown in Figure 6D.

The probability of finding the object in the field increases if the robot searches each single area from two different viewpoints. This is because the camera might be rotated to a viewpoint in which the camera acquires the object’s image with higher contrast features. In this research, the sight angle of the vision sensor used was 36°. The decision taken was to use a 20° pan angle, which is almost equal to half of the vision angle and satisfies all the above conditions.

The hexapod mobile robot has a fixed camera; therefore, the pan rotation is provided by revolving the robot about its central axis. As such, the searching time is increased by the duration of the robot’s rotation.

![Figure 6. Viewpoints of the camera: (A) the state when the blue object is on the boundary of the camera’s view area (part of the object cannot be seen by the robot); (B) the camera is rotated to the new viewpoint (cyan area) by an angle that is the same as the camera’s old viewpoint (yellow area); (C) the state when the camera is rotated by a pan angle greater than the camera’s sight angle; and (D) shows the desired state when the pan angle is less than the camera’s sight angle](Image)

The hexapod robot has five searching modes. First, searching mode (1) is used to search for the object in the starting location. If the target is not detected in the first image taken by camera, the robot turns 20° clockwise about its central axis and another image is taken. This process is repeated until the target is detected. After a full rotation of 360°, if the target has not been detected, this loop is terminated and the robot starts the navigation process.

Second, the searching mode (2) is used when the robot is navigating, i.e., it locates the wall at one of its sides; therefore, the robot needs to search the space that is directly ahead and also the space that is inside the environment (the middle of the room). In this case, the robot rotates the camera 180° instead of the 360° that is used in mode (1). In both modes, the robot has been programmed to execute the object approaching function once the target is found, or to continue the search process. Third, the searching mode (3) is employed to terminate the robot motion if the searching process is completed without finding the target. In this case, the robot finds out that either the encountered obstacle is the start location or not? If
so, the robot terminates its motion; otherwise, it continues the navigation process. The searching mode (4) is used to search the area, which is directly ahead, for the target object while the robot moves forward. Finally, the searching mode (5) is employed within the navigation system in order to enable the robot to find out that either it encounters the entrance of the environment or not. If so, the robot steers 90° to the right and moves beside that entrance.

3.2 Approaching the Object

When the target object is detected, the main task for the robot is to find out the target’s location and then execute a specific duty. In this study, the robot must grasp and relocate the target to the delivery location. In this scenario, the robot must approach the object by keeping its image within the center of the image plane [23]. As such, the object’s center (horizontal coordinate) in the image is determined and forwarded as input to the robot controller. In this process, the captured image is divided into three areas (see Figure 7); consequently, there are three logical cases that can be fed to controller. The first is that the object’s center is on the right of the middle area (green area); therefore, the robot has to rotate to the right using a specific rotation angle, dependent on the amount of the error value, in order to correct the error and bring the object’s center to the middle area. The second is that, if the center is on the left side, the robot will rotate to the left side and the third, if the center is in the middle, the robot moves forward to approach the object.

![Figure 7. The desired area (green area) location in the image](image)

During the approach process, the robot controller continuously updates the position of the robot relative to the object by using the information coming from the front ultrasonic sensors. When the robot reaches a pre-determined distance from the object, which enables the robot to grasp it, the robot does this. In the grasping action, the robot opens its gripper, moves it down to place the object in an appropriate grip point (in the middle of the object), closes the gripper to grasp the object firmly, and then moves it up a desired distance. As mentioned in section 2, a force sensor was attached in the gripper to detect the amount of the grip force that is applied to the object. This sensor is employed as on/off switch. If the force exceeds the pre-defined maximum value, which depends on the grasped object’s weight, the robot will stop the gripper servo motors’ shafts in the current locations. Once the grasping operation is accomplished, the robot then executes the relocating process (as will be explained in the next section). If the robot loses the object in the image during the approaching process, it will again carry out the searching mode (1).

3.3 Relocating Process

Once the target has been approached and grasped, the robot starts searching for the delivery location that is defined by a landmark (see section 2.2.3). As such, the robot performs the same codes that were used initially to find the target object, but this time it searches for the delivery location. At the location where the object is grasped, the robot searches the environment for the delivery location by performing the searching mode (1). If the location is found, the robot will approach it and deliver the object. In this case, the robot carries out the approaching function except that the image processing is performed to find the location. When the robot reaches the delivery location, it moves its gripper to a point above the delivery location surface. Then, the gripper is opened and moved up to release the object on the surface. However, if the location is not detected in the previous process, the robot will carry out the navigation and searching functions as explained in section 3.

