

Advanced IOT Applications
Dr. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

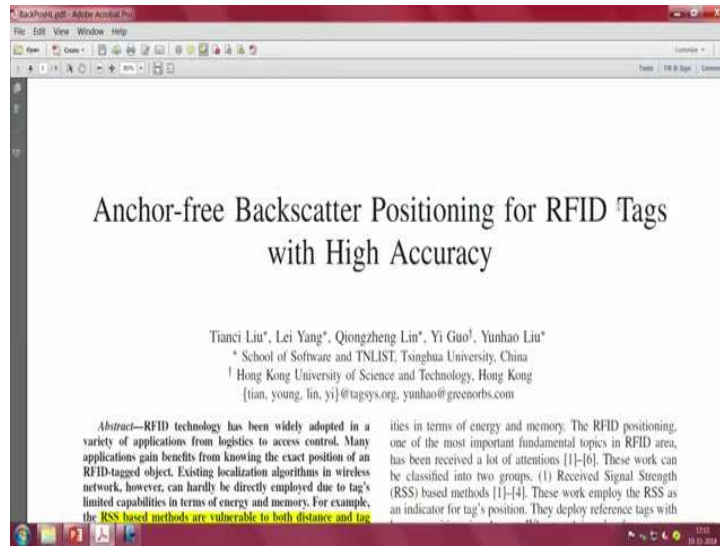
Lecture – 09
RFID based localization – II

So let us keep one practical example in mind. The practical example that you should perhaps concentrate is on a case where you enter a departmental store like Walmart; large warehouses, large wholesale shops like Walmarts or Big Bazaars which are split over several floors.

And you are interested in, sort of a mechanism where at the end of the day, a robotic platform equipped with all the necessary infrastructure for reading tags is made available. And this robotic platform walks across several narrow corridors from rack to rack. There is a narrow corridor; it uses that corridor space, scans on one side for all the items, comes back, scans on all the items on the other side and sort of localizes that particular section of the floor area. And it does this for all sections of the large departmental store.

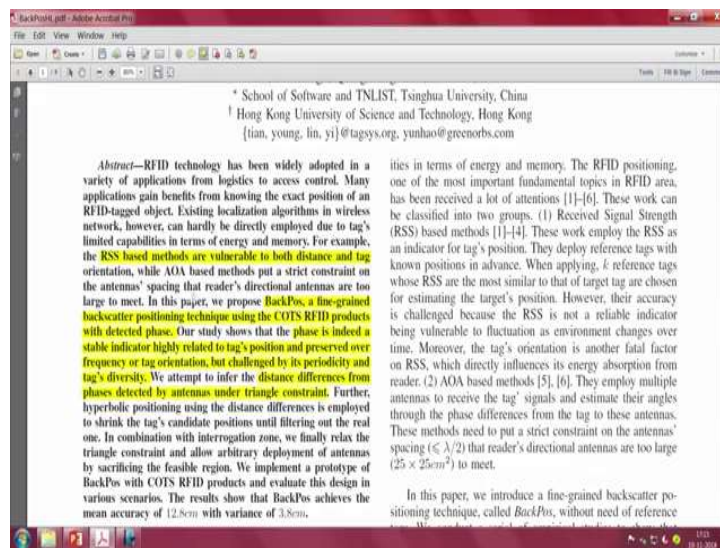
So, we let us keep that as the goal you want to do. We will see if we can show you a very small part of this course; whether we can cover that little demonstration using all the principles that we have learnt for RFID based localization. Again, it is still not complete in terms of our discussion unless we see one paper or one literature work which sort of summarizes all these into a possible implementation of a system.

(Refer Slide Time: 02:24)



Let me point you to a paper that which is quite recently published and this paper is Anchor free Backscatter Positioning for RFID Tags with High Accuracy.

