Low Cost Indoor Location Management System using Infrared Leds and Wii Remote Controller[*]

Baris Tas, Nihat Altiparmak and Ali Şaman Tosun
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
{btas,naltipar,tosun}@cs.utsa.edu

Abstract

Many applications in wireless sensor networks can benefit from position information. However, existing accurate solutions for indoor environments are costly. RF based approaches are not suitable for some indoor environments such as factory floors where heavy machinery can cause interference. In this paper, we propose a low cost and simple location management system using the Wii Remote Controller and infrared leds. Proposed solution is motivated by the need to find the location of a mobile robot used for data collection in a wireless sensor network. In proposed scheme, Wii Remote Controller is placed on the mobile robot pointing upward and several IR leds are placed on the ceiling. Proposed scheme uses the resources efficiently and can cover a large area using a single Wii Remote Controller and multiple IR leds. Proposed scheme is easy to implement and requires minimal bandwith for location management.

1. Introduction

A wireless sensor network (WSN) consists of potentially hundreds of sensor nodes and is deployed in an ad hoc manner for collecting data from a region of interest over a period of time. Even though the technology is new, WSNs received an enthusiastic reception in the science community as WSNs enable precise and fine-grain monitoring of a large region in real-time. Some examples of successful large-scale deployments of WSNs to date are in the context of ecology monitoring (monitoring of micro-climate forming in redwood forests [23]), habitat monitoring (monitoring of nesting behavior of seabirds [17]), and military surveillance (detection and classification of an intruder as a civilian, soldier, car, or SUV [3, 4]).

[*]Partially supported by US National Science Foundation (NSF) Grant CCF-0702728

To improve the scalability and performance of WSNs, there has been a flurry of work on employing a mobile node for data collection. The data mules [19] work exploit random movement of mobile node to opportunistically collect data from a sparse WSN. Here, the nodes buffer all their data locally, and upload the data only when the mobile node arrives within direct communication distance. Zebranet [14] system uses tracking collars carried by animals for wildlife tracking. Data is forwarded in a peer-to-peer manner and redundant copies are stored in other nodes. Shared wireless infostation model [20] uses radio tagged whales as part of a biological information acquisition system. Mobility of the mobile node is not controlled in these approaches. Mobile element scheduling (MES) work [21] considers controlled mobility of the mobile node in order to reduce latency and serve the varying data-rates in the WSNs effectively. The MES work shows that the problem of planning a path for the mobile node to visit the nodes before their buffers overflow is NP-complete. Heuristic based simple solutions are proposed to address this problem [21, 11, 27]. Sencar [16] uses a mobile observer to collect data. Area is divided into regions and the mobile node moves in straight lines in each region. Multihop forwarding is used to relay packets from distant sensors to sencar. Data salmon [7] constructs a spanning tree and moves the mobile basestation on this tree to optimize the cost of retrieval. To reduce the size of the path the mobile node travels, rendezvous points are used as regional collection points and the mobile node collects the data from the rendezvous points [26]. Mobile nodes are also used for data collection, storage and retrieval in underwater sensor networks [24]. This work assumes a single mobile node. Multiple mobile nodes are also proposed to improve the performance [13] using load balancing between the mobile nodes.

Efficient use of mobile nodes require location information. Mobile robot should know its approximate location to follow predetermined optimal paths and make location dependent decisions. Individual sensors should also know the location of the mobile to coordinate sleep-wake up schedules and save resources. Location management schemes using GPS [9] are not suitable for indoor environments. Existing approaches include RADAR [5] which uses RF to locate and track users inside buildings. RF based schemes use RSSI (Received Signal Strength Indicator), which is obtained automatically with the received messages in most sensor radios. Although RF based schemes provide the cheapest localization technique [22], they yield very noisy estimations, especially for indoor systems [25]. Also, the RSSI values depend on many factors ranging from the antenna orientation to the environment specifics [22]. In a factory setting, heavy machinery causes RF signal interference impeding the accuracy of localization service based on RF signals. To overcome limitations of RF only schemes, combination of RF and ultrasound are used to provide location information in some applications [18, 12].

In this paper, we propose a framework for indoor location management using Wii Remote Controller (WRC) and IR leds. WRC has a resolution of 1024×768 and can detect each IR led as one pixel having Wii.x and Wii.y coordinates. Each WRC has capability of broadcasting the coordinates through bluetooth. By carefully placing IR leds on the ceiling, we can compute the location of a mobile robot using WRC.

The rest of the paper is organized as follows. Proposed tracking system is described in section 2. We show how to find the location of the mobile element in section 3. In section 4, rotation of the mobile element is discussed. We describe simulation results in section 5 and conclude with section 6.

