Progress In Electromagnetics Research C, Vol. 52, 13–25, 2014

Real Time RSS Based Adaptive Beam Steering Algorithm for Autonomous Vehicles

Mohamed A. Ibrahim and Mohammad S. Sharawi*

Abstract—A real time, low complexity algorithm is developed to steer a planar patch antenna array beam to the maximum received signal strength (RSS) direction for communication link enhancement. The beam steering towards the maximum incoming signal direction is based on an iterative technique utilizing a set of RSS measurements taken from specific locations in the search space, these locations collectively form an ellipse. The algorithm is denoted as "elliptical peeking". It was simulated for a flying unmanned aerial vehicle (UAV) as the vehicle tries to identify the maximum signal strength incoming direction of a stationary ground signal and it was tested on an embedded platform to validate its low demand for computational power. Such an algorithm is suitable for autonomous platforms due to its simplicity and low cost.

1. INTRODUCTION

Antenna arrays are widely used for beam focusing and steering [1]. This is done by controlling the amplitude and the phase of the array elements in order to point the beam to a predefined point in space (beam target). If the beam target changes its location, the beam must be readjusted to the new location. The problem of estimating the location of the beam target is a popular one which is tackled using several different techniques. Such techniques were summarized and presented in [2, 3]. Each technique is based on a certain signal parameter being measured. The most common of these signal parameters are the angle of arrival (AOA) [4], time of arrival (TOA) [5], time difference of arrival (TDOA) [6], and the received signal strength (RSS) [7–13].

AOA, TOA, and TDOA usually require multiple receivers (signal detectors) distributed in space in order to identify the location of the signal source [14]. RSS on the other hand can be implemented using a single receiver. The RSS is known to be a noisy signal parameter [14] which sometimes requires channel estimation [7]. Thus for a highly dynamic communication channel such as the one between an unmanned aerial vehicle (UAV) and its base station, such estimation will be difficult to be performed in real time.

At any given time instant, the UAV will have a certain heading for its antenna beam. If the beam heading is misaligned with the direction of the maximum RSS, then this misalignment will cause degradation in the wireless signal strength. The objective of the proposed algorithm is to estimate the beam heading angles (θ_b, ϕ_b) in real time and with low complexity in order to steer the antenna beam to the direction of the maximum RSS. Notice that for a highly dynamic communication channel, the RSS distribution in space will change frequently. Thus the algorithm is required to be fast enough to cope with this dynamic behavior. Also it is required to be simple enough to be executed by a small hardware platform that can be installed on a low payload UAV.

In this work, the proposed elliptical peeking (EP) algorithm will conduct a series of RSS measurements and according to these measurements it will produce an estimate for the best beam

Received 4 May 2014, Accepted 19 June 2014, Scheduled 14 July 2014

* Corresponding author: Mohammad S. Sharawi (msharawi@kfupm.edu.sa).
The authors are with the Electrical Engineering Department, King Fahd University of Petroleum and Minerals (KFUPM), Dharan 31261, Saudi Arabia.

heading angles (θ_b, ϕ_b) which produces the maximum achievable RSS. RSS measurements have a variance that increases with the range [14]. Localization algorithms based on RSS are greatly dependent on the propagation model used. Usually it is required to validate the propagation model by tuning the path loss exponent (distance-power gradient). This is done by characterizing the situation through a lot of RSS measurements. Thus a practical implementation is a must in this case. Since we have a dynamic channel, it is not practical/possible to characterize the link via measurements to develop an accurate propagation model.

In [10], different RSS based localization methods were compared in terms of the calibration effort required by each method, where range-based algorithms were found to outperform fingerprinting and proximity-based algorithms. In [11], the localization was based on RSS, yet it was utilizing multiple reference stations like [15]. In [16] the characteristics of an RSS signal were studied and characterized, where it was obvious that the RSS value will change dramatically by changing the environmental circumstances. In [9], a successful attempt was made to estimate the path loss exponent jointly along with the location estimation process. However, this estimation procedure is a computationally intensive task; hence it is not suitable for real time applications.

In this work, we propose and compare two techniques utilizing the RSS readings in identifying the maximum signal strength direction for adaptive beam steering applications. They are the elliptical Peeking (EP) and the differential evolution (DE) algorithms. The EP algorithm is proposed as the main contribution in this work while the DE algorithm is only used to benchmark the performance of the EP algorithm. Both techniques will require a simulation environment that will allow us to calculate the RSS at certain points based on a realistic virtual trajectory for a UAV. Every few seconds or when the RSS drops significantly, the array goes into beam alignment mode, where it adaptively calculates the direction of maximum signal strength and steers the beam accordingly to improve the communication link.

2. RSS BASED LOCALIZATION ALGORITHMS

In this work, we introduce an RSS based Beam steering algorithm. For the flying/moving vehicle at any time instant the values of the RSS will have a different distribution in space. To clarify the situation let's consider a certain time instant and measure the RSS value in all directions around the vehicle's antenna array. Such exhaustive scan will produce the power distribution shown in Figure 1 at a distance of 492.4 m from the base station in a clear sky condition.