(Refer Slide Time: 02:36)



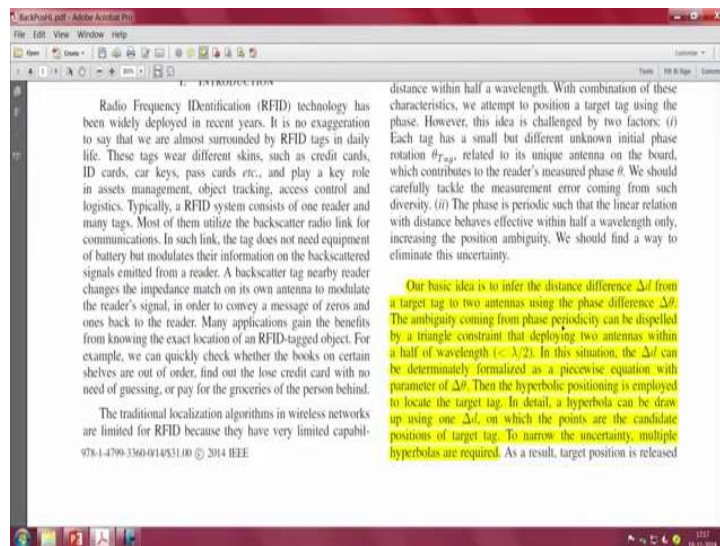
Look at the portions where things are marked that will drive you home when you read this paper.

First thing is this paper dismisses the same problem that we discussed a while ago: RSS based methods are vulnerable. You cannot do anything with respect to distance and tag orientation. So, let us put that out of the picture.

AOA methods put a strict constraint on number of antennas, in terms of antennas spacing that reader's directional antennas are too large to meet. So, this is essentially what we have also described with respect to the different localization methods that are applied in RFID. And so, essentially they are all using angle of arrival based methods only where there are multiple receiver antennas.

So, here again it is again focusing back on phase which is indeed a stable indicator, highly related to tag's position and preserved over frequency or tag orientation, but challenged by its periodicity and tags diversity. So, essentially it is trying to say; let us go and explore phase to the maximum possible extent to see if you can localize these tags, keeping a few constraints in mind.

(Refer Slide Time: 04:07)



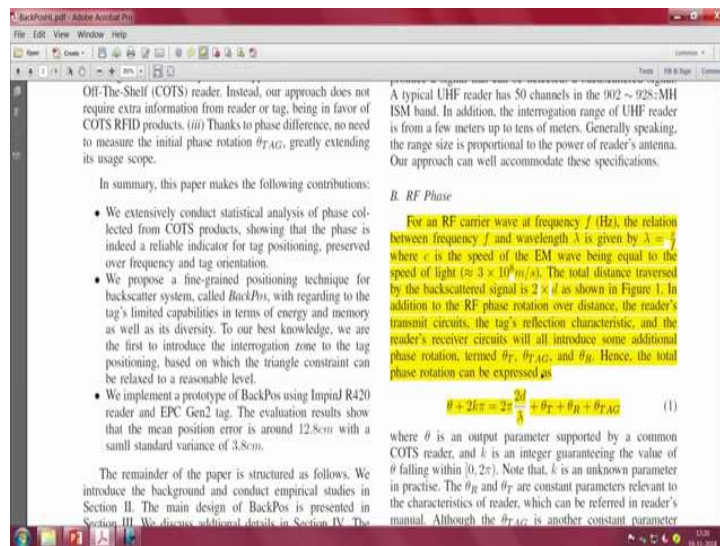
Again you recall the antenna spacing that we discussed in detail; here again it is talking about this antenna spacing should be less than $\lambda/2$ and the idea is that if you want to infer distance difference Δd from a target tag to two antennas using phase difference method ($\Delta\theta$), you must have this $\lambda/2$ constraint applied to this. So; that means, “a” (the term “a” that we remember very well) should be less than $\lambda/2$.

In this situation, the Δd can be determined; determinately formulize as a piecewise equation with parameter $\Delta\theta$. Then the hyperbolic positioning is employed to locate the target tag. So, let us see how this hyperbola can be drawn using one Δd on which the points are candidate positions of a target tag.