2. Proposed Tracking System

WRC is placed on the mobile device in an upright position pointing to the ceiling with the tip of its IR camera sensor. IR light sources are placed above a certain distance from the floor, pointing to the floor as shown in Figure 1. First, we consider the robot moving only vertically and horizontally, without changing its orientation and then we consider rotations of the robot. To expand the coverage area to arbitrary sized grids, it is enough to increase the number of IR light sources used. No additional WRCs are needed. Therefore, the cost is quite low since the cost of one WRC corresponds to the cost of nearly 200 IR light sources. If a fixed WRC was placed on the ceiling and one IR sensor was placed on the mobile robot, multiple WRCs would be needed to expand the coverage area resulting in a costly solution.

WRC is Nintendo Wii game console's controller, released in November 2006. WRC has an Infrared(IR) camera, which provides high resolution and high speed tracking

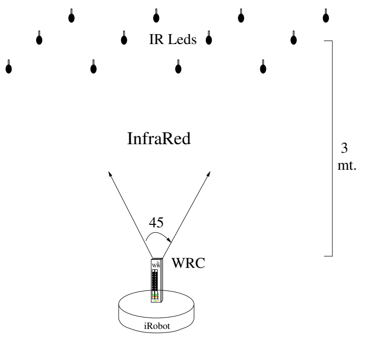


Figure 1. Tracking System

of up to four IR light sources at the same time. The camera provides location data with a resolution of 1024×768 pixels with a 100Hz refresh rate, and a 45 degree horizontal field of view [15]. Besides the IR camera, it has a wireless bluetooth connectivity as well, which makes it a perfect tracking device being compatible with PC. WRC has a suggested retail price of US$40.

The IR leds we used are TSAL6400 high power Infrared emitting diodes, with a peak wavelength of 940 nanometer, radiant power of 35 milliwatt and half intensity angle of $\pm 25°$. It costs about US$0.20.

We use iRobot Create and Command Module [1], which is a programmable robot, as our mobile device. A Tmote Sky sensor [2] is connected to the iRobot Command Module(ICM) using serial communication. The connection between the sensor and robot can be seen in Figure 2.

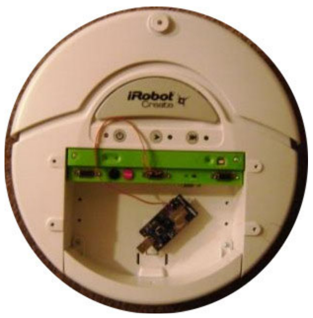


Figure 2. The configuration between iRobot and Tmote Sky sensor.

In our system, the data required to compute the location of the mobile element is sent via the bluetooth interface of WRC to the PC(base station) which in turn processes the data, and broadcasts the location of the mobile robot. In addition to providing the mobile robot with its location, this allows other sensors to learn the position of the mobile robot. Other sensors can switch to the sleep mode and save energy if the mobile robot is far away from them. When the mobile node is close, they can wake up and transmit the collected data to the mobile robot. In case the static sensors can not receive the broadcast messages from the PC, the Tmote Sky sensor on the robot can also broadcast this information enlarging the communication area.

3. Finding Location Via Local Information

Our goal is to find the location of the mobile robot using the positions of IR leds detected by the WRC. The area covered is large and the WRC is able to see only a fraction of the area and detect only the IR leds in this local area. The size of the observation window of WRC depends on the height. The WRC should be able to determine its position using the IR leds in its observation window. One challenge is that IR camera cannot differentiate IR light sources. There are no unique IDs corresponding to IR light sources. In this setting, we have to differentiate these IR leds in some way in order to understand where our mobile robot is. One way of solving this problem is irregular placement of IR light sources. Using relative position of each static IR led with respect to each other, we can compare all the vectors between each pair of static IR leds and differentiate these pairs. After differentiating them, it is easy to find the physical location of the mobile robot. An example of an irregular placement of IR leds is given in Figure 3. The vectors we consider are {(1,2); (1,3); ... ; (4,5)} where a vector is represented as $(startLed, endLed)$. The corresponding slope and length pairs of these vectors are {(-3, $\sqrt{10}$); (-$\frac{1}{2}$, $\sqrt{5}$); (-$\frac{2}{3}$, $\sqrt{13}$); (-1, 4$\sqrt{2}$); (2, $\sqrt{5}$); ($\frac{1}{2}$, $\sqrt{5}$); (-$\frac{1}{3}$, $\sqrt{10}$); (-1, $\sqrt{2}$); (-$\frac{3}{2}$, $\sqrt{13}$); (-2, $\sqrt{5}$)} where a pair is represented as $(slope, length)$, and none of the IR led pairs has the same slope and length. Assuming the thick rectangle in the figure is a window that the WRC sees, then the location of the mobile element is figured out by examining the IR leds in that window. The slope-length pairs in WRC's window is computed. In the example, the WRC sees two IR leds, and the slope-length pair is computed as (-1, $\sqrt{2}$). Since all the slope-length pairs are unique, the corresponding vector, (3, 4) is found. Once the vector is found, it is trivial to find out the location of the mobile element because we know the locations of the IR leds that constitute the found vector. We talk about the details later in this section. Costas arrays are one way of providing this irregular placement and they are explained next.