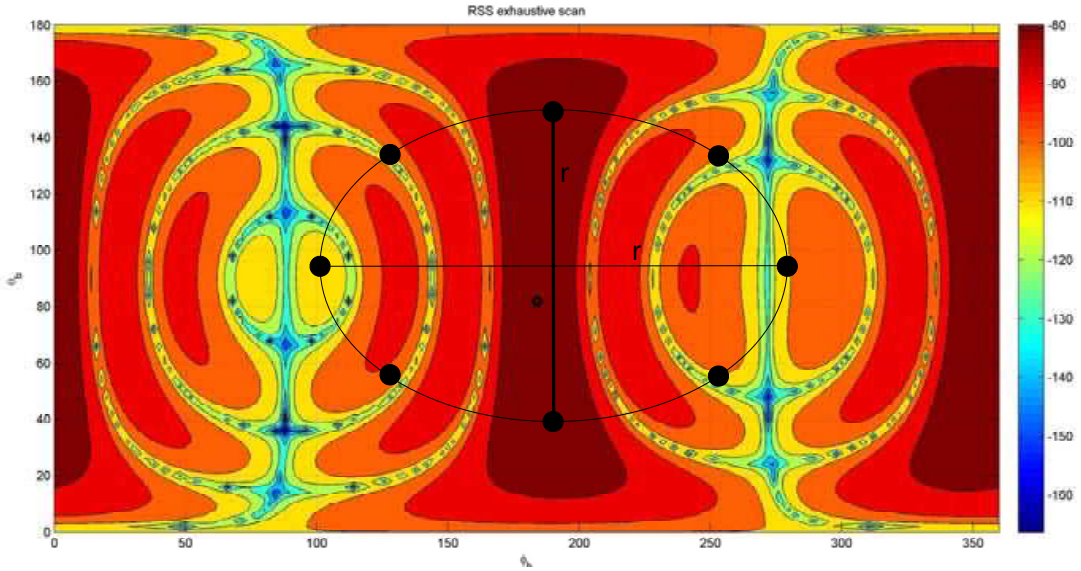

Figure 1. Contour plot of RSS distribution in the θ-ϕ space at a certain time instant. The black circle denotes the measured RSS value. The color bar represents the RSS values in dBs.

Such exhaustive scan is produced with 1° resolution. Thus 64800 (180×360) readings are required. Since one RSS measurement takes 5 ms [17] on average, then by the time one complete scan is produced ($\frac{64800*5}{1000*60} = 5.4$ mins), the vehicle would have a totally different location and attitude where the produced scan would be inappropriate. Hence we need an algorithm that utilizes the least number of readings to accurately localize the global RSS maximum value or get as close as possible to it in a short period of time.

2.1. Elliptical Peeking Algorithm (EP)

By directing the beam towards a certain point in the $(\theta - \phi)$ space and recording its RSS, we are peeking at the RSS distribution through this point. After a number of peeks we might have enough information to localize the maximum RSS. The points at which we are peeking are divided into groups, where each group of points forms an elliptical shape, hence the name elliptical peeking (EP).

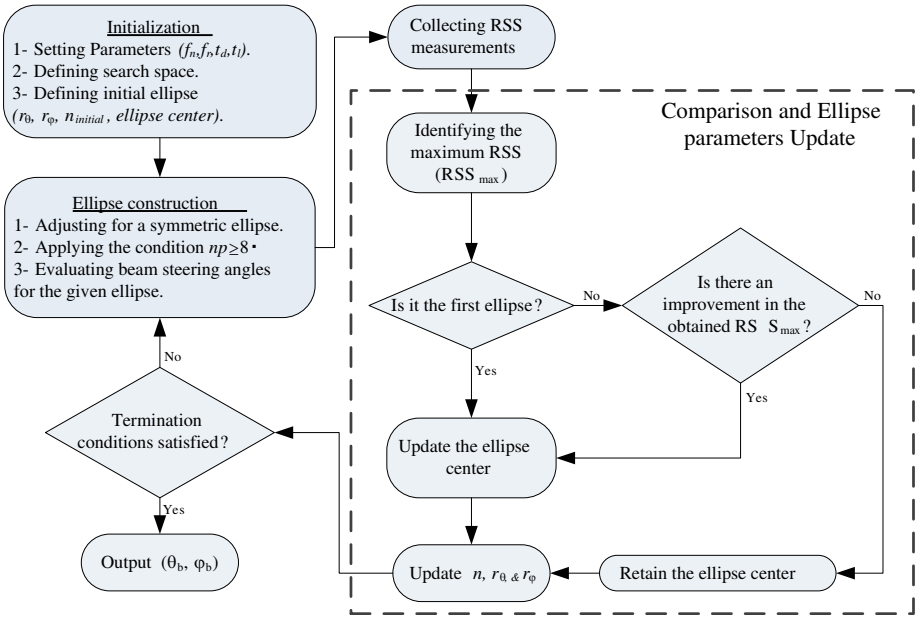

Figure 2. Flow chart of the elliptical Peeking (EP) algorithm.

The flow chart of the algorithm is shown in Figure 2. After the algorithm initializes, it constructs the ellipse points at which the RSS readings will be collected, and then it reads them all and saves them in memory. It then compares all of these RSS values and picks the greatest among them and sets its corresponding beam steering angles (θ_b, ϕ_b) as the center of the next ellipse. But before it goes into the loop again, it checks for the termination conditions in order to decide whether to complete the search process or to stop and produce its final (θ_b, ϕ_b) estimate. The algorithm can run during the time periods at which the transmitter is idle in order to switch to the receiving mode and extract the necessary RSS readings required for the search, keeping in mind that the ground station is sending a continuous stream of signals all the time to allow the UAV to extract RSS readings at any instant. Forming the elliptical points is done via a beam steering algorithm that changes the array beam directions towards the calculated (θ_b, ϕ_b) angles, then it records the RSS measured taking into account the prior knowledge of the array gain in that direction. A hardware implementation of a planar array for such application is shown in [18, 19].

2.1.1. Initialization

The initialization starts the EP algorithm by executing three consecutive procedures. The first procedure sets the values of the algorithm parameters (f_n, f_r, t_d, t_l). These four parameters define how the

algorithm will execute. They can be tuned for better performance in different scenarios. The values used were achieved by extensive simulations and were 0.5, 0.8, 0.4 dB, and 2 s respectively.

The factor f_n is the reduction factor of n, where n is the number of ellipse points. This is the factor by which the number of ellipse points will be reduced every iteration. f_r is the r_ϕ reduction factor by which the ellipse radius r_ϕ will be reduced every iteration. Note that the ellipse has two radii (r_ϕ and r_θ) as shown in Figure 1. t_d is the RSS termination difference and it is used as a termination condition when the improvement in the RSS is less than t_d. t_l represents the time limit at which the algorithm will terminate regardless of any other factor and produce its last estimate as its best estimate. This condition protects the algorithm from infinite loops.

The second procedure in the initialization process is to define the search space. This is an easy step where all we have to do is to set the values for $\theta_{\max}$, $\theta_{\min}$, $\phi_{\max}$ and $\phi_{\min}$.