3.4 Locating the Robot beside the Wall

When the robot follows the walls or obstacles boundaries, it must move within a desired distance from these objects to avoid collision. When the robot wants to walk along the wall, it starts ranging the distance between its current location and the wall. If one or both of the range sensors’ measurements is greater than the maximum pre-defined interval distance (2), the correction loop is ended. Otherwise, the loop is continued by comparing both measurements. If both measurements are not equal, the robot will be either moving away from the wall or risking collision with the wall. Therefore, the robot needs to correct its direction to be parallel to the wall. For instance, if the side front sensor measurement is greater than the rear side sensor measurement, the robot will correct its direction by rotating a specific angle to the left. This process will be continued until it reaches the minimum difference between the two sensors’ measurements, which is specified by the user to enable the robot to move within a minimum error. This process helps the robot to move along the wall in a straight line.
4. EXPERIMENTAL RESULTS

4.1 The Image Processing

The proposed algorithms for image processing were implemented using C++ and some libraries were used, including OpenCV [30] and OpenSURF [25]. They were initially tested by using a PC with a 1.7 GHz microprocessor and 1 GB of RAM. A Linux operating system was installed in the computer. A webcam, model C200 was used as the vision sensor. This takes images with a maximum resolution of 640 by 480 pixels with an image capture rate of 30 frames per second. The camera was connected to the PC via a USB port.

The experiments were done off-line and then in real-time. In the former, environmental images were taken and processed to extract the object’s features. In this scenario, it was possible to control the environmental factors, such as light intensity and background colors. Therefore, optimal results were obtained in these experiments. However, the robot executes its tasks in dynamic environment; therefore, the object detection algorithms must be evaluated under these conditions. Thus, the object’s features were extracted from the images captured in real-time.

![Coca Cola detection by the color segmentation technique](image)

Figure 8. The Coca Cola detection by the color segmentation technique: (A) the original image; (B) the image segmentation using RGB; and (C) the image segmentation using HSI color space

In order to evaluate the presented algorithms, a Coca Cola can, which is 6 cm in radius and 13 cm in height, was used and it was located at 60 cm from the camera (see Figure 8A). First, the color segmentation techniques were tested to determine the processing time, and the can’s area and its horizontal location in the image, as shown in Figures 8B and C. Table 1 illustrates the experimental results. Then, the template matching and SURF methods were evaluated (see Table 2). It is important to note that the time was determined after processing ten frames while the area, center and match of the can were calculated for the last frame.

<table>
<thead>
<tr>
<th>Item</th>
<th>RGB</th>
<th>HSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>1.653</td>
<td>1.901</td>
</tr>
<tr>
<td>Area (pixel)</td>
<td>7527</td>
<td>8297</td>
</tr>
<tr>
<td>Centre, X axis (pixel)</td>
<td>340</td>
<td>341</td>
</tr>
</tbody>
</table>

Table 1. Experimental results for the color segmentation methods by using the PC. HSI was more efficient than RGB (the detected object’s details in the image have a high quality); however, the processing time was longer. The center horizontal coordinate (X axis) was slightly different because the numbers of pixels in object area were different.

<table>
<thead>
<tr>
<th>Item</th>
<th>SURF</th>
<th>Template matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>4.244</td>
<td>30.171</td>
</tr>
<tr>
<td>Match (matched points for SURF and correlation for template matching)</td>
<td>8</td>
<td>0.785</td>
</tr>
<tr>
<td>Centre, horizontal coordinate, X axis (pixel)</td>
<td>339</td>
<td>337</td>
</tr>
</tbody>
</table>

Table 2. Experimental results for template matching and SURF (the match was determined by the number of the matched points that exist between the template and the captured image in SURF method while it was calculated by the correlation number in the template matching). Note that the time is for processing ten frames while the robot only needs to process one frame; therefore, the processing time will be reduced to 0.4 and 3 seconds for the SURF and template matching, respectively.

From the experimental results, it was concluded that:

- In both RGB and HSI color segmentation methods, some image noise might appear in the binary image background because of the segmentation process; therefore, this noise must be removed or reduced. This was done using some morphology operations see section 2.2.5.

