ISSN: 0067-2904

Robotic System for Tracking Moving Objects Based On Their Color

Sajjad H. Hendi*, Faisel G. Mohammed
Department of Computers, College of Science, Baghdad University, Baghdad, Iraq

Abstract

Tracking moving objects is one of a very important applications within the computer vision. The goal of object tracking is segmenting a region of interest from a video scene and keeping track of its motion and positioning. Track moving objects used in many applications such as video surveillance, robot vision, and traffic monitoring, and animation. In this paper a four-wheeled robotic system have been designed and implemented by using Arduino-Uno microcontroller. Also a useful algorithms have been developed to detect and track objects in real time. The main proposed algorithm based on the kernel based color algorithm and some geometric properties to tracking color object. Robotic system is a compromise of two principal parts which are the hardware and software parts. Hardware includes Bluetooth model to connect the phone with Arduino-Uno. Robot body consists the L298 dual H-bridge motor driver to drive four geared motor, two battery as a power supply and two servomotors to move the camera in both horizontal and vertical axis. Software is responsible for making the right decision based on the analysis of data that receives from the digital camera. Color-based tracking algorithm and border following algorithm used to detect the location of the target object in the images have been showed in the paper. All computations are accomplished within android device. Through applying the object tracking method, several parameters have been considered like frame rate, motor period time and speed of target object. All experiments were in the real environment. The proposed robotic system succeeded to track the target object with a success rate up to 97% in indoor environment.

Keywords: Robotic, Android, Arduino, Tracking, Real time.

نظام روبوتي للتتبع الكائنات المتحركة اعتماداً على لونها

سجاد هيثم هندي*، فيصل غازي محمد
قسم الحاسبات، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة

في هذا البحث تم تصميم نظام روبوت رباعي الدفع باستخدام متحكم اردوينو أونو.كما تم تطوير خوارزميات مفيدة لكشف وتعقب الكائنات المتحركة في الوقت الحقيقي اعتماداً على لون الكائن وبعض الخصائص الهندسية للكائن. استخدم في هذا البحث هاتف ذكي يعمل بنظام الاندرويد كمنصة برمجية لتنفيذ الإجراءات المقترحة. يشمل النظام جزئين رئيسيين هما : جزء الأجهزة وجزء البرمجيات. جزء الاجهزة يشمل أجهزة الاتصال والحركة ومزودات الطاقة. جزء البرمجيات هو المسؤول عن اتخاذ القرار المناسب بناء على تحليل الصور التي يتلقاها كاميرا الهاتف الذكي. تم استخدام خوارزمية تتبع الكائنات اعتمادا على لونها و خوارزمية تتبع الأطر للكشف عن مكان وجود الكائن في الصور المتعاقبة. يتم إنجاز جميع العمليات الحسابية داخل الهاتف الذكي.خلال تطبيق الخوارزمية المقترحة تم الاخذ بنظر الاعتبار مجموعة من المعلمات المهمة مثل معدل الاطر الملتقطة والمعالجة في كل ثانية و سرعة المحركات وسرعة الكائن الهدف. جميع التجارب تم

*Email: sajjadhendy@scbaghdad.edu.iq

تنفيذها في بيئة حقيقة واظهرت النتائج ان النظام المقترح نجح في تتبع الكائن الهدف بنسبة نجاح تصل الى %97.

1- Introduction

Robots are machines designed to perform a particular tasks repeatedly with a particular speed and accuracy. Usually is guided by a computer program or electronic circuitry, there are many types of robots are classified by the work being done [1]. Build a robot with human-like intelligence and ability is one of the most important challenges facing the development of the robot. Although the human brain and body did not close at perfection, but they represent the basic model for the developers robots as well as for its users. Therefore, it is natural to find robots and humans share many key characteristics as the use of associative memory and rely on visual information [2]. In the last four decades, there has been considerable activity in the field of robotics; both in terms of scientific research as well as in terms of tightening public attention to their large ability and that seem unlimited. In this period included the technological maturity of the robot, robots development in this period from just a simple pick, place, painting and welding robot to more sophisticated assembly robot, introduced integrated circuit chips in its industry, and then became mobile robot, such as mobile cars [3]. Mobile robots are robots that have ability to move around on legs, tracks or wheels. Researches in the field of mobile robotics concerned with at least a two directions: first direction the robot mobility, which is interested in the movement of the robot, ability to move within a particular field. Second direction is independent movement of the robot where it has the ability to navigate through the environment to reach the goal. In this direction, the robot must have appropriate sensors. Vision system can be provide most information about the environment to the robot but also needs to process that complex information [4]. Autonomous robots are mobile robots which can perform desired tasks in unstructured environments without continuous human guidance [5]. In 2005, B. Browning and M. Veloso developed a technique for fast color vision algorithm for object recognition that is suitable for use in robot platforms where lighting levels may vary. Where it is able to adapt its segmentation process to different lighting conditions, within reason, basis of this approach resides in the soft labeling of pixels with an adaptive threshold technique [6]. In 2008, Islam, M. Z., Oh, C. M., and Lee, developed color based particle filter. In this approach, the object tracking system relied on the deterministic search of window, whose color content matched a reference histogram model. A simple HSV histogram-based color model was used to develop observation system. They described an approach for moving object tracking with particle filter by shape information [7]. In 2013, M. Crneković and etal, Presented a real time objects recognition by their color when they used HSV color space and implemented on eMIR mobile robot that had the infrared rangefinders and camera. Where the vision system gave the robot ability to finding the object that had a particular color to be detected and move closer to this object within a certain distance. Process of object recognition applied only around the area where the object was detected in the previous recognition step [4]. In 2015, M. Mohanty and S. Rout, present a motion control method for mobile robots based on color object detection in indoor environments. Used camera attached to a robot to provide an information about the environment to the robot and focuses on color as the primary discriminating feature. Images send from a camera by USB connection to computer, which are processed and extracted the movement commands.