3.1. Costas Arrays

In order to position static IR leds, we make use of Costas arrays, firstly defined in [10] and used in [6]. In an $N \times N$ Costas array, there is exactly one dot in each row and column and none of the dot pairs have the same slope and

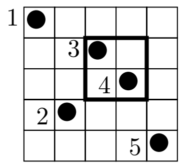
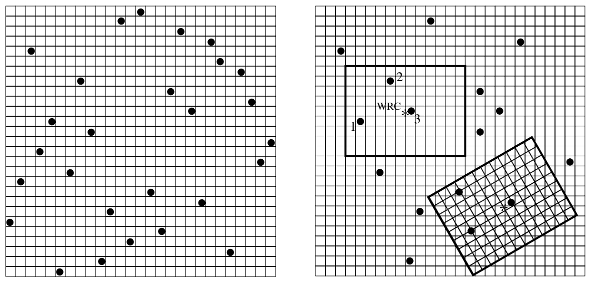


Figure 3. Irregular placement of IR leds

length. In other words, there are distinct vector differences between all pairs of dots. Figure 4(a) is an example of a Costas array of order 27.

Let us assume that the 27×27 grid in Figure 4(a) is the area that we should cover for tracking, which is our global area. At this point, since Wii can see a 4×3 (1024×768) rectangle, which is our local area; we should choose a window with a size of 4×3 or its integer multiples. In order to find the physical position of mobile robot, we need at least two IR leds in our local area since we need at least one pair of IR led to differantiate the leds using the unique vector difference between them. The window size that provides this constraint is generally very large. Therefore, it is reasonable to start with a window size that covers at least 1 IR led in any place of our tracking area. In our specific example, 12×9 window contains at least 1 but generally 2 to 4 static IR sources at any place of the whole grid. This window size can be found by using the matrix structure that we will explain later. The next step to be taken is first to remove unnecessary static IR leds, then to add some more leds in order to satisfy the condition of having at least two leds in local area. Figure 4(b) demonstates the final state of our system, with a window size of 12×9 and with 16 static IR leds in a 27×27 grid. This is probably not an optimal solution for our problem, however; optimal solution is NP-complete as follows.



(a) A Costas array of order 27. (b) Our modified system

Figure 4. Costas Array Based Tracking

3.2. NP-Completeness of Wii Coverage Problem

Definition 1. ***Wii Coverage(WC) Problem***
Given $N_1 \times N_2$ array with points, find minimum number of leds that satisfy the condition that each $N_3 \times N_4$ subarray of $N_1 \times N_2$ array has at least k leds in it, where k is a positive integer.

Definition 2. ***Hitting Set***
Given a collection C of subsets of a finite set S, a positive integer $K \leq |S|$. Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that S' contains at least one element from each subset in C? This problem is defined as hitting set problem and it is NP-complete [8].

Theorem 1. *The WC problem is NP-complete.*

Proof. Hitting Set is polynomial-time reducible to Wii Coverage Problem. Create the collection C using all the $N_3 \times N_4$ windows. Having an IR led in each window is finding Hitting set in this collection. □

3.3. Matrix Structure

Since the Wii Coverage Problem is NP-Complete, first, we need to verify the correctness of a given solution to the problem. Let's assume that our example Costas Array is a solution to our problem. In order to verify that whether this solution is correct or not, we build the following matrix structure.
The basis for the structure comes from the set theory. In Figure 5, we have the sets A_1, A_2, A_3, A_4 each covers a specific area of the whole rectangle. Then, the number of points in the area, A_4 is found by the following equation.

$$A_4 = (A_1 + A_2 + A_3 + A_4) - [(A_1 + A_2) + (A_1 + A_3) - A_1] \quad (1)$$

The first step in finding the number of points in a specified rectangle is to create and initialize a matrix, MatrixPoints, with the same size as the Costas Array's size (the same number of rows and columns). The existence of a point in the costas array is represented by a 1, and nonexistence of a point is represented by a 0 in the matrix. The goal is to determine how many points there are in a $X \times Y$ window area at a given time. We need an auxiliary matrix, M, to achieve the goal. Each entry of this matrix is set with the number of points in the rectangle defined between (0,0) and the coordinates of the related entry. Lines 1-4 in Algorithm 1 show how to build the auxiliary matrix. Once we constructed the auxiliary matrix, M, we can determine the number of pairs in a specific window. The equation 2 finds the number of points in a window of which the right bottom coordinates are i, j; width is X; and height is Y using the matrix M.

$$A_4 = M(i,j) - [M(i,j-Y) + M(i-X,j)] + M(i-X,j-Y) \quad (2)$$

Figure 5 shows an example array and WRC window area A_4 inside of our array. Finally, we derive the algorithm 1 that verifies whether all $X \times Y$ windows contain at least k number of points(IR leds) or not. The complexity of this algorithm is $O(mn)$ where *m*, and *n* are the dimensions of the given matrix.