When you draw hyperbolas; you read a tag. You are exactly not talking about one single hyperbola, but several hyperbolas which are sort of ideally expected to intersect at one point. Once that all these hyperbolas intersect at one point, that point indeed is the actual (x, y) location of the tag. But far from reality this is never going to happen because when you read the tags; you will form hyperbolas which are all never going to intersect at one point; so you have to find a method to sort of localize them.

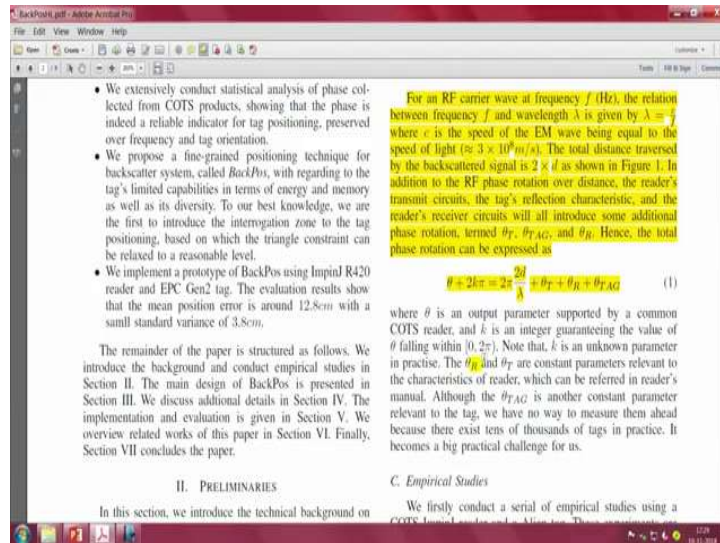
In other words, to reduce the uncertainty you will not only need multiple hyperbolas, but you will also need a good way to arrive at what would have been an actual intersection of all these hyperbolas. We would see that you need a clustering algorithm as well. At the moment; it may sound a little unclear about how all this is to be done, but I am sure as we read this paper you will see clarity emerging from its practical application in order to do localization.

(Refer Slide Time: 06:39)



Now, this is all basics. But do keep in mind that many issues; many parameters are associated with the actual phase that you read which includes the reader's transmit circuits, the tag's reflection characteristics, the reader's receiver circuits and so many other issues. So, essentially there are terms such as θ_T , θ_{TAG} , θ_R . All these are contributing to the actual phase that a reader antenna actually obtains.

(Refer Slide Time: 07:19)



So, all of this will have to be expressed in some form and you can see that this indeed is the simplest way what one can represent. Let us look at this expression

$$\theta + 2k\pi = 2\pi \frac{2d}{\lambda} + \theta_T + \theta_{TAG} + \theta_R$$

$\theta + 2k\pi$ is the phase that you actually get between the tag and the reader. θ_T , θ_{TAG} and θ_R are all the impairments that will get added when you are measuring this actual phase $2\pi \frac{2d}{\lambda}$.

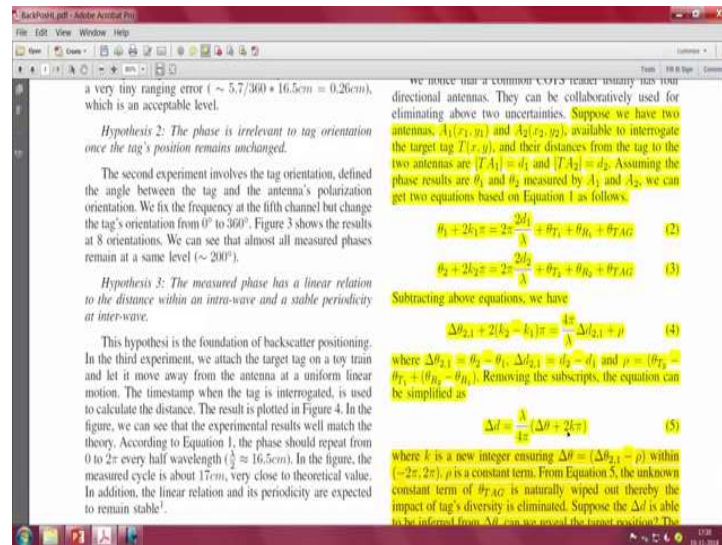
Now, if you take 1 and you take 1.1; supposing you take 1.1; 1.1 is another phase for another distance. You take the reader to another position; you get back the same expression. Right, which is in another position that is why we did 1.1 and you do a phase difference. Now substitute for this, you substitute as θ_1 ; the other this for other separation you substitute as θ_2 then you do $\theta_1 - \theta_2$. It is clear that this part of the expression becomes 0, it gets cancelled out.