The task to track the moving objects involves many challenges and difficulties that may facing researchers in this area such as detect the moving object and isolate it from the background. In some cases, some important changes affect the robotic system arise from the object is movement like change its shape (i.e., zooming).Unstable lighting condition is one of the common problems faced by the object detection process, especially when working in dynamic and unstructured environments. Camera view-point changes leads to change the observed features from different points of view, angles, and distances and under different illumination conditions. The noise in the captured images causes reducing in the reliability and requires additional treatment processes to reduce its effects on the results. Implementation object tracking algorithm using autonomous mobile robot is one of the most important challenges in this domain that have its own sub-problems. Navigation in an unstructured environment includes such problems as obstacle avoidance, avoidance of hazards, such as holes, boulders, or dangerous locations. The ability to move to a desired location and sometimes search an entire region. In a standalone system must provide appropriate portable power supplies where do not effect on the system navigation of mobile robot. In the case that the robot is not connected with an external processing unit such as a host computer, must provide portable processing unit has ability to process the data that have collected by sensors from the surrounding environment. This paper aims to introduce design and implementation to an autonomous mobile robot system. This system is based on using four wheels locomotion methods that have ability to perform real time moving detection and tracking. This will achieve in non-controlled environment based on their color and some geometric properties of objects with low computational cost and has the ability to avoid any barriers ahead. Carry out the task of video capturing using a five mega pixel digital camera of android smart phone and the image processing task accomplishes by the built-in processor of android smart phone device in order to help a mobile robot system to fully perform the tasks independently without relying on any external host. Attempting to guarantee the necessary power provider to run robot using portable rechargeable batteries.

2- Proposed Robot System

The proposed system is compromises of two principal parts are the hardware part and software part.

2.1 Hardware Equipment

Hardware consists of different devices and tools which have a specific functions that are collected together to achieve the main goal of the design of the robot. Table-1 shows all hardware components that using in the current robot tracking system.

Table 1- Hardware components of current tracking robot system

S	Device or Tool	N	Main Components
1	acrylic glass plate 100x213x5 mm	1	Robot body
2	DC Geared motor	4	
3	Plastic Tire Wheel	4	
4	Motor fixing	4	
5	L298N motor driver	1	
6	Shield V5.0 Sensor Expansion Board	1	
7	18650-battery holder	1	
8	18650 battery	2	
9	servo motor 180°	2	
10	Bluetooth adapter HC6	1	Connection device
11	Arduino UNO328 controller board	1	Microcontroller
12	Samsung Galaxy Core GT-I8262 (Android device)	1	Visual sensor and Image processing device

2.1.1 Robot body

Robot body consists the following:

A - Geared motors: Motors are used to move the body of the robot in various directions.TT geared motor is dedicated to operate with Arduino-Uno and can be connected with it without interface. Operating voltage (3-12) volts, the speed based on the operating voltage. The motor have only two pins one to supplying voltage and other to ground pin.

B –Power supply: Autonomous robots need a separate power source which provides a necessary to operate the system of the robot components and achieve the required tasks. Two dry batteries have been used in current project to fulfill hardware electricity requirements. It is placed directly on the robot's body.

C - L298 Dual H-Bridge Motor Driver: The DC Motors are widely used in many devices, for example, home printers. The DC motor rotation speed depends on the amount of voltage that pass through it. As well as, change the rotation direction by reversing the voltages that passes through the DC motor. H-bridge is a convenient and useful module for controlling DC motors.

D-Servo motors: To move camera to the four directions (left, right, up and down), two servomotors are required one for horizontal movement while another for vertical movement. Both the motors are connected to Arduino-Uno and to the power source.

2.1.2 Microcontroller

The Arduino Uno is a microcontroller board based on the ATmega328 .It has 14 digital input/output pins a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything necessary to support the microcontroller; simply it connects to a computer with a USB cable or it supplies by the power with an AC-to-DC adapter or battery to start working. In this project, the android device is considered as input unit to Arduino-Uno and connects with it wirelessly. Output unit includes L298N motor driver and two servomotors. Arduino comprises of both a physical programmable circuit board and Integrated Development Environment IDE that runs on many different operating systems. IDE is used to write and upload code to the physical board. It uses USB cable to upload the new code to the microcontroller. Additionally, the Arduino IDE uses a simplified version of C++ that making it easier to write codes.

2.1.3 Image processing device

Decision-making depends on the analysis of the received data from the visual sensor. The type of data that received are images. Operations of image processing and analysis for real time application may be too complex for Arduino-Uno. Therefore, image processing device have been used to carry out these operations. The current robot system is not connected to any external server or processor. Therefore, it is necessary to find image processing device can carry by the robot and move without affecting the safety of the movement of the robot. Using processor of Samsung Galaxy Core GT-I8262 as image processing device which can satisfy the pervious requirements mentioned. The results of image analysis are commands for movement. These commands transferred from the image processing device to Arduino-Uno wirelessly via Bluetooth.

2.1.4 Visual sensor

Used five-megapixel of Samsung Galaxy Core GT-I8262 as visual senser, It instell on the robot and carry out by servomotors and capable of taking pictures with Resolution 2592×1944, can take 30 frames per second as maximum, and can reduce the frame rate as needed. Image processing device and Visual sensor is same device.