Algorithm 1 Verify(MatrixPoints, k, X, Y, RowSize, ColumnSize)

```
1: M = MatrixPoints
2: for $i = 1$ to RowSize do
3:     for $j = 1$ to ColumnSize do
4:         $M[i,j] += M[i-1,j] + M[i,j-1] - M[i-1,j-1]$
5: for $i = X$ to RowSize do
6:     for $j = Y$ to ColumnSize do
7:         if $M[i,j] - M[i,j-Y] - M[i-X,j] + M[i-X,j-Y] < k$ then
8:             return NOT VERIFIED
9: return VERIFIED
```

For the array in Figure 5 for example, we need to create a 9×9 matrix and fill each entry of this matrix with the number of points in the rectangle defined between (0,0) and the coordinates of the related entry. For example, number of points in the area of A_1 is kept in the 3rd row and 4th column of our matrix, $M(3,4)$. By using our matrix, we can easily check the number of points in our WRC window as we show in equations 3 and 4.

$$A_4 = (A_1 + A_2 + A_3 + A_4) - [(A_1 + A_2) + (A_1 + A_3) - A_1] \quad (3)$$

$$A_4 = M(6,8) - [M(3,8) + M(6,4) - M(3,4)] \quad (4)$$

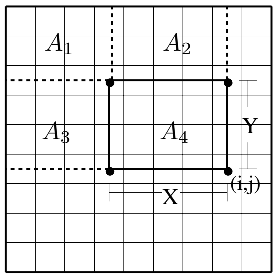


Figure 5. Finding number of points in WRC window.

3.4. Modifying Costas Array

Our system needs to have at least two IR leds in a 12×9 window. However, in costas arrays, some windows may contain more than two, and some may contain only one IR led. Therefore, we represent two algorithms for modifying a chosen costas array: *removing IR leds* for windows having more than two leds, and *adding IR leds* for windows which contains one IR led. The sequence of applying these algorithms differs depending on the costas array. This sequence is discussed later in this section. In the removing IR leds algorithm, we remove the points until we guarantee at least two IR leds in a 12×9 window. Our heuristic is removing the IR leds which are closer to each other, algorithm 2, line 3. When removing a point, we know the number of IR leds in any window thanks to the matrix structure. Our attention is on the 12×9 windows. Algorithm 2 shows the steps.

Algorithm 2 Removing IR leds

```
1:  Construct the matrix structure
2:  Compute the distances between points forming Pairs Array
3:  Sort the Pairs Array in ascending order, using distances as sort
    keys
    // provides starting with the points which are close to each
    other
4:  for each pair in the Pairs Array do
5:      Select the point which maximizes the number of distinct
        distances between points after removing
6:      if removing the selected point does not violate the con-
        straint that every $12 \times 9$ window contains at least two points
        then
7:          remove the selected point
8:          update the matrix structure
9:      else if removing the other point in the pair does not violate
        the constraint that every $12 \times 9$ window contains at least
        two points then
10:         remove this point
11:         update the matrix structure
```

Since we want as many distinct distance values as possible, we apply a second heuristic, algorithm 2, line 5. After determining the closest pair of points, first we select the one which maximizes the number of distinct distance values among all points when it is removed. This point will be our first candidate to remove. If our constraint is violated, we try to remove the other point in the pair. If the constraint still does not let the removal of this point, we continue with the next pair of points where the distance between the points is the next smallest one.

In algorithm 2, the initialization phase, lines 1 - 3, runs in $O(s^2 log(s))$ where s is the number of IR leds (or the number of points in the matrix structure). The rest of the algorithm runs in $O(s^3 nm)$ where *n*, and *m* are the dimensions of the matrix. The total run-time of algorithm 2 is $O(s^2 log(s)) + O(s^3 nm) = O(s^3 nm)$.