So, you are only; so in other words the point that you have to note is that θ_T , θ_{TAG} and θ_R are easy to eliminate if you do phase difference method. This is the basic underlying principle under which this can be calibrated out by simply cancelling out these terms.

Also observe the interval over which θ is falling. Ok, this is very important. The interval is not a completely closed interval. You can see it is closed on the left side. But because of the distance that the tag and antenna (the reader) are situated, the right side is not a closed interval. There can be n rotations after which you can read θ and therefore, this is not really a closed interval.

Recall I mentioned about one equation 1 and equation 2 right. So, equation 1 is this. We see, you can see here as I said: let us put this as θ_1 and then let us put the other one as θ_2 and do a phase difference and let us estimate the path difference and all that. This paper actually has captured all of that.

(Refer Slide Time: 10:19)



And you will get a clearer picture as you look here. You see exactly equations 2 and 3 are that; you can see here he assumes one position, antenna position; gets you the same expression.

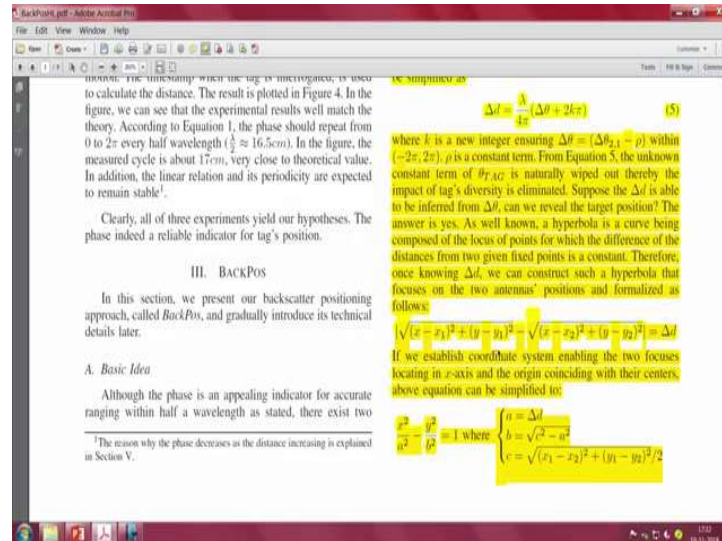
$$\theta_1 + 2k_1\pi = 2\pi \frac{2d_1}{\lambda} + \theta_{T_1} + \theta_{R_1} + \theta_{TAG}$$

$$\theta_2 + 2k_2\pi = 2\pi \frac{2d_2}{\lambda} + \theta_{T_2} + \theta_{R_2} + \theta_{TAG}$$

Now, you subtract 2 of them and you simplify you will get the expression.

$$\Delta d = \frac{\lambda}{4\pi} (\Delta\theta + 2k\pi)$$

(Refer Slide Time: 10:52)



In other words, a beautiful relation between the path difference and the phase difference; essentially that is what RFID localization is all about. If you are using hyperbola based positioning you find out; you do a phase difference and then you find out the path difference.

Now, suppose the Δd is able to be inferred from $\Delta\theta$, can we reveal the actual tag position? That's the beauty: the answer is yes. And as you know, you can apply a well known hyperbola. And this is nothing, but a curve that is composed of locus points for which the difference of the distances from 2 given fixed points is a constant.