2.1.5 Bluetooth Set

The Bluetooth is a communication set that provides a data transform for short distance about of 1-100m with low power consumption of 3.3-5v. It is commonly used between carrier devices and peripherals. Bluetooth HC-06 Module used in the current project to send guidance commands from Android device to Arduino-Uno microcontroller.

2.2Software Part

Software is the responsible part for making the right decision whether stopping or moving based on the analysis of data that received from the visual sensor. In general, using five-megapixel digital camera of smart phone with android operating system as visual sensor to monitor moving objects depending on their color and some geometric properties. Both computations of the image processing algorithm processes and issue of guidance commands are computed inside of android device. Object tracking module includes two stags an object selection and an object tracking that explain in details in following two sections:

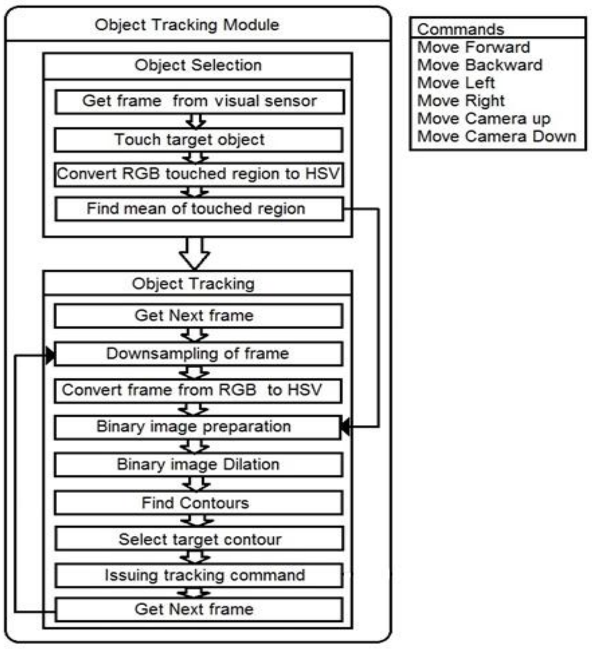

Figure 1- The Software Structure of Proposed Robotic System.

2.2.1 Object Selection

The Initial stage of the Object tracking module is by selecting a particular object to be tracking manually. Object selection includes a number of steps are explain in following points:

-Touch the target object when it appears on the smartphone screen. The result from touching is a single RGB pixel called **Touched Pixel**.

-Touched Pixel is not enough to determine the color of the target object because it is a single RGB pixel. Therefore, a set of pixels with window size 9×9 pixel have been defined to determine the color of the target object called **Touched Region**. The center pixel of Touched Region is the Touched Pixel and its four neighboring pixels in each side represent the remaining elements of the Touched Region as shown in Figure-3.

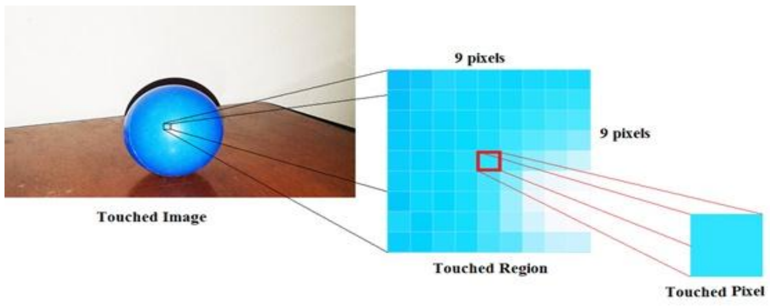

Figure 2- Touched Region.

-Due to the highly correlation of sub pixels in RGB space, the touched region converts from RGB to HSV color space. The result of this conversion is called **HSV Touched Region**.

-The mean will reduce noise in HSV Touched Region and summarize all information of HSV Touched Region through the value of a single HSV pixel which is called **Mean HSV**. Calculate the mean for each HSV component using the following:

$$\text{meanH} = \frac{\sum_{i=0}^{N} H_i}{N} \quad \textbf{(1a)}$$

$$\text{meanS} = \frac{\sum_{i=0}^{N} S_i}{N} \quad \textbf{(1b)}$$

$$\text{meanV} = \frac{\sum_{i=0}^{N} V_i}{N} \quad \textbf{(1c)}$$

Where, N is the number of pixels in touched region, H represents hue value of pixel, S is a saturation value of pixel and V is an intensity value of pixel. These values represent mean of touched pixels.

2.2.2 Color Object Tracking

After the stage of the object selection, the detection and tracking the target object in subsequent frames is the next stage. This step depends on the definition a number of rectangles that reflects the behavior of the movement of the target object. Four rectangles are defined in the present project and explained in the following:

Center Rectangle: The object tracking system always tries to put the object in the center of the frame, during the moving of the camera or the robot by commands. The center of the frame is determined by drawing a rectangle inside the frame and it is called Center Rectangle. The dimensions of the Center Rectangle and its location in the frame are calculated through the following equations:

$$\text{height of Center Rectangle} = \frac{\text{height}}{2} \quad \textbf{(2)}$$

$$\text{width of Center Rectangle} = \frac{\text{width}}{2} \quad \textbf{(3)}$$

$$\text{TLCR} = \text{frame}\left(\frac{x+\text{height}}{4}, \frac{y+\text{width}}{4}\right) \quad \textbf{(4)}$$

Where frame (,) is an image frame, height represents the height of frame, width represents the width of frame, frame (x,y) represents the top left point of frame and TLCR represents the top left point of Center Rectangle.