Given the 12×9 windows which have 1 IR led in them, we have to guarantee that those windows contain at least 2 IR leds by adding IR leds. Our example Costas Array has six such windows. If we can find a spot in the intersection of all these windows, we guarantee at least two IR leds in every 12×9 windows. However, we have the unique distance and slope constraint between the points. Therefore, we may not find a suitable spot in the intersection of all these windows and we go with a greedy approach. Algorithm 3 demonstrates the steps to be taken. In the algorithm, a window is represented as a pair which holds the coordinates of its bottom right corner.

Algorithm 3 Adding IR leds

```
1:  CWS = Ø// candidate windows set
2:  W = set of windows having 1 IR led // using matrix structure
3:  while W$\neq$Ø do
4:      Sort the windows in W by y-coordinates
5:      Form a 2D range: [maximum y-coordinate; maximum y-
        coordinate $-$ window height]
6:      Add the windows in the range to CWS
7:      Sort the windows in CWS by x-coordinates
8:      Update the 2D range forming a rectangle area: [minimum
        x-coordinate; minimum x-coordinate $+$ window width]
9:      Update CWS according to the 2D range
10:     while 1 do
11:         if CWS.size $>$ 1 then
12:             Find the intersection of the windows in CWS
13:             Find a valid IR location that does not violate the
                distance-slope constraint for the intersection
14:             if found then
15:                 add the IR led
16:                 update W // using matrix structure
17:                 break
18:             else
19:                 update the CWS by removing the farthest window
                    // farthest window is the right-most window in
                    CWS
20:         else if CWS.size $==$ 1 then
21:             Find a valid location that does not violate the
                distance-slope constraint for the window in CWS
22:             if found then
23:                 add the IR led
24:                 update W // using matrix structure
25:                 break
26:             else
27:                 remove the window from CWS
28:         else
29:             exit
```

Algorithm 3 runs in $O(|W|^2 log(|W|)) + O(|CWS|^2 |W| s)$ time where s is the number of points in the array, $|W|$ is the number of windows containing one IR led, and $|CWS|$ is the number of candidate windows.

The sequence of applying algorithms 2 and 3 depends on the alignment of the costas array which is used. An IR

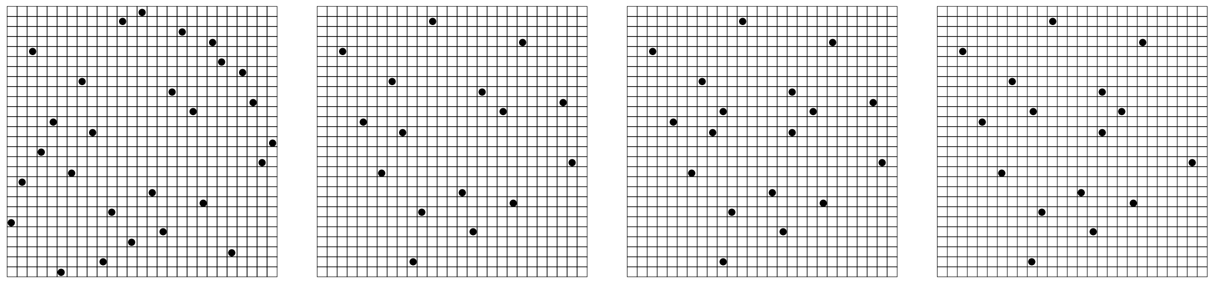

(a) Initial with 27 points. (b) After algorithm 2: 16 points (c) After algorithm 3: 18 points (d) Final system with 16 points.

Figure 6. Modifying Costas Array

led cannot be added to a particular window containing one IR led in it, unless the slope-distance constraint allows. In our specific example, trying to add IR leds for the windows with one IR led as a first step does not work because of the slope-distance constraint. Therefore, we first run the removing leds algorithm. The result of this algorithm applied on the original costs array, figure 6(a), is shown in figure 6(b). Now, we turn back to the adding leds algorithm, because there are windows which contain just one IR led. This time IR leds are added preserving the slope-distance constraint: two IR leds are added to guarantee at least two IR leds in spots, (10, 11) and (17,13), figure 6(c). Finally, if we run the removing IR leds algorithm again, the IR leds located at (25, 10) and (9, 13) are removed. In this way, we guarantee at least two IR leds in each and every 12×9 windows preserving slope-distance constraint by using only sixteen IR leds, figure 6(d). Even our example costas array having 27 IR leds could not achieve this.

It is possible to have some windows with less than 2 IR leds depending on the costas array we started as a result of slope-distance constraint. In this case, the mobile robot solves the problem by keeping its previous location and direction. After passing a few windows, the mobile runs into a window which contains at least two IR leds in it. Now, the mobile robot validates its location by using IR leds again.

3.5. Handling the Tracking Area

We work with small costas arrays. However, global area will have much larger resolution. Following approach shows how to map the small costas array to the global area while meeting the desired number of IR leds in each window.