The limits of the search space are defined by inequalities (1) and (2). The search space can be reduced by 1/2 if only one hemisphere is of interest.

$$0^\circ \leq \theta \leq 180^\circ \tag{1}$$

$$0^\circ \leq \phi \leq 360^\circ \tag{2}$$

The third and final procedure in the initialization process that defines the initial ellipse. The ellipse can be defined by four parameters, the ellipse center, the major radius r_ϕ, the minor radius r_θ, and the number of points n on the ellipse edge. These parameters are initially defined according to the minimum half power beam width ($HPBW$) of the utilized antenna array. This is done to make sure that the first ellipse will have at least one point inside the footprint of the antenna's main lobe. Equation (3) defines the initial value of r_ϕ. Equation (4) can be used consequently to calculate the initial value of r_θ, where the ratio stated in this equation must hold through all the iterations. We also use Equation (4) to calculate r_θ in the middle of the algorithm execution whenever r_ϕ changes as a result of being multiplied by the reduction factor f_r. Equation (5) is used to calculate the initial value of n.

$$r_{\phi_{initial}} = \frac{1}{2}\left[\phi_{span - \frac{HPBW_{\min}}{2}}\right] \tag{3}$$

$$\frac{r_\theta}{r_\phi} = \frac{\theta_{span}}{\phi_{span}} \tag{4}$$

$$n_{initial} = \frac{r_{\phi_{initial}}}{\frac{1}{2}HPBW_{\min}} \tag{5}$$

2.1.2. Ellipse Construction

Ellipse construction is the process of adjusting the number of ellipse points and generating their corresponding beam steering angles (θ_b, ϕ_b). The first step is to make sure that n is divisible by 4 in order to construct a symmetric ellipse. Thus in this step is approximated to the nearest greater number divisible by 4. The second step ensures $n \geq 8$. The last step is the calculation of the beam steering angles (θ_b, ϕ_b) for all the ellipse points in order to measure the RSS while the beam is steered to each one of these angles. The number of points selection was optimized based the gain map coverage of the array in 3D space as well as the accuracy of locating the maximum power location of the incoming signal.

2.1.3. Comparison and Update

The comparison and update block is shown in Figure 2. After collecting a set of RSS readings we pick the highest value. Around the picked value we construct another (smaller) ellipse and collect another group of RSS readings and also pick the highest. Then we repeat these steps to fine tune the position estimate to an acceptable accuracy.

The EP algorithm is based on an iterative process that starts at the global level and is expected to evolve to close on the global maximum RSS, where further iterations will allow the algorithm to fine tune its estimation and improve accuracy, this will consume more time which is a critical factor due to

the dynamic scenario considered (UAV). A compromise should be made to reach an acceptable accuracy within a reasonably short period of time.

Updating n and r_ϕ is performed according to Equations (6) and (7) where k is the algorithm iteration counter. Updating r_θ is performed after r_ϕ using Equation (4) in order to maintain the same ratio between the ellipse radii.

$$n(k+1) = n(k) * f_n \tag{6}$$

$$r_\phi(k+1) = r_\phi(k) * f_r \tag{7}$$

2.1.4. Termination Conditions

Every iteration the maximum produced RSS is compared with the one produced in the previous iteration in order to check for the termination condition. If the improvement (increase) in the RSS value is more than t_d, then the algorithm decides to proceed further. Otherwise the amount of improvement expected in the next iteration is just not worth the time to be consumed. A dominant condition protects the algorithm from infinite looping by specifying a hard limit ($t_l = 2s$) beyond which the algorithm will stop executing regardless of the latest improvement in the RSS.

2.2. Differential Evolution Algorithm

The differential evolution (DE) algorithm is an evolutionary optimization technique. It is characterized by its simplicity, robustness, fast convergence, and small number of control variables [20]. DE is used in this work only to benchmark the performance of the introduced EP algorithm.

3. SIMULATION RESULTS

In this section, we will present the simulation results of both algorithms, and evaluate their performance by simulating a large number of flight trajectories (500) considering two scenarios. The first scenario employs a smooth flight trajectory, while the second scenario employs a turbulent flight trajectory with a lot of tight maneuvers. This is done to assess the algorithm performance in different flying conditions. The performance evaluation is based upon the 3 different parameters. The first parameter is the RSS estimation error (RSS_e) which is the difference between the maximum achievable RSS at the given instant (using a perfectly aligned beam) and the RSS achieved by the algorithm estimated steering angles. In other words the maximum possible value for RSS_e is 0 which indicates a perfectly aligned beam. The second parameter is the time of convergence (TOC) which is the time taken by the algorithm to converge to its optimum estimation for the steering angles (θ_b, ϕ_b). The third parameter is the error in estimating such angles.

In Figure 3 the smooth and the turbulent flight trajectories are shown on the same scale for comparison. Such flight trajectories were generated using an open source flight simulator called "*FlightGear*" [21] which was also used in [22]. After generating the full trajectory in the flight simulator, it was imported into MatLab [23] to be chopped into 500 small trajectories which are then used in testing both algorithms in order to produce a performance evaluation in an aggregate statistical form.

3.1. EP Algorithm

In Figure 4 the (RSS_e) values for the EP algorithm in both trajectories (smooth & turbulent) are shown. The average error values were -0.24 dB for the smooth and -2.45 dB for the turbulent trajectory. It is worth mentioning that more than 94% of the runs have errors within -0.84 dB.

Figure 5 shows the Time of convergence (TOC) for the 500 runs of the EP algorithm in both trajectories. The average TOC 431 ms is for the smooth while it was 423 ms for the turbulent trajectory. Note that more than 99% of the 500 runs converged within 55 ms.