Object Rectangle :After the object selection process ,the target object will be surrounding by a rectangle which called Object Rectangle. The Object Rectangle is moving inside the frame with the target object wherever it goes. As long as, the object within the center rectangle the robot will not move.

Original Rectangle: At a first frame and after an object frame selection, the Object Rectangle will be stored in rectangle and called Original Rectangle to keep the initial state information of the target object. This operation will perform only one time.

Last Object Rectangle: For each frame that is processed; the target object information is stored through keeping the Object Rectangle that is called a Last Object Rectangle is to keep the previous state information of the target object. Last Object Rectangle is used in the case of losing the target object to determine the direction of target object movement. This allows to search for the missing target object depending on its last appearance of the target object. The final result of the detection and tracking stage is movement commands. There are seven movement commands that move the robot and camera, as describe in the following:

1-Forward: move the robot in the forward direction by speed 8 -20cm / sec for 100ms and carry on until receive another commands.

2-Backward: move the robot in the backward direction by speed 8 -20 cm / sec for 100ms and carry on until receive another commands.

3- Left: the robot will rotate to left and continue until receive another commands.

4-Right: the robot will rotate to right and continue until receive another commands.

5-Stop: stop the movement of a robot because it completing the move task.

6-Camera Up: tilt up the camera in 10 degree.

7-Camera Dawn: tilt down the camera in 10 degree.

The detection and tracking stage includes a set of sequential steps that applied on subsequent frames for the frame that has been selecting the target object through it. These steps are divided into two different groups, one for find the target object by analysis of captured images and the other to issue appropriate movement commands depending on the results received from the first step.

A-Find target object

Captured images are analyzed by a number of steps that carry out separately on each frame to find the right target object. In the following sections, the description of steps for one frame:

Downsamples: The resolution of captured image is 640 x 480 pixels and this resolution is high in order to achieve the process in the appropriate time. Therefore, Resolution of captured image is downsamples using the downsampling step of the Gaussian pyramid construction:

1-Convolves the source image with the Gaussian pyramid kernel

2-Downsamples the image by rejecting even rows and columns. In this project, the image is downsampled twice.

Convert from RGB to HSV: Resulting image after downsample operations is converted from RGB to HSV color space.

Binary image: HSV image is converted to a binary image. A specific threshold value is used in the matched pixels searching operation. Minimum and maximum values will be defined for each three mean components (HSV); Instead of the comparison with a single value for each components of the MeanHSV, the comparison will be with a range of values between the maximum value and the minimum value. The pixel value in the binary image will be **1** if the value of the corresponding pixel in the image lies between the maximum and minimum value. While it is **0** if it is not within this range. As shown in following the equations:

$$\text{HSVMax} = [\text{MeanHSV.H} + 25,\quad \text{MeanHSV.S} + 50,\quad \text{MeanHSV.V} + 50]$$

$$\text{HSVMin} = [\text{MeanHSV.H} - 25,\quad \text{MeanHSV.S} - 50,\quad \text{MeanHSV.V} - 50]$$

$$\text{Binaryimage}(i,j) = \begin{cases} 1 & \text{if HSVMin} \le \text{HSVimage}(i,j) \le \text{HSVMax} \\ 0 & \text{Otherwise} \end{cases} \quad \textbf{(5)}$$

Where Binaryimage(,) is produced binary image, HSVimage(,) is a pixel of HSV image, HSVMax is a HSV pixel, HSVMin is a HSV pixel.

Dilation: Dilates the binary image by using a 3×3 rectangular structuring element to fill holes of a size equal to or smaller than the structuring element.

Contours: Finding the contours in a binary image using border following algorithm. It determines the surrounding relations among the borders of a binary image. Since the outer borders and the holes borders have a one-to-one correspondence to the connected components of 1-pixels to the holes, respectively, the algorithm yields a representation of a binary image, from which one can extract some sort of features without reconstructing the image [8].

The comparisons: the result from the previous step is an array of contours. Every contour in this array is a candidate to be the target object. Two comparisons have been used to determine which one is the target object. The first comparison between the Original Rectangle and Object Rectangle. The second comparison between the Last Object Rectangle and the Candidate Object Rectangles. The two comparisons are explained in the following:

-The first comparison: the change in area of the target object from frame to frame is relatively very small. Therefore, the comparison between the Last Object Rectangle and the candidate Object Rectangle will give the ratio of change in the area of object between the current frame and the previous frame. This ratio is called **Current Ratio** and used to avoid choosing the wrong target object depending on its color only. **Current Ratio** is calculated as follows:

$$\text{Current Ratio} = \left|1 - \frac{\text{Candidate Object Rectangle (i). area}}{\text{Last object Rectangle . area}}\right| \tag{6}$$

Where Candidate Object Rectangle ( ) is array of candidate object rectangles.

The value of the Current Ratio will determine whether this is the contour of the target object or not. If the value of a Current Ratio is small or equal to 0.25, the candidate Object Rectangle will be passed to the second comparison to make sure that it is the target object. But if current ratio is greater than 0.25, the current candidate target object will be rejected and the Current Ratio will be calculated to the next Candidate Object Rectangle. If the current ratio is greater than 0.25 for all candidate object rectangles, the candidate object rectangle with the smaller current ratio will be the target object.