1. Since we know the size of coverage area that we have to cover, firstly, we should find the ratio of $(\frac{total\ coverage\ area}{one\ WRC's\ coverage\ area})$ in order to decide about the amount of static IR leds we are required. Let us say we found t after division, and in the coverage area of WRC, there should be about p number of static IR leds. Now, we roughly know that $t \times p$ number of static IR leds are required.

2. As a second step, since we know the amount of static IR leds required, we can start from a costas array slightly bigger than the number of static IR leds required, and we can drop or add some points as necessary by following Algorithms 2 and 3.

3. Now, we have a small array with the required number of points in it and unique vector property; however, we have to map it into a bigger array until we fit our window size to 1024×768. By this way, each location of our array will represent a pixel. Assume that A is our $M \times M$ small array and B is our $N \times N$ bigger array, where $N = k \times M$ and k is a positive integer. If we map A into B by following Algorithm 4, B preserves the property of A that there are still distinct vector differences between all pairs of dots. Figure 7 shows an example of mapping a 3×3 costas array into a 9×9 array.

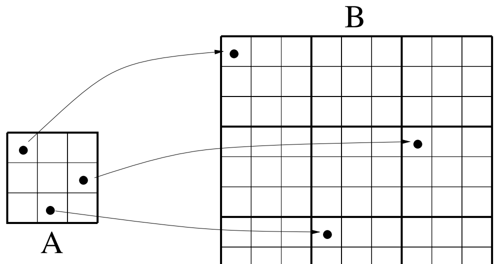


Figure 7. Mapping arrays.

Algorithm 4 Mapping Costas Array (A,B,M,k)

```
1: $Initialize\ B\ to\ all\ 0's$
2: for $i = 1\ to\ M$ do
3:     for $j = 1\ to\ M$ do
4:         if $A[i, j] == 1$ then
5:             $B[((i-1) \times k + 1),((j-1) \times k + 1)] = 1$
```

3.6. Matching Wii and Physical Coordinates

In order to match Wii and physical coordinates, we use the properties of our tracking system. First of all, as it is mentioned, we know the global physical coordinates of our static IR leds. Secondly, no two of the $\binom{16}{2}$ elements have the same slope and length. By using these properties, we can create a table-based data structure such that it has a table with index of unique distances among any two static IR leds. Since there may be some equal distances for some pairs of static IR leds, each entry table[i] is a pointer to a linked list where different slope values and the static IR led pair information are kept. By using this table, we can distinguish each static IR led and find the global physical coordinates of mobile WRC as we show in the following example.

Let us define that the area we have to cover is shown in Figure 4(b) and the area that WRC can see is defined as the upper window shown in that figure. WRC is in the middle of its coverage area and Wii coordinates of WRC is (512,384) since our window size is 1024 x 768. Our purpose is to find the global physical coordinates of WRC, which is also the location of the mobile robot. For now, we know that there are three IR light sources in the coverage of WRC and therefore, we also know their Wii coordinates; however, we do not have any idea about their global physical coordinates. In order to find their global physical coordinates, we use our data structure which we have created before. For three IR light sources, there are 3 different pairs, and for each pair, either the $distance$ among them or the $slope$ of the vector that joins them are different thanks to the property of Costas arrays. For each pair, by looking up our table using the $distance$ value between leds as an index, we can learn the global physical coordinates of these led pairs. For this example, Wii and global physical coordinates of each static IR led together with WRC are given in the following table;

	Wii Co.	Global Physical Co.
Led1	(128,469)	(4.5,11.5)
Led2	(384,128)	(7.5,7.5)
Led3	(554,384)	(9.5,10.5)
WRC	(512,384)	?

Since WRC is in the middle of our window and window size is pre-defined, we know the local physical coordinate of WRC, which is (6,4.5). For now, we know the Wii coordinates of WRC, Wii coordinates of IR leds, and the local physical coordinates of WRC. By using Wii coordinates and direct proportion, we can find the local physical coordinates of the IR leds, too. After finding all the local physical coordinates, by simply subtracting the local physical coordinate from the global one that we found using our table structure for any of the static IR led, we can find the value difference between the local and global physical coordinates for that window, which is found as (3,6) in our example. Finally, it is possible to find the global physical coordinate of WRC by simply adding that constant value difference to its local physical coordinate and (9,10.5) is found as a result of this addition.

4. Rotations of Mobile Device

If the mobile device is rotated as in Figure 4(b), two problems arise. First of all, it may not be always possible to guarantee at least two static IR leds in the local area. Secondly, we will not be able to use slope information, which will cause a problem in differentiation of static IR leds if there are more than 2 leds in the local area and there is no unique distance between any of the pairs.