In Figure 6 and Figure 7 the error in estimating the beam steering angles θ_b and ϕ_b respectively in both trajectories are shown. In the smooth trajectory the average error in the angle θ_b ($\bar{\theta}_e$) was 3.5°, while this value goes up to 8° in the turbulent trajectory. To explain this increase, note that in the turbulent trajectory the average rolling angle increases, and consequently the actual θ_b will fluctuate

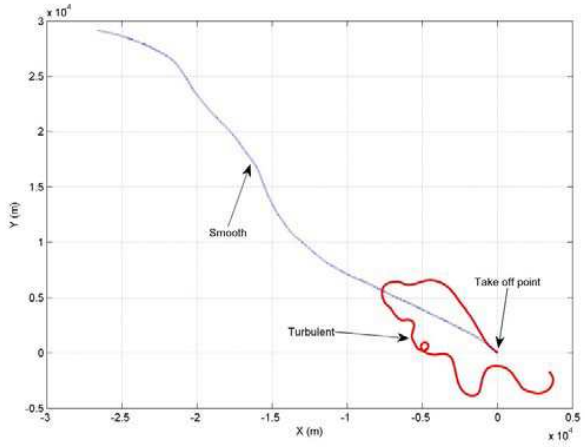

Figure 3. Smooth vs. Turbulent flight trajectories (top view).

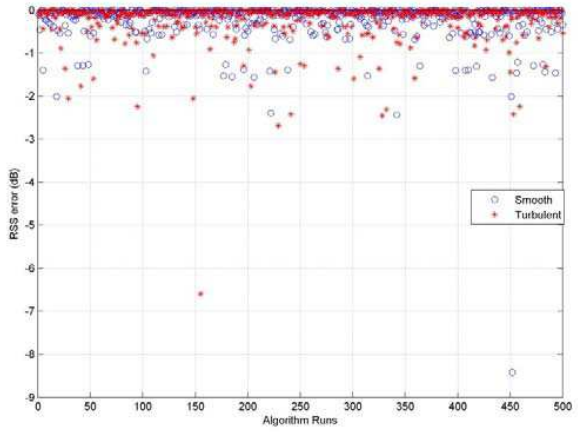

Figure 4. (RSS_e) using the EP algorithm for both trajectories.

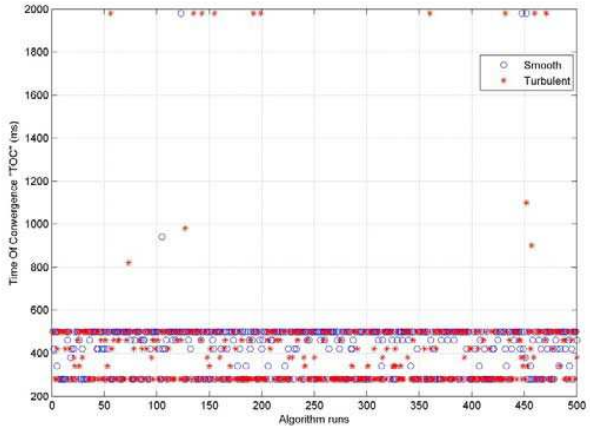

Figure 5. (TOC) for the EP algorithm in both trajectories.

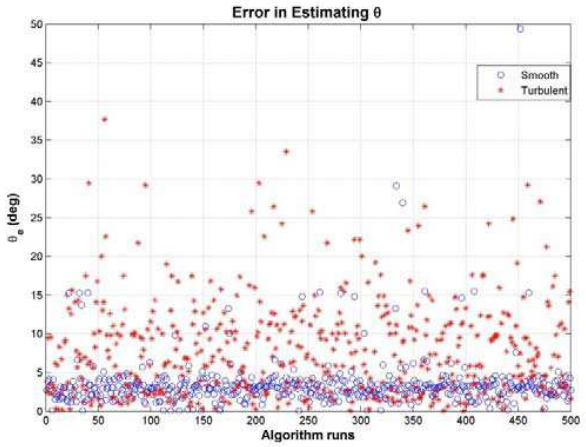

Figure 6. Error in estimating θ_b in both trajectories for the EP algorithm.

inside the θ space ($0° \leq \theta \leq 180°$), which will result in a bigger average for the half power beam width in the θ direction ($HPBW_\theta$). Thus, estimation errors will increase accordingly. In order to visualize this situation we must refer to the $HPBW_\theta$ contour plot in [19].

The average error in estimating the error in the angle ϕ_b is similarly denoted ($\overline{\phi_e}$). By considering only the smooth trajectory and comparing $\overline{\theta_e}$ with $\overline{\phi_e}$, we notice that $\overline{\phi_e} = 3.5°$ is much less than $\overline{\phi_e} = 72.5°$. This can be easily explained by recalling the fact that the used planar 2×6 antenna array presented in [19], has a $HPBW_\theta$ much smaller than that in the ϕ direction ($HPBW_\phi$). This will result in higher localization accuracy in the θ axis and a less localization accuracy in the ϕ axis.

By carefully observing the simulation results, we deduced that errors in estimating the steering angles don't degrade the RSS because at low elevation angles a secondary beam that is opposite to the main one arises, in addition, the beam has a wide footprint within which the gain doesn't have considerable changes when considering a 2×6 planar array that is to be embedded within the wing structure of a UAV. So as long as the maximum RSS point falls inside the footprint, the RSS value will be acceptable and the algorithm will terminate itself.

If we were to use an antenna array with low HPBW then any small error in the steering angles will have a great effect on the RSS value, because in this case the footprint will be small. This can be utilized for localization or navigation reasons, but one has to keep in mind that the TOC in this case

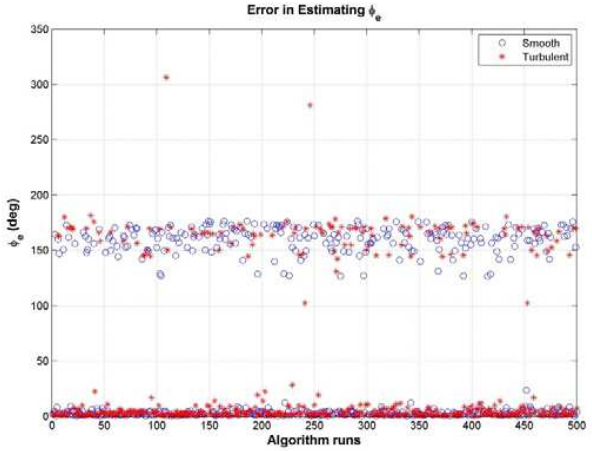

Figure 7. Error in estimating ϕ_b in both trajectories for the EP algorithm.