-The second comparison: the ratio between the height of the object and its width is relatively constant. Therefore, this ratio is used to distinguish the target object from among other candidate target objects. This ratio is called **Dimensional ratio** and the calculation for the original rectangle once only at the beginning of the tracking step. Dimensional ratio is computed using the following:

$$\text{H} = \text{Original Rectangle.Height}$$
$$\text{W} = \text{Original Rectangle.Width}$$
$$\text{Dimensional ratio} = \begin{cases} \frac{H}{W} & ,H \geq W \\ \frac{W}{H} & ,H < W \end{cases} \tag{7}$$

Where Dimensional Ratio of an Original Rectangle is compared with a Dimensional Ratio that will be calculated for the candidate object Rectangle that is passed from the first comparison. The candidate object that has the closed Dimensional ratio to Dimensional ratio of original rectangle will be the target object.

B-Issuing the Commands

Forward and backward commands are based on the amount of change in the area of the target object between the current area and the original area that previously stored and computed at first frame after object selection frame. If the ratio of the change in area of selected object is less than 0.8 from the original stored area, this means that the object moved away and it seems smaller on the screen. Therefore, the robot will move toward the object. While if ratio more than 1.5 from the original stored area this means that the object approached and it seems bigger on the screen. Therefore, the robot will move backward.

$$\text{ratio} = \frac{\text{Object Rectangle.area}}{\text{Original Rectangle.area}} \tag{8a}$$

$$\text{Command} = \begin{cases} \text{Forward} & \text{if ratio} \leq 0.8 \\ \text{Backward} & \text{if ratio} > 1.5 \end{cases} \tag{8b}$$

Left and right commands are depended on the amount of horizontal movement of the target object and determined by comparing the horizontal coordinate of top left point and top right point of the object rectangle with the horizontal coordinate of top left point and top right point of center rectangle according to the following equations:

$$\text{HTLO} = \text{object rectangle.x}$$
$$\text{HTLC} = \text{center rectangle.x}$$

$$\text{HTRO} = \text{object rectangle.x} + \text{object rectangle.width}$$
$$\text{HTRC} = \text{center rectangle.x} + \text{center rectangle.width}$$
$$\text{Command} = \begin{cases} \text{Left}, & \text{HTLO} < \text{HTLC} \\ \text{Right}, & \text{HTRO} > \text{HTRC} \end{cases} \qquad \textbf{(9)}$$

Where HTLO is the horizontal coordinate of top left point of the object rectangle, HTRO is the horizontal coordinate of top right point of the object rectangle, HTLC is the horizontal coordinate of top left point of center rectangle, and HTRC is the horizontal coordinate of top right point of center rectangle.

Camera up and Camera down commands are depended on the amount of vertical movement of the target object and determined by comparing the vertical coordinate of top left point and down left point of the object rectangle with the vertical coordinate of top left point and down left point of center rectangle according to the following equations:

$$\text{VTLO} = \text{object rectangle.y}$$
$$\text{VTLC} = \text{center rectangle.y}$$
$$\text{VDLO} = \text{object rectangle.y} + \text{object rectangle.height}$$
$$\text{VDLC} = \text{center rectangle.y} + \text{center rectangle.height}$$
$$\text{Command} = \begin{cases} \text{Camera up}, & \text{VTLO} < \text{VTLC} \\ \text{Camera down}, & \text{VDLO} > \text{VDLC} \end{cases} \qquad \textbf{(10)}$$

Where VTLO is the vertical coordinate of top left point of the object rectangle, VTLC is the vertical coordinate of top left point of the center rectangle, VDLO is the vertical coordinate of down left point of object rectangle, and VDLC is the vertical coordinate of down left point of center rectangle.

In the case of losing of the object and not appearing on the screen, the center point of Last Object Rectangle will compare with the center point of frame to estimate the current position of target object according to the following equations:

$$\text{LCtr.x} = \frac{\text{LastObjectRectangle.x+LastObjectRectangle.width}}{2} \qquad \textbf{(11a)}$$
$$\text{LCtr.y} = \frac{\text{LastObjectRectangle.y+LastObjectRectangle.height}}{2} \qquad \textbf{(11b)}$$
$$\text{FCtr.x} = \frac{\text{frame.width}}{2} \qquad \textbf{(11c)}$$
$$\text{FCtr.y} = \frac{\text{frame.height}}{2} \qquad \textbf{(11d)}$$
$$\text{Command} = \begin{cases} \text{Left}, & \text{LCtr.x} \le \text{FCtr.x} \\ \text{Right}, & \text{LCtr.x} > \text{FCtr.x} \end{cases} \qquad \textbf{(11e)}$$
$$\text{Command} = \begin{cases} \text{Camera Up}, & \text{LCtr.y} \le \text{FCtr.y} \\ \text{Camera Down}, & \text{LCtr.y} > \text{FCtr.y} \end{cases} \qquad \textbf{(11f)}$$

3- Experimental Results

The object tracking system have been developed by using Java for mobile programming language and Arduino Integrated Development Environment (IDE) programming language. The developed programs work under windows 8 operating system. The size of the used robot was 15cm width, 30cm length, and 10cm height of body as shown in Figure-3. All tests have been conducted using android device Samsung Galaxy Core GT-I8262 and Arduino UNO328 controller board.