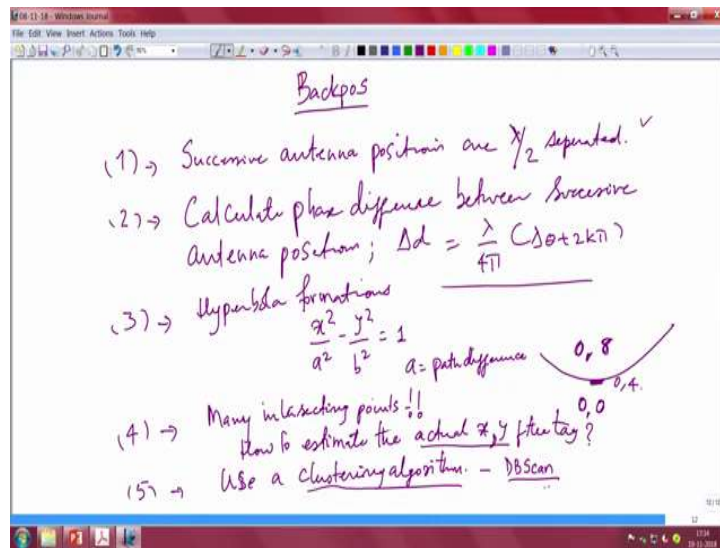
And once you know Δd , we can construct such a hyperbola that focuses on 2 antenna positions. And you can nicely formulize this, as this simple expression which is Δd . (x,y) are the tagged tags location. (x_1, y_1) is the first position of the antenna. (x_2, y_2) is the second position of the antenna; from there you will be able to estimate this delta d.

Now, the hyperbola equation is quite simple.

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \text{ where } \begin{cases} a = \Delta d \\ b = \sqrt{c^2 - a^2} \\ c = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} / 2 \end{cases}$$

c is the simple Euclidian method which this divide by 2 is essentially saying that the hyperbola is forming in between the 2 antenna positions and that is easy to infer.

(Refer Slide Time: 12:57)



Let me just pull out this thing here and concentrate on this picture. Supposing you did one reading from (0, 0) and the other reading you did from (0, 8). When I say 8, I mean 8 centimeters. That means, I moved 8 centimeters and made 2 readings, or I had 2 antenna one at (0, 0) and one at (0, 8) and I was throwing power at a tag which was located at (x, y). This is (x1, y1) and this is (x2, y2). The hyperbola is essentially formed in between the 2 positions. So, at (0, 4) you can actually draw a nice hyperbola.

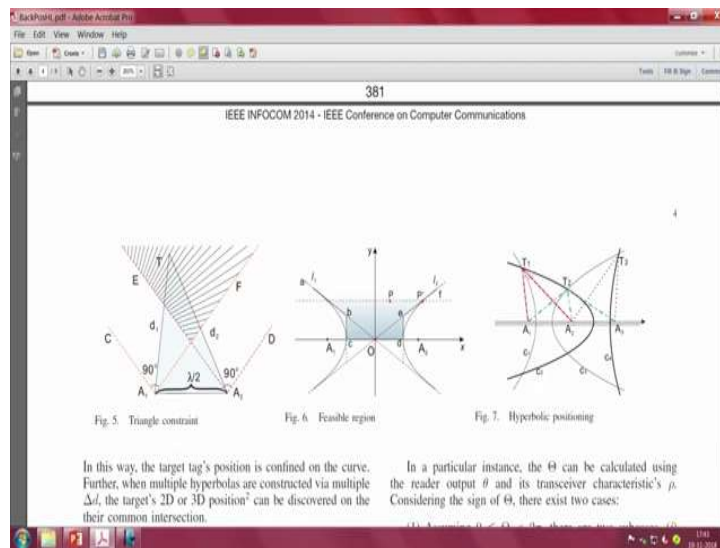
If you look at this little chart that I have here; it is actually capturing all that we discussed. First step, you need to have a successive antenna positions which are separated by $\lambda/2$. You calculate the phase difference between successive antenna positions. You arrive at

$$\Delta d = \frac{\lambda}{4\pi} (\Delta\theta + 2k\pi) ; \text{ then you have hyperbola formations. Then you will have}$$

multiple intersection points; the question is how to actually estimate the actual (x, y) of the tag; if you have so many intersecting points.