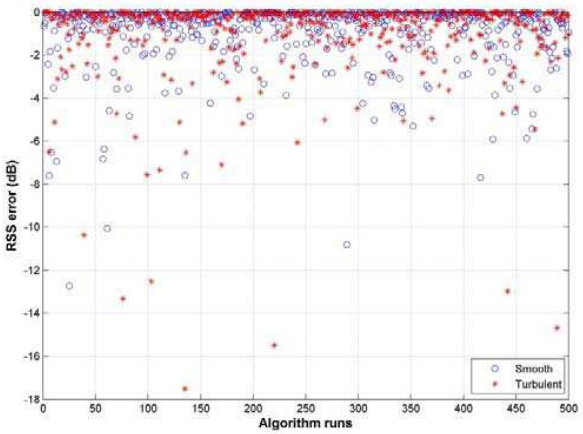

Figure 8. (RSS_e) in the DE algorithm for both trajectories.

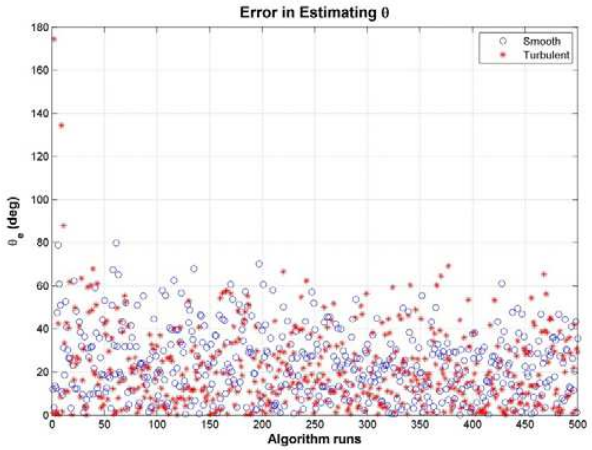

Figure 9. Error in estimating θ_b in both trajectories for the DE algorithm.

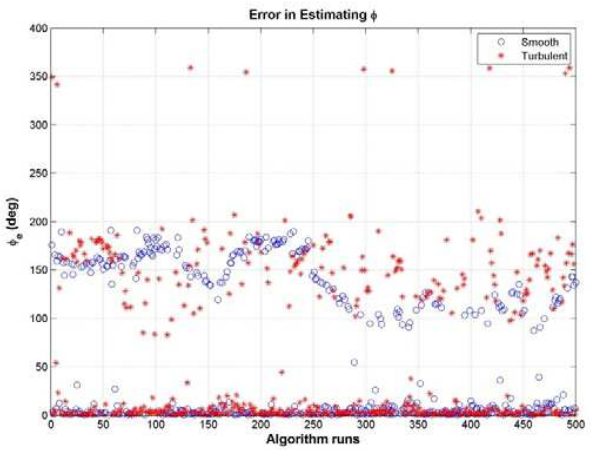

Figure 10. Error in estimating θ_b in both trajectories for the DE algorithm.

might increase.

By comparing the results of the smooth trajectories to the turbulent trajectories, we can notice that $\overline{\phi_e}$ decreased from 72.5° in the smooth to 41.6° in the turbulent. This can be explained by the fact that for a turbulent trajectory, the aircraft's rolling angle will have a relatively high average value during most of the flight time. This will lead to a less possibility of having the secondary beam instead of the primary pointing at the maximum RSS. Thus fewer values will be around 160° in the turbulent trajectory compared to the smooth one. This is noticed clearly by comparing the number of red stars to the number of blue circles around in Figure 7.

3.2. DE Algorithm

In this section the simulation results of the DE algorithm are presented, and its performance is evaluated based on the same flight trajectories used to evaluate the EP algorithm. Similarly the same two scenarios are considered. Note that the TOC of the DE algorithm is fixed at 500 ms by setting the population size to 20 and letting the populations evolve for only 5 generations. Since one RSS reading consumes typically 5 ms [17], therefore $TOC = 20 \times 5 \times 5\,\text{ms} = 500\,\text{ms}$.

Notice that in Figure 8 the error pattern is scattered compared to those produce by the EP algorithm in Figure 4. This is due to the randomness nature of evolutionary algorithms in general. In other words each run will produce totally different results yet all the runs will maintain the same statistical

Table 1. Simulation results comparison.

	$\overline{RSS_e}$		TOC		$\overline{\theta_e}$		ϕ_e	
	Smooth	Turbulent	Smooth	Turbulent	Smooth	Turbulent	Smooth	Turbulent
EP	-0.244 dB	-0.245 dB	431 ms	423 mx	3.5°	8°	72.5°	41.7°
DE	−0.991 dB	−0.996 dB	500 ms	500 ms	23.7°	21.4°	59.1°	61.6°

characteristics.

Table 1 summarizes all the parameters recorded in the simulations and compares the performance of the EP with the DE.

For the smooth trajectory when we compare the DE algorithm to the EP algorithm in terms of θ_e, we notice that its value for the EP was 3.5° and it increased for the DE to 23.7°. This is due to the randomness nature of the DE algorithm where each generation is produced randomly from its ancestor. Thus the trend in updating θ_b values need not be necessarily improving, and that will produce large variance in estimated θ_b values in the DE Algorithm (see Figure 9).

Also the average value for ϕ_e in the EP was 72.5° compared to 59.1° for the DE. This reduction is due to the randomness nature of the DE algorithm. By observing the error trends in Figure 7 we can see that the values are more gathered around 160° which was not the case in Figure 10, instead they are scattered and this is why the average ϕ_b error value decreased in the DE algorithm compared to the EP algorithm.

The randomness nature of the DE algorithm is also the reason for getting almost the same performance in the turbulent (−0.97 dB) compared to the smooth trajectory (−0.99 dB). This is also is the case for the EP algorithm where its average RSS error in the smooth trajectory was −0.244 dB, and in the turbulent trajectory was −0.245 dB, but in the EP algorithm this observation is explained by the randomness of the trajectories used to simulate the algorithm. In other words, each time the algorithm is executed, the trajectory used is picked randomly from the 500 trajectories population. This was done to give a more comprehensive statistical sense of the algorithm performance.