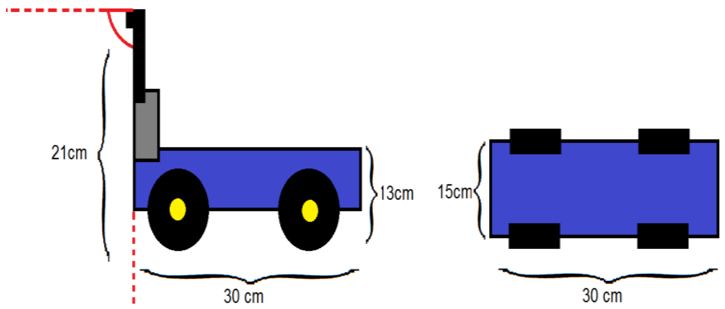

Figure 3 - The side view of the robot

The using map is a square with dimensions 2×2 meter. Map start point is (0.0) and ends at the point (200,200) as clear in Figure- 4. The circumference of map is 8 meter. Inside that map, there is a circle with a radius of one meter. This map allows us to measure the distance traveled and the distance between the robot and the object and the angle of the robot movement.

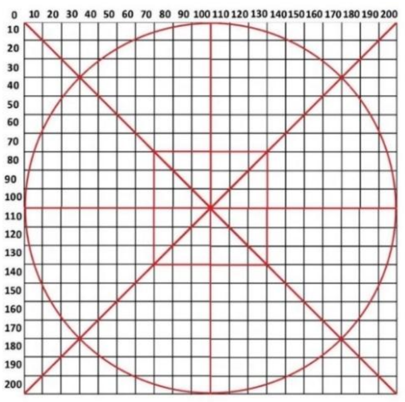

Figure 4 - The planned path used in the considered tests.

Through applying the object tracking method, there were many variables needed to be set. In the present work, there are three main parameters will affect the obtained results from the experiments:

A-Frame rate: is the number of frames that processed in one minute. The resolution of each frame is 640×480 pixel. In this work, the eight selected cases are 1fps, 2fps, 3fps, 4fps, 5fps, 6fps, 7fps and 8fps, to test the system.

B-Motor period time: the mobility characteristic of the robot depends on its motor type, which determines the motion response of the robot. The used motor is DC electric gear motor. The use of this type does not give the chance to control the angular velocity. Instead, the available method to control the time rate of motor revolutions is the time duration (or, period) only. Therefore, there was a need to determine the best period of the motor that corresponding to optimal response of the robot. It should be mentioned that the period time less than 50ms and greater than 200ms and frame rate greater than 8fps caused a fluctuate behavior for the robot when follow the object. Therefore, in this work were selected four periods are 200ms, 150ms, 100ms and 50ms to test the system.

C-Target Object: the selected object in our experiments is fully color red ball with 10 cm diameter. Because it has a uniform color and can be moved to any direction easily, as well as for use in many previous works. The speed of the ball is fixed at 14 cm / sec. Because the minimum speed of the robot is approximately 8 cm / sec, while the Maximum speed of the robot is approximately 20 cm / sec.

To determine the appropriate values for each of the frame rate and the motor period, 64 cases have been tested that involves all possible eventualities for each frame rate and motor period. Calculate the mission time for each case and which will be the criterion for the selection of the appropriate parameters. All experiments were done using the mentioned map on the two path that are circle path and the square path, the results of experiments and analysis of it's explain as following**:**

The analysis of the robot behavior according to the information that extended horizontally at the default value of motor period MP (i.e.100ms) in Figure-5 and 6 shows that the mission time consumed short temporal amount of about 50sec at frame rate 7fps for circle path and about 57sec at frame rate 6fps for square path. In a comparison with other frame rates, the variation of mission time in such case is expected and exponentially increasing. The frame rate values less than 3fps make the robot late. Because the robot completes the processing of the current frame and waits for receipt of the next frame. The wait time is lost without any action. While the frame rate values greater than 4fps make the robot does not give stable and clear result for motor period greater than 100ms.Therefore, one can conclude that the best value of frame rate that enables the controller to analyze the captured image in time with less mission time and give stable and clear result for most motor period which is the 3fps and 4fps frame rate. Whereas, the analysis of the robot behavior according to the information that extended vertically at frame rate=3fps and 4fps in Figure -5 and 6 shows that the value of mission time with motor period =100ms and 150ms approximately is half the mission time in motor period =50ms. The difference between the value of mission time in case of 100ms and 150ms for motor period is very small. Motor period 150ms with frame rate more than 4fps does not give stable result. It is clear the value of motor period that gives result stable along the frame rate is 100ms. One can conclude that the value of motor period that give the best result of mission time with frame rate 3fps and 4fps is 100ms and 150ms.

1-Circle path

Table 2 - Mission time (in Sec) versus frame rate (fps) at the considered motor period on circle path.

Frame Rate (fps)	MP 50ms	MP 100ms	MP 150ms	MP 200ms
1	184	111	78	66
2	126	58	48	46
3	90	52	46	0
4	61	50	48	0
5	57	52	0	0
6	66	54	0	0
7	70	50	0	0
8	56	55	0	0

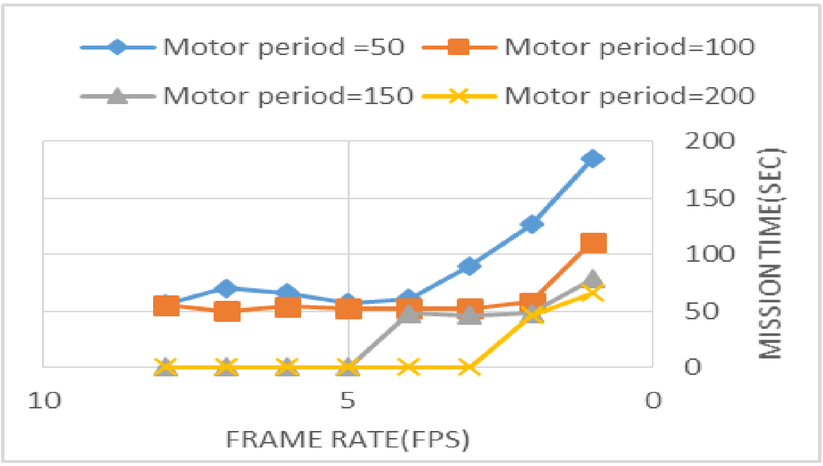

Figure 5- Mission time (sec) versus frame rate (fps) for different Motor period (ms) for circle path.