- The HSI technique was more effective than the RGB color space for finding an object by its color (see Figure 8). This is because each uniform color in HSI, from the darkest to the brightest, is assigned a particular period of Hue values, whereas the Saturation and Intensity periods specify only the amount and brightness of the color, respectively. It was also concluded that the HSI technique was less sensitive to changes in light intensities in agreement with [23]. However, the segmentation process in HSI took longer than in RGB (see Table 1). This was because the output of the camera was in the RGB color space and thus, the color space transformation between
RGB and HSI incurs computational cost, which affects real-time behavior [31].

- When a target object with a single color is used, optimal results are achieved by using the color segmentation method. In this case, the segmentation techniques are ideal for detecting the target from any viewpoint because the object is specified by its uniform color. The detection task depends on the number of pixels that follow the object pixels in the image and this is not affected by the object’s rotation. However, the number of object’s pixels is influenced by the object’s distance from the camera. In this case, if the object is close to the camera, it will cover a large area in the image. Conversely, if it is far away from the camera, it appears in the image as only a few pixels.

- The background colors of the environment affected the segmentation process.

- The template matching and SURF methods achieved better results when multiple-colored objects were used. In this case, an accurate matching process is attained because the templates that represent the objects’ images have the highly detailed and unique regions [32]. The SURF method and specifically the associated calculation process, was quicker because only one object template was used, instead of the 16 used in template matching (see Table 2).

The image processing algorithms were then implemented on the real robot for validation. The hexapod has a computer-based Roboard controller RB-100 that has a Vortex86DX and a 32 bit x 86 CPU running at 1 GHz with 256 MB on-board memory. The main control program and all image processing functions were implemented on-board. The main drawback was that this board has a processor with a low computation capacity. Table 3 shows that although the PC’s processor used is 1.7 times faster than the Roboard processor, the image processing time within PC was approximately four times shorter than that determined within the hexapod.

<table>
<thead>
<tr>
<th>Item</th>
<th>RGB</th>
<th>HSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>7.105</td>
<td>7.315</td>
</tr>
<tr>
<td>Area (pixel)</td>
<td>7595</td>
<td>7784</td>
</tr>
<tr>
<td>Centre, X axis (pixel)</td>
<td>332</td>
<td>333</td>
</tr>
</tbody>
</table>

### 4.2 The Navigation System

The experiments have been conducted on the hexapod. The main control program and all image processing algorithms were written using C++. The experimental environment was the office room with a total area of 3200 × 4000 mm. Figure 9 represents the environmental map of the room. The brown rectangle represents the robot and the green rectangle denotes the starting point (delivery location) to which the robot must relocate the target object. The letter scripts (in Figures 9A and B) identify the locations of the robot in Figures 10 and 11, respectively, and the green arrows represent the robot’s navigation path.

The experiments were carried out to evaluate the hexapod’s control system. First, the system was tested without the robot finding the target object. The objective of the robot was to navigate along the walls, starting from the starting point and to return to this location, avoiding collisions with obstacles in its path. The obstacles were rectangular shaped objects of various sizes (see Figure 9A).

In the initial location, the robot started searching for the target object by executing searching mode 1. When the object was not found, the robot started moving from its current location and approached the nearest wall (see Figure 10A). After approaching the wall, the robot turned in the desired direction (for example, turned right). When the robot reached the first obstacle, it viewed this obstacle as a wall, turned right and navigated along it. The robot continued moving until it reached the boundaries of the obstacle. In this case, the robot turned left, in order to keep the obstacle on its left side. The robot kept looking for the object while it moved forward by performing searching modes 2 and 4. These processes were repeated for other obstacles until the robot went...
back to the wall (see Figures 10B and C). The robot continued moving along the walls or obstacles until it encountered the start location (delivery location), where the robot terminated its motion (see Figure 10D).

Two heuristics, which are characterized by selecting the points that the robot must stop and search its surrounding environment [11], were used to examine the relative effect of varying search and travel cost (the operating time needed). In the first heuristic, the robot travels 1.2 m, which is specified by counter (2), and then stops to execute searching mode (2) to search the environment for the target. In the second heuristic, the robot travels 1.5 m before performing searching mode (2). Table 4 describes the results of both searching heuristics.