4. HARDWARE IMPLEMENTATION

In this section, the results of loading and executing the EP algorithm on an embedded hardware platform are shown to validate its low computational power demands. The hardware platform used was the Arduino DUE [24] with the AT91SAM3X8E microcontroller chip running at 84 MHz. The code for the EP algorithm occupied only 23 kB (4%) of the program memory size.

The flow chart in Figure 2 was coded as it is except the RSS measurements collection block where it was replaced by an intentional delay introduced in the code to simulate the typical hardware delay for each RSS reading (5 ms) [17]. Also a random value for the RSS was generated and assigned as the current reading. This is acceptable because we are interested in assessing the execution speed of the code and not its accuracy.

By running the algorithm code repeatedly and forcing it to terminate after a pre-specified number of iterations, we will be able to expect the execution time of one run of the code by simply adding up all the delays. Every time the code terminates, an output pin is toggled to bookmark the new run. This will result in producing a square wave whose pulse width represents the actual execution time of the algorithm. Then we compare the measured pulse width to the expected pre-calculated execution time. If both time periods are closely matched, then we conclude that the algorithm is running smoothly and in real time.

Three different attempts were recorded with their corresponding measured waveforms. The first one whose waveform is shown in Figure 11 (S_1) had 10 ellipses, each ellipse had 8 points, and each point is an RSS reading whose delay is 5 ms. Thus the total expected execution time should be $10 * 8 * 0.005 = 0.4$ s. By noticing the measured waveform (S_1) we can see that the pulse width is exactly 0.4 s, hence we can conclude that no other code delays are affecting the execution except the introduced intentional delay (5 ms/reading). Thus for the given hardware platform, the algorithm is running smoothly in real time.

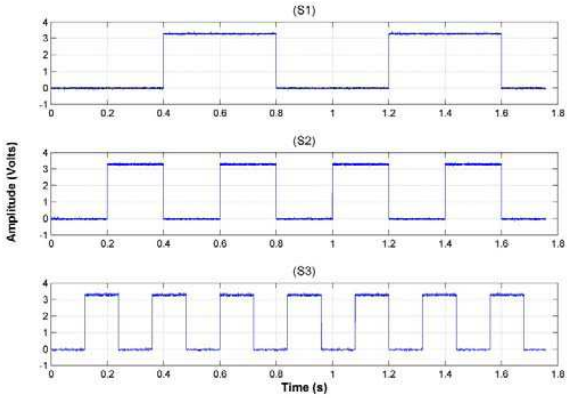

Figure 11. Measured waveforms.

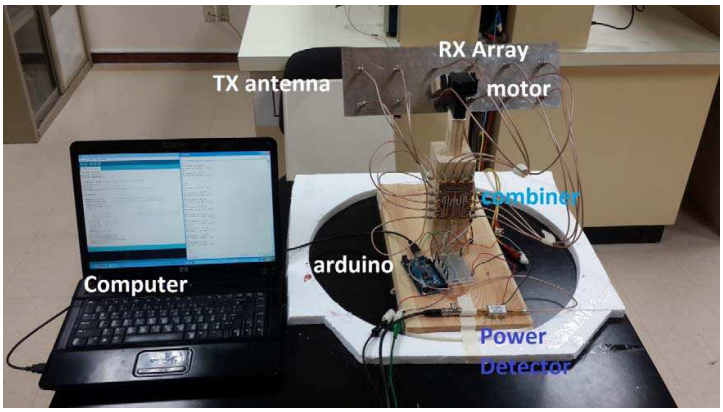

Figure 12. Experimental setup of the EP algorithm.

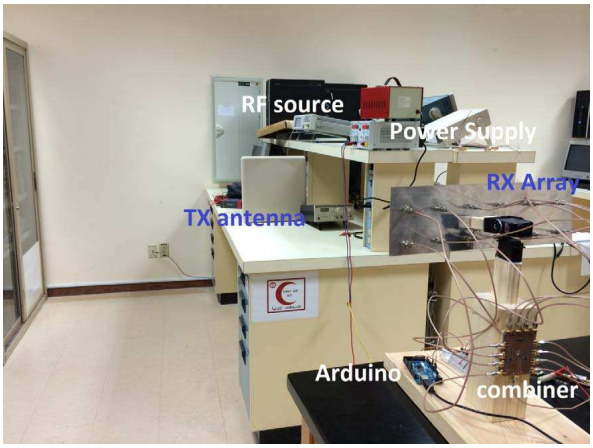

Figure 13. Experimental setup showing the various components of the system.

Two other attempts were made. The first one contained 5 ellipses, thus the expected execution time is $(5 * 8 * 0.005 = 0.2\,\text{s})$. The corresponding measured waveform is (S_2) whose pulse width is also 0.2 s. The second one contained 3 ellipses, thus the expected execution time is $(3 * 8 * 0.005 = 0.12\,\text{s})$. The corresponding measured waveform is (S_3) whose pulse width is also 0.12 s. Thus our former observation is confirmed.

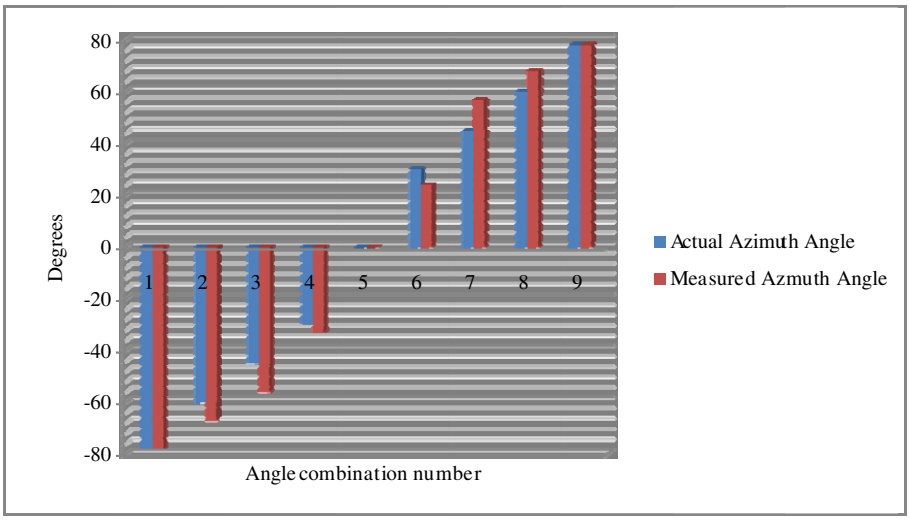

Figure 14. Comparing the actual and the obtained direction results for azimuth changes.