2-Square path

Table 3- Mission time (in Sec) versus frame rate (fps) at the considered motor period on square path.

Frame Rate (fps)	MP 50ms	MP 100ms	MP 150ms	MP 200ms
1	211	133	96	81
2	149	91	73	68
3	136	67	59	0
4	95	59	64	0
5	89	65	0	0
6	79	57	0	0
7	116	62	0	0
8	77	75	0	0

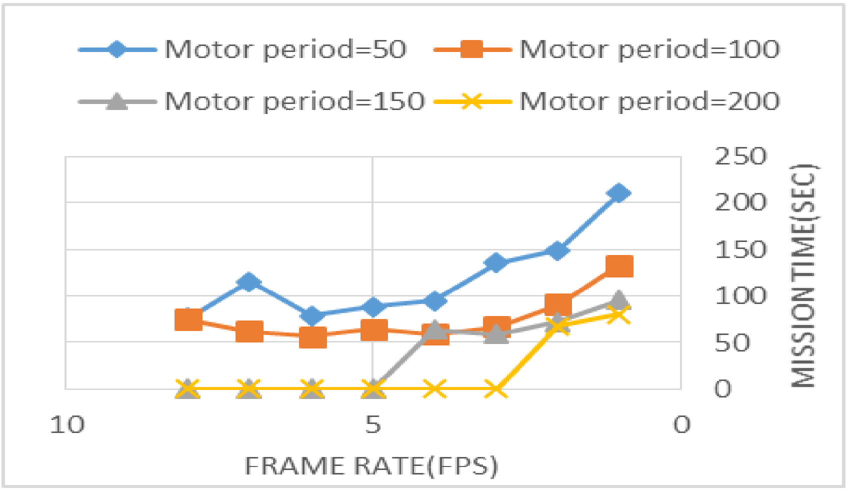

Figure 6 - Mission time (sec) versus frame rate (fps) for different Motor period (ms) for square path.

The Table-3 shows the success rates for each experiment. The success ratios has been calculated based on comparing actual results of tracking objects with ideal results in each case. In the following tables and maps, are shown the actual experiences and the behavior of the robot in several different cases. Four cases are clarified for each of the square and circle path. The Table-4 illustrates the details of the four cases. The distance between the robot and target object at start point is 78cm in circle path and 98cm in square path.

Table 4 - The four cases for the actual experiences of object tracking.

Case	Frame rate(fps)	Motor period (ms)
1	3	100
2	3	150
3	4	100
4	4	150

Table 5 - Results of object tracking tests on circle path for mobile robot.

No.	Object Path		Case(1)		Case(2)		Case(3)		Case(4)	
	X	Y	X	Y	X	Y	X	Y	X	Y
1	98	0	0	0	0	0	0	0	0	0
2	154	0	43	1	65	0	56	4	46	1
3	200	24	96	14	128	5	121	7	96	7
4	200	94	137	38	184	51	171	45	148	19
5	200	164	166	71	192	100	197	97	183	61
6	166	200	178	107	169	140	177	140	194	109
7	96	200	165	140	132	176	142	169	168	156
8	26	200	122	150	81	196	99	176	117	181
9	0	156	85	139	34	186	57	176	74	186
10	0	86	64	126	10	145	25	140	43	168
11	0	16	46	109	1	95	15	90	17	129
12	42	0	41	88	7	57	21	50	8	76
13	98	0	37	78	20	45	34	35	33	37

Figures-8, 9,10,11are show the actual shape of the path of movement of the robot on the map for circle path.

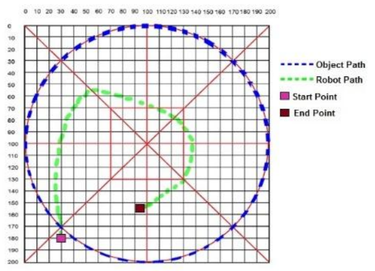

Figure 8 - Robot behavior of object tracking when frame rate is 3fps, motor period is 100ms and mission time is 52sec for circle path.

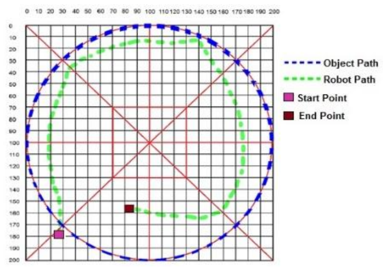

Figure 9 - Robot behavior of object tracking when frame rate is 3fps, motor period is 150ms and mission time is 46sec for circle path.

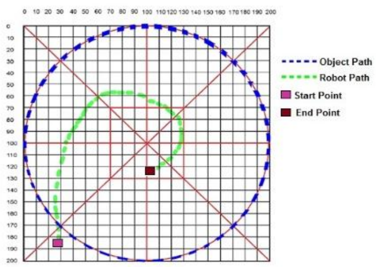

Figure 10 - Robot behavior of object tracking when frame rate is 4fps, motor period is 100ms and mission time is 50sec for circle path.