As I mentioned one simple way is to try and use a clustering algorithm and so one such clustering algorithm is indeed the DB-SCAN. DB-SCAN is a very popular clustering algorithm and we can go back and look at how this DB-SCAN works in a few moments. But before that let us see if there is any further discussion. So, this ‘divide by 2’ essentially is coming because of the fact that the hyperbola is formed in the middle between the 2 antenna separation points.

(Refer Slide Time: 15:01)



Just look at this picture which is sort of capturing several parts of the RFID hyperbola positioning. Let us assume that the reader's antennas can be positioned in A 1, A 2, A 3, A 4, A 5 and so on.

And the, so if you look at the hyperbola that formed between A 1 and A 2; you can see between A 1 and A 2; C 1 hyperbola and C 3 hyperbola are available. They are formed between A 1 and A 2.

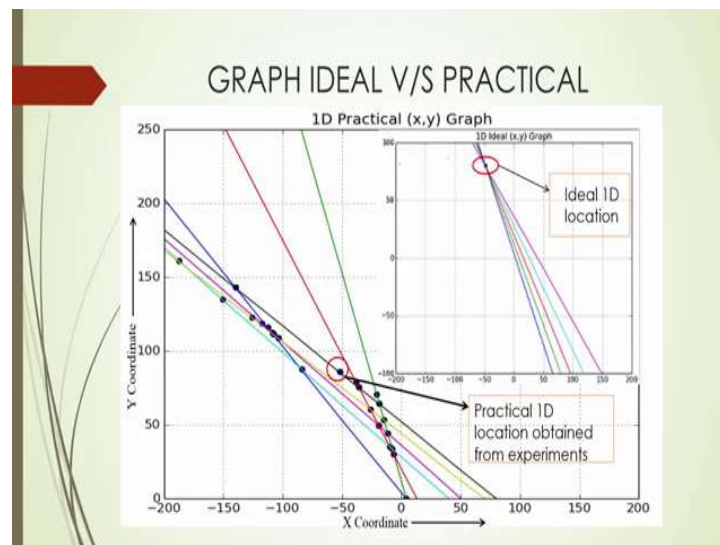
If you look at A 2 to A 3; you will see that C 2 and C 4 are being formed between the two of them. Now you can go on like this. So you go to the next A 4 position and so on and so forth, and then you realize that the tag itself can be anywhere in these positions: can either be in T 1, can be in T 2 or can be in T 3.

As a result, if you want to reduce this error in estimation of where it is; one simple way is you divide A 1, A 2 that the distance again into by half and have one more position. Let

us say A 1 prime (let us say something like A 1 prime here). And here you have A 2 prime and then again form 2 hyperbola there and then look at the intersection of these hyperbola; then come to know which is the actual position of the tag.

So, now we did this hyperbola formation between 2 antenna positions and intersection of several hyperbola happening. And then you are interested in knowing the actual (x, y) location of the tag. We did some measurement and I just wanted to show you, point you back to where exactly the problem is.

(Refer Slide Time: 17:29)



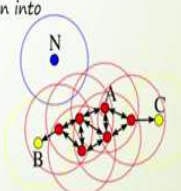
Look at this picture; this is some position; 1D position which I am interested; this has some (x, y) location in reality. And that would be easy if all these hyperbolas which are going are all intersecting; I just taken simplified line segment of a hyperbola. And I have just shown you where exactly the intersection is happening; that is happening at a single point only.

But that is not really the case; look at how these line segments from different hyperbola are actually intersecting. You will see that they are all at different points and you have to infer somewhere (x, y) that is available from these points. So, this is not going to be easy; you have to do some form of clustering and it is found that DB-SCAN is a good way to cluster them. It is basically density based SCAN.

(Refer Slide Time: 18:36)

DBSCAN CLUSTERING


- It stands for *Density Based Spatial Clustering of Application with Noise*.
- Groups together points with many nearby neighbours (high-density region), eliminating other points as noise.
- Used to form multiple customized clusters.
- Densest and closely packed cluster is taken into consideration.