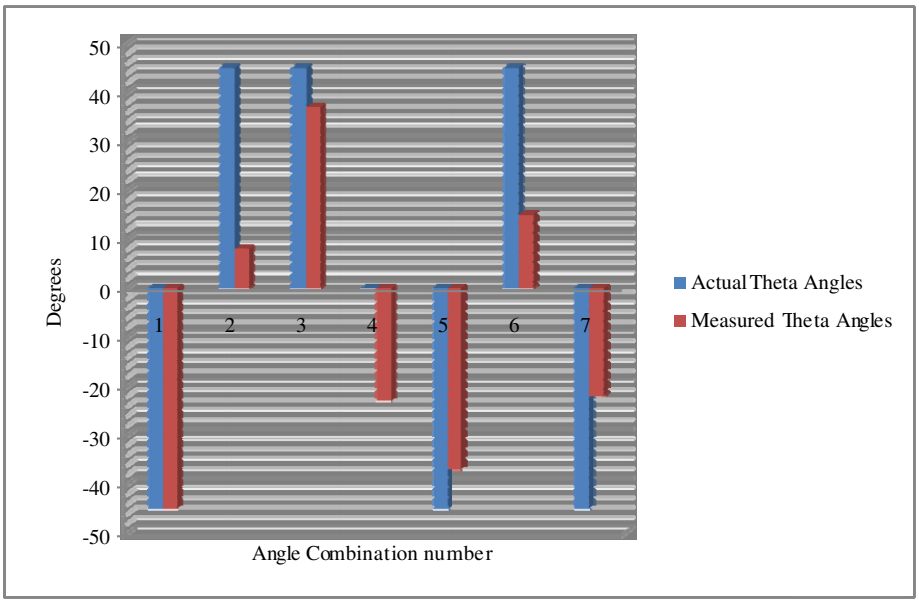

Figure 15. Comparing the actual and the obtained direction results for elevation changes.

To verify the algorithm operating on an array, the setup in Figure 12 is built. Due to the unavailability of a phased array design, a mechanical system with a two axis rotation motor was built to proof the concept of the algorithm and show its accuracy. Although the response time of this configuration will not be as fast as the one mentioned earlier, we are here trying to validate the concept with the hardware available. The figure shows a fixed beam antenna array with the same number of antenna elements as algorithm assumes (i.e., with 12 patch elements) with its beam pointing at the normal to its surface (i.e., towards $\theta = 0$, $\phi = 0$ direction). The 12-outputs of the array are connected to a 12-port combiner board that is directly connected to an RF power detector (ZX47-40LN-S+). The DC output from the power detector is mapped to the appropriate power levels as per the manufacturer specifications. The DC voltage is passed to an analog-to-digital (ADC) converter power at the Arduino microcontroller board and then interfaced to a PC that displays the output of every measurement during each loop. The location of the array is changed mechanically using a two axis motor (HS-422) that is performing the algorithm mechanically. At the end of the algorithm, the array will point to the maximum power location. Figure 13 shows another angle for the system in action.

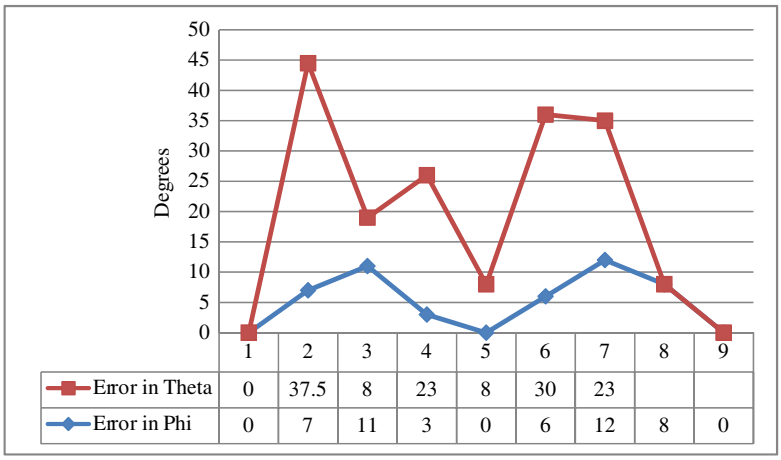

Figure 16. Error curves for the various cases in elevation and azimuth.

The accuracy in the azimuth direction was much better located compared to the elevation direction due to the narrower HPBW in the azimuth plane. Figure 14 shows the results of 9 different angle combinations shows in Table 2 when the elevation angle was fixed. Figure 15 shows a similar chart when the elevation angles were changed but for 7 cases only (also listed in Table 2). Each angle combination case was repeated three times to verify algorithm convergence and repeatability results. Figure 16 summarizes the error levels obtained for the cases tested. It is clear that the algorithm can accurately track and locate the direction of the maximum incoming power direction in the azimuth plane below the wing of the UAV, while the errors in the elevation plane are more significant. Although the errors are high in the elevation plane, this should not compromise the accuracy of the system as the wide beam width in that direction will allow for good beam steering towards the base station with wider coverage spots. The low complexity of the algorithm allows it to be easily programmed on a commercial small size microcontroller and placed on an unmanned vehicle.

It should be noted that minimizing the number of points within the RSSI based algorithm might not necessarily degrade the system accuracy in pointing the antenna beam towards the maximum incoming power location, as the HPBW of the array is wide enough to provide acceptable alignment.