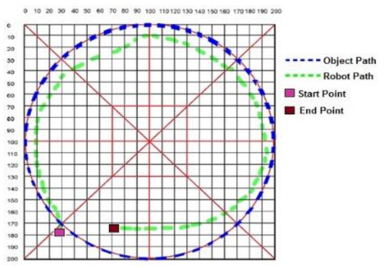

Figure 11 - Robot behavior of object tracking when frame rate is 4fps, motor period is 150ms and mission time is 48sec circle path.

Table 6 - Results of object tracking tests on square path for mobile robot.

No.	Object Path		Case(1)		Case(2)		Case(3)		Case(4)	
	X	Y	X	Y	X	Y	X	Y	X	Y
1	0	100	30	170	30	170	30	170	30	170
2	13	48	26	138	20	107	27	140	12	125
3	68	4	34	87	28	56	34	103	14	69
4	131	5	51	58	70	19	48	77	53	32
5	184	47	87	63	119	16	74	57	106	10
6	198	113	122	78	157	40	98	60	160	36
7	165	171	136	103	172	77	122	74	189	86
8	104	200	132	126	174	125	128	96	183	137
9	37	180	118	140	146	162	121	111	142	168
10	1	122	103	152	100	160	107	121	90	174
11	0	94	98	154	90	157	103	123	74	175

Figures (12-13-14-15) are show the actual shape of the path of movement of the robot on the map for circle path.

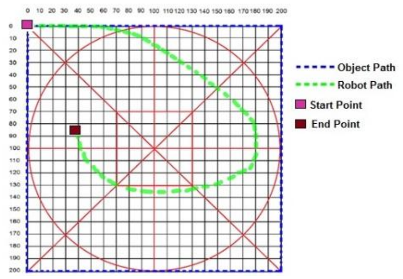

Figure 12 - Robot behavior of object tracking when frame rate is 3fps, motor period is 100ms and mission time is 67sec for square path.

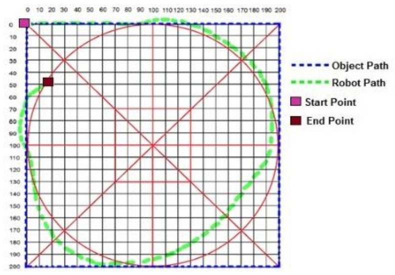

Figure 13 - Robot behavior of object tracking when frame rate is 3fps, motor period is 150ms and mission time is 59sec for square path.

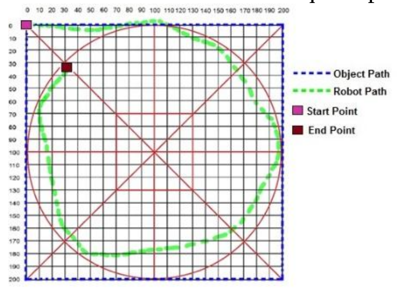

Figure 14 - Robot behavior of object tracking when frame rate is 4fps, motor period is 100ms and mission time is 59sec for square path.

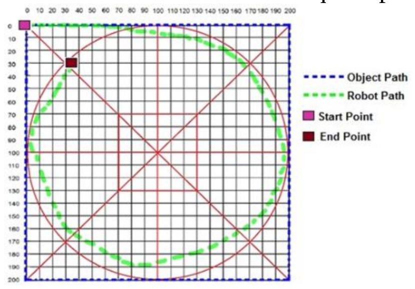

Figure 15 - Robot behavior of object tracking when frame rate is 4fps, motor period is 150ms and mission time is 64sec for square path.

3- Conclusion

The results of the experiments on the object tracking system showed the following conclusions:

-The useful frame rate values were 3fps and 4fps, the frame less than 3fps were excluded since it was consuming a longer mission time. The frame rate more than 4fps were excluded because it gave unstable result.

- The best motor period is 100ms and 150ms.Motor period values smaller than 100ms caused slowly down system while value greater than 100ms caused a reducing in the reliable of detection the object.
- Objects depending on their color was an effective way as it consumed a little time and required uncomplicated calculations.
- Method of tracking the object was reliable, keeping the distance between the robot and the object made the object visible always to the camera.

References:

1. Hildebrandt, H. and Mireille, F. **2015**. *Smart Technologies and the End (s) of Law*: *Novel Entanglements of Law and Technology*. Edward Elgar Publishing.
2. Mendes, M.Q. **2010**. *Vision-based Navigation Using an Associative Memory*. INTECH Open Access Publisher.
3. Murray, R.M. **2013**. *A mathematical introduction to robotic manipulation*. CRC press.
4. Kunica, Z., Davor, Z., and Mladen C.**2007.** Mobile Robot Vision System for Object Color Tracking. 14th International Scientific Conference on Production Engineering.
5. Barnea,A.K. **2011**. *Specific Problems Regarding Localization of Autonomous Mobile Robots*. Mangeron no.61-63, Iasi 700050, Romania.
6. Browning, B. and Manuela, V. **2005**. Real-time, adaptive color-based robot vision. Intelligent Robots and Systems, 2005. (IROS 2005). IEEE/RSJ International Conference on. IEEE.
7. Islam, M. Z. and Chi-min, O. and Chil-Woo, L. **2008**. *Real time moving object tracking by particle filter*. Computer Science and its Applications. CSA'08. International Symposium on. IEEE.
8. Suzuki, S. **1985**. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1), pp: 32-46.