And essentially it is a spatial clustering which is used from all the hyperbolas that are formed. So, you group together points with many from nearby region and eliminate all other points; which are considered as noise. And basically this is used to form multiple customized clusters. And straight forward densest and the closely packed cluster is taken into consideration. The point N is definitely not part of any of the other clusters. It appears to be an outlier; so actually you can eliminate them.

Similarly, look at the C and look at the B: they are part of the cluster. It is not that there is no intersection or anything, but they appear to be far away from any dense point. So, they can also be eliminated and the tag position indeed is within RED color: quite dense. So, this is the exact idea behind density based SCAN.

(Refer Slide Time: 19:56)



DBSCAN

- Given a set of points in a space, it groups together points that are closely packed together, marking as outlier points that lie alone in low-density regions.
- It takes in two parameters: Epsilon and MinPts.
 - Epsilon: It is the maximum radius of the neighbourhood of any point.
 - MinPts: Minimum number of points to be not considered as outlier.
- A point p is a core point if at least "MinPts" points are within distance ϵ of it (including p). Those points are said to be directly reachable from p .
- A point q is reachable from p if there is a path which consists only of core points, from p to q .
- The core points and reachable points form a cluster.
- All points not reachable from any other point are outliers.

Alright. How do you do this? Let us run through a few steps which will allow you to actually implement this DB-SCAN. First of all if you look at this picture there is a certain radius of this each circle here.

So, you need to define what should be your radius. If you have a large radius all the circles will vanish and become a very large circle. Slightly larger radius will also encompass N B A and C; so you do not want to do that. At the same time you do not want very small circles such that only one or two hyperbola are sitting inside them.

An optimal circle radius which will allow you to group all these hyperbola intersections.

So, this epsilon is indeed the radius. And number of points which are sitting inside them essentially will tell you whether a circle will form or not at all in the first place. Look at this epsilon: maximum radius of the neighborhood of any point sorry any point, and minimum points: minimum number of points to be not considered as outlier.

So, which means point N is indeed an outlier. So, you need some points, minimum number of points to be declared to be part of a nice cluster and only then you can say something about it. A point p is a core point if at least minimum points "MinPts" points (this is indeed like a variable). So, "MinPts" points are within the distance epsilon of it: these points are said to be directly reachable from this point p .

So, point q is reachable from p if there is a path which consists only of core points from p to q . Look at this picture again, these arrows are indicative of that the point q is reachable from point p ; if there is a path which consist only of core points from p to q . The core points and reachable points form a nice cluster.

So, I will go back and then I will show you; you can see that within this nice circle here; there are p and q which are within them and they form a nice beautiful cluster. This is one particular cluster. There can be another many overlapping cluster because as long as these definitions are adhered, you can actually call them clusters.

All points not reachable from any other point are essentially outliers; so point N becomes an outlier as you can see. In other words this is easy to implement. So, you could actually try. We will provide you some data files we have from the experiment that we are going to show you. And these traces you can use and you can use any programming tool of your choice. You can try MATLAB, you can try Python or whatever and try and find out the actual positions of the tag; this is like an exercise which you can implement and try.

So you will have to implement this algorithm and data files is something that we will make available to you.

So, welcome to RFID demonstration; you might have already seen that there are 3 ways by which you can do phase difference of arrival based localization. And specifically let us look at this spatial domain based phase difference of arrival based mechanism and let us just look at how we do that.

The big picture is very simple; the arrangement is of a departmental store and in a supermarket there are lot of items kept inside a rack and you are interested in localizing each one of these items.

(Refer Slide Time: 24:58)



So, the problem statement is very simple: there is a rack on the left side and then there is a rack on the right side.

(Refer Slide Time: 25:02)



The left side of the rack essentially has 2 items here which are both tagged; you can see that on extreme left there is a tag on top.

(Refer Slide Time: 25:12)



You can see the orientation is almost vertical.

(Refer Slide Time: 25:17)