Table 2. Angle pair for the two cases, azimuth changes and elevation changes (θ, ϕ).

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
Azimuth changes	$(0, -78)$	$(0, -60)$	$(0, -45)$	$(0, -30)$	$(0, 0)$	$(0, 30)$	$(0, 45)$	$(0, 60)$	$(0, 78)$
Elevation changes	$(-45, -45)$	$(45, -45)$	$(45, 0)$	$(0, 0)$	$(-45, 0)$	$(45, 45)$	$(-45, 45)$	-	-

5. CONCLUSIONS

In this work an RSS based beam steering algorithm is proposed and compared to an evolutionary algorithm in terms of accuracy and execution time. Both algorithms were evaluated based on two UAV trajectories, a smooth and a turbulent trajectory. The proposed Algorithm denoted as (EP) showed better results in terms of RSS beam miss-alignment errors as well as the steering angles estimation errors compared to the evolutionary one. The algorithm performance in localization applications is highly dependent on the radiation pattern of the utilized antenna array. The algorithm was tested on an embedded platform and its low computational power as well as accuracy in locating the direction of the maximum incoming power demand was validated.

ACKNOWLEDGMENT

This work was supported by King Abdulaziz City for Science and Technology (KACST) through the Science and Technology unit at King Fahd University of Petroleum and Minerals (KFUPM) with project No. 11-ELE2150-04 as part of the national science, technology and Innovation plan (NSTIP). Also the authors would like to thank H. Alrubaii, O. Alhobail, A. Alhayem and A. Alyami, for their help in the measurement setup.

REFERENCES

1. Scott, H. and V. F. Fusco, "Antenna array beam-steering by the integration of a series phase shifter," *6th IEEE High Frequency Postgraduate Student Colloquium*, 25–29, 2001.
2. Munoz, D., F. Bouchereau, C. Vargas, and R. E. Caldera, *Position Location Techniques and Applications*, Elsevier Inc., United Stated of America, 2009.
3. Yu, K., I. Sharp, and Y. J. Guo, *Ground Based Wireless Positioning*, Wiley, United Kingdom, 2009.
4. Amundson, I., X. Koutsoukos, J. Sallai, and A. Ledeczi, "Mobile sensor navigation using rapid RF-based angle of arrival localization," *7th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS)*, 316–325, 2011.
5. Izquierdo, F., M. Ciurana, F. Barceló, J. Paradells, and E. Zola, "Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN," *2006 1st International Symposium on Wireless Pervasive Computing*, 1–6, 2006.
6. Yoon, J., J.-W. Kim, W.-H. Lee, and D.-S. Eom, "A TDoA-based localization using precise time-synchronization," *2012 14th International Conference on Advanced Communication Technology (ICACT)*, 1266–1271, 2012.
7. Tarrio, P., A. M. Bernardos, and J. R. Casar, "An RSS localization method based on parametric channel models," *International Conference on Sensor Technologies and Applications, SensorComm 2007*, 265–270, 2007.
8. Capkun, S., S. Ganeriwal, F. Anjum, and M. Srivastava, "Secure RSS-based localization in sensor networks," ETH Department of Computer Science, Zurich, 2011.
9. Li, X., "RSS-based location estimation with unknown pathloss model," *IEEE Transactions on Wireless Communications*, Vol. 5, No. 12, 3626–3633, 2006.
10. Dil, B. J. and P. J. M. Havinga, "An the calibration and performance of RSS-based localization methods," *Internet of Things (IOT)*, 1–8, 2010.
11. Pathirana, P. N., A. N. Bishop, and A. V. Savkin, "Localization of mobile transmitters by means of linear state estimation using RSS measurements," *10th International Conference on Control, Automation, Robotics and Vision, ICARCV*, 210–213, 2008.
12. Bianchi, G., N. B. Melazzi, and F. L. Piccolo, "Impact of chosen error criteria in RSS-based localization: Power vs distance vs relative distance error minimization," *12th IEEE Symposium on Computers and Communications, ISCC*, MW–9, 2007.
13. Paul, A. S. and E. A. Wan, "RSSI-based indoor localization and tracking using sigma-point Kalman smoothers," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 3, No. 5, 860–873, Oct. 2009.
14. Martin, R. K., A. S. King, R. W. Thomas, and J. Pennington, "Practical limits in RSS-based positioning," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2488–2491, 2011.
15. Tsuji, H., P. Cherntanomwong, and J.-I. Takada, "Experiential evaluation of outdoor radio source localization using spatial information of array," *Asia-Pacific Microwave Conference, APMC 2007*, 1–4, 2007.
16. Wu, R.-H., Y.-H. Lee, H.-W. Tseng, Y.-G. Jan, and M.-H. Chuang, "Study of characteristics of RSSI signal," *IEEE International Conference on Industrial Technology, ICIT 2008*, 1–3, 2008.
17. "XBee®ZB — Digi International," Oct. 9, 2012, Available: http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#specs.

18. Ibrahim, M., S. Deif, and M. S. Sharawi, "A 14-element printed planar antenna array embedded within a UAV structure," *Loughborough Antennas and Propagation Conference (LAPC)*, 1–4, 2012.
19. Sharawi, M. S., M. Ibrahim, S. Deif, and D. N. Aloi, "A planar printed antenna array embedded in the wing structure of a UAV for communication link enhancement," *Progress In Electromagnetics Research*, Vol. 138, 697–715, 2013.
20. Storn, R., "On the usage of differential evolution for function optimization," *Fuzzy Information Processing Society, NAFIPS, Biennial Conference of the North American*, 519–523, 1996.
21. "Introduction | flightgear flight simulator," Nov. 19, 2013, Available: http://www.flightgear.org/about/.
22. Kumar, V., H. Yong, D. Min, and E. Choi, "Auto landing control for small scale unmanned helicopter with flight gear and HILS," *5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, 676–681, 2010.
23. Mathworks Inc., *MATLAB*, Release 2012b. Massachusetts, USA.
24. "Arduino — ArduinoBoardDue," Nov. 18, 2013, Available: http://arduino.cc/en/Main/ArduinoBoardDue.