The hexapod walked six steps (30 cm) forward and then stopped to update the sensory information, which consists of the ultrasonic sensors and camera information (performing searching mode 4). In this study, the robot required 12.8 s for travelling (30 cm) and 2.9 s for updating the environmental information. In the first heuristic, the robot repeated this process four times to travel 1.2 m so it required 62.8 s (51.2 s travelling and 11.6 s updating sensors’ information) to perform this task. Then, the robot stopped to execute searching mode (2), which required 84 s (26.1 s updating camera information and 57.9 s rotating the camera) to be accomplished. The robot navigated in the previous environment (Figure 9A) 10.8 m to complete searching the whole area. In this scenario, the search process consumed 67% of the operating time. This involves the time of rotating the robot’s body (pan the camera) (779 s) and image processing (365.4 s). The travelling time represent of 33% of the total time and it only consists of moving forward and updating the sensory information. In the second heuristic, the robot travelled 1.5 m without performing searching mode (4); as such it only required 64 s. Then, it carried out searching mode (2). Although the travelling time was staying at 460.8 s, it increased to 44% of the operating time. This was because the robot travelled the same distance; however, it consumed less time within the search process.

In both heuristics, the robot was extremely slow because it had a fixed camera so that it performed searching modes by rotating itself about its central axis. This contributed with the image processing and calculations of the robot’s gait signals for making the navigation process extremely slow. The travelling time will dramatically increase if the robot travels in an environment that has more obstacles. The experimental results also demonstrated that with the decrease of the number of the heuristic’s points that the robot stops and searches the surrounding area, the searching cost was reduced from 67% to 56% (see Table 4). However, this might reduce the probability of finding the target object. Therefore, the search process must be carefully planned, which means the heuristic should have a smallest number of points that can cover the entire environment.

<table>
<thead>
<tr>
<th>Item</th>
<th>Heuristic 1</th>
<th>Heuristic 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searching time (s)</td>
<td>944.40</td>
<td>588.00</td>
</tr>
<tr>
<td>Travelling time (s)</td>
<td>460.80</td>
<td>460.80</td>
</tr>
<tr>
<td>Total time (s)</td>
<td>1405.20</td>
<td>1048.80</td>
</tr>
<tr>
<td>Searching cost (%)</td>
<td>67</td>
<td>56</td>
</tr>
<tr>
<td>Travelling cost (%)</td>
<td>33</td>
<td>44</td>
</tr>
</tbody>
</table>

Then, the control system was tested by the robot finding and relocating the desired object (see Figure 9B). In this scenario, the robot executed the above
process until it detected and recognized the target object. In this case, the robot abandoned the wall and moved towards the target. In doing so, it kept the image of the target within the image plane. When the robot reached the pre-determined distance from object, it grasped it and put it in the delivery location (see Figures 11A-D). A specific artificial landmark (as explained in previous sections) defined the delivery location. The system was executed successfully, which proves the effectiveness of the methodology used.

Figure 12 illustrates the hexapod motion in an environment where the target is not placed close to the wall (here the target cannot be seen from the starting point because it is covered by the obstacle Figure 12A). Therefore, the robot moved along the walls and continued executing the searching modes (as explained before) until it could find the object. In this case, the robot left the wall and approached the object to grasp it (Figure 12B). Then, the robot performed searching mode (1) in order to find the delivery location; and because this location could not be seen from the current location, the robot moved to the closest wall (in this case the obstacle). Then, it followed the obstacle’s boundaries until it detected the delivery location (Figures 12B to E). The robot then approached this location to put the object there (Figure 12F). More environmental states are given in [1].

5. CONCLUSIONS

In this work the navigation system that enabled the mobile robot to search and find the object in the environment has been implemented and tested within a hexapod mobile robot. The hexapod had inherent hardware problems and limitations that also affected its performance. First, it belongs to the legged robots that are inherent to be slower than the wheeled ones. Second, it had a fixed camera so that it performed searching modes by rotating itself about its central axis. This contributed to making the navigation process extremely slow.

It was concluded that if the number of the heuristic’s points that the robot must stop and search the surrounding area is reduced, the searching cost is dramatically decreased. However, this reduces the probability of finding the target object. Therefore, the search process must be carefully planned which means the heuristic should have a minimum number of points that can cover the entire environment. It was also concluded that it is possible to implement a navigation system within a smallest number of sensors if they are positioned and used effectively on the robot’s body. Experiments proved that the methodologies used and that the codes developed made the robots capable of performing their tasks of finding, approaching and relocating objects as planned. As future work, it will be important to use a processor that has high computation capacity within the hexapod and then implement the same functions used. It is also important to implement an intelligent controller for the robot such as a Fuzzy controller to perform the mentioned tasks.
REFERENCES