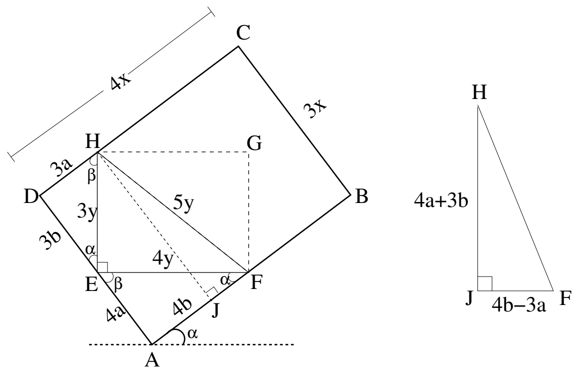


Figure 8. Finding the maximum $4k \times 3k$ axis aligned rectangle in ABCD.

First of all, we deal with guaranteeing at least two static IR leds in the local area. When a window of size $4m \times 3m$ is rotated it will have an axis aligned $4k \times 3k$ window in it. If we can guarantee two IR sensors in this $4k \times 3k$ window then we can gurantee two IR sensors in rotated $4m \times 3m$ window. We compute the maximum $4k \times 3k$ axis aligned rectangle area inside the big rotated $4m \times 3m$ rectangle for a given rotation angle α as shown in Figure 8. The ratio of area of axis aligned rectangle to rotated rectangle is $(\frac{3}{4sin(\alpha)+3cos(\alpha)})^2$. The value of alpha that minimizes this is $\alpha = 53°$ and the area ratio is $\frac{9}{25}$, which means that the smallest axis aligned $4k \times 3k$ rectangle is 36% of the bigger one. If our window size is 1024×768, we have to guarantee 2 static IR leds in every 614×460 axis aligned area to make our system work in rotations as well.

Now lets consider what happens if distance information suffice for location computation. Figure 9 shows the frequencies of different distance values of each led pair in the global window for Figure 4(b) . Maximum frequency is 5 and more than 3 is quiet rare, which means not having a unique distance pair is a low probability. Besides, in a large number of cases, more than two IR sensors will be in the window resulting in at least 3 pairs. If a pair does not produce a unique distance another pair can be used. If there are less than 3 leds in the local window and the distance value is not unique, the locations of pairs in the previous window can be kept and distinct distance pairs can be found. By making the calculation for the nearest previous window, location of the WRC can be found.

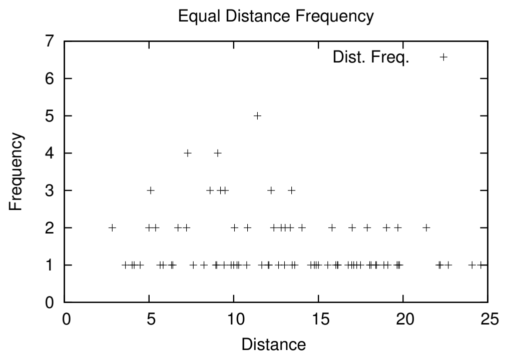


Figure 9. Equal Distance Frequencies.

5. Simulation Results

In this section, we evaluate the performance of our system using simulation. We simulated our system in a java program using the tacking area and led locations in Figure 4(b). In order to convert the costas array in Figure 4(a) to Figure 4(b), we first implemented the Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4. After finding the costas array that satisfies our minimum requirements shown in Figure 4(b), we filled a 2-dimensional array according to the values of costas array we found.

In the simulation, mobile robot picks a random destination inside our 2 dimensional array, turns and moves towards that destination by calculating its location in every window. Once it reaches that destination, another destination is picked. The path that mobile robot follows is given in Figure 10(a). Figure 10(a) corresponds to both the actual and the calculated path since it is guaranteed that there will be at least 2 leds in each and every window using our rotation scheme. Actual number of leds in each window on the path is plotted in Figure 10(b). According to simulation results, location of mobile device is calcuated in every window that robot visits, which validates the practicality of proposed scheme.

6. Conclusion

Many applications in wireless sensor networks can benefit from position information. However, existing solutions for indoor environments are costly. We propose a low cost and simple location management system using the Wii Remote and infrared leds. In proposed scheme Wii Remote is placed on the mobile robot and multiple infrared leds are placed on the ceiling according to a costas array based pattern. Mobile robot finds its current position using the infrared leds in its current window. The relative position of infrared leds with respect to each other is used to find the global position. Proposed scheme can handle rotations and can compute current location of robot very efficiently using the distance between detected IR leds.

References

[1] iRobot Create manuals, iRobot Corporation. Website. `http://store.irobot.com/family/index.jsp?categoryId=2591511&cp=3311368`.

[2] Ultra low power IEEE 802.15.4 compliant wireless sensor module, Tmote Sky datasheet, Moteiv Corporation, 2006.

[3] A. Arora and et. al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, 2004.

[4] A. Arora and et. al. Exscal: Elements of an extreme scale wireless sensor network. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 102–108, 2005.

[5] P. Bahl and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *IEEE INFOCOM*, 2000.

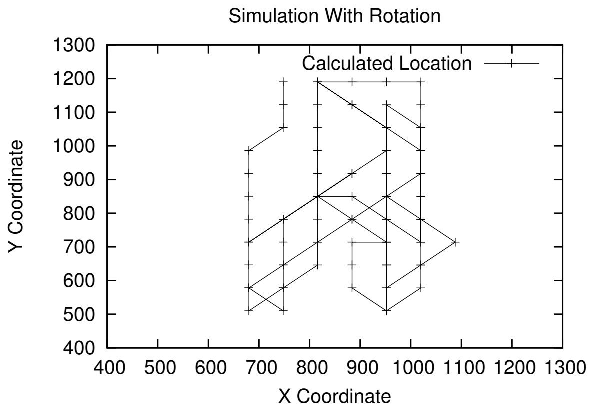
[6] J. Costas. A study of detection waveforms having nearly ideal range-doppler ambiguity properties. In *Proceedings of the IEEE*, volume 72, pages 996–1009, Aug. 1984.

[7] M. Demirbas, O. Soysal, and A. S. Tosun. Data salmon: A greedy mobile basestation protocol for efficient data collection in wireless sensor networks. In *IEEE International Conference on Distributed Computing in Sensor Systems*, 2007.

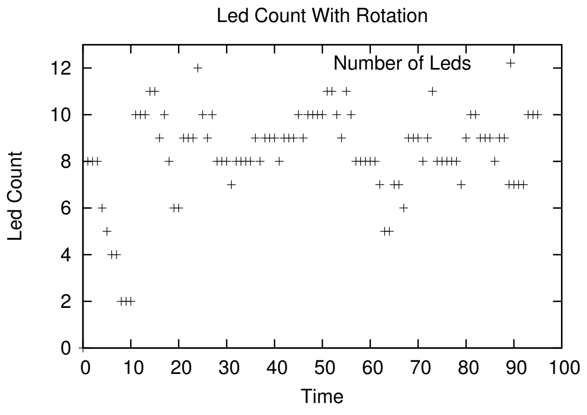
[8] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[9] I. Getting. The global positioning system. *IEEE Spectrum*, 1993.

[10] S. Golomb and H. Taylor. Constructions and properties of Costas arrays. *Proceedings of the IEEE*, 72(9):1143–1163, Sept. 1984.



(a) Location of Mobile



(b) Number of Leds in Window

Figure 10. Proposed Scheme

[11] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C. Lee. Partitioning based mobile element scheduling in wireless sensor networks. In *IEEE SECON*, pages 386–395, 2005.

[12] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In 5^{th} *ACM International Conference on Mobile Computing and Networksing*, 1999.

[13] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *IEEE International Conference on Distributed Computing in Sensor Systems*, 2005.

[14] P. Juang, H. Oki, and Y. Wang. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In 10^{th} *International Conference on Architectural Support for Programming languages and Operating Systems.*, October 2002.

[15] J. C. Lee. Hacking the nintendo wii remote. *IEEE Pervasive Computing*, 7(3):39–45, 2008.

[16] M. Ma and Y. Yang. Sencar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1476–1488, October 2007.

[17] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM Int. Workshop on Wireless Sensor Networks and Applications*, pages 88 – 97, 2002.

[18] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location support system. In 6^{th} *ACM International Conference on Mobile Computing and Networking*, August 2000.

[19] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 30–41, 2003.

[20] T. Small and Z. Haas. The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way). In *ACM MobiHoc*, pages 233–244, 2003.

[21] A. Somasundara, A. Ramamoorthy, and M. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 296–305, 2004.

[22] T. Stoyanova, F.Kerasiotis, A.Prayati, and G. Papadopoulos. Evaluation of impact factors on rss accuracy for localization and tracking applications. In *Proceedings of the 5th ACM international workshop on Mobility management and wireless access*, pages 9–16, 2007.

[23] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems*, pages 51–63, 2005.

[24] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 154–165, New York, NY, USA, 2005. ACM.

[25] K. Whitehouse, C. Karlof, and D. Culler. A practical evaluation of radio signal strength for ranging-based localization. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 11, pages 41–52, Jan. 2007.

[26] G. Xing, T. Wang, Z. Xie, and W. Jia. Rendezvous planning in mobility-assistedwireless sensor networks. In *28th IEEE International Real-Time Systems Symposium*, pages 311–320, 2007.

[27] W. Zhao and M. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 308– 314, 2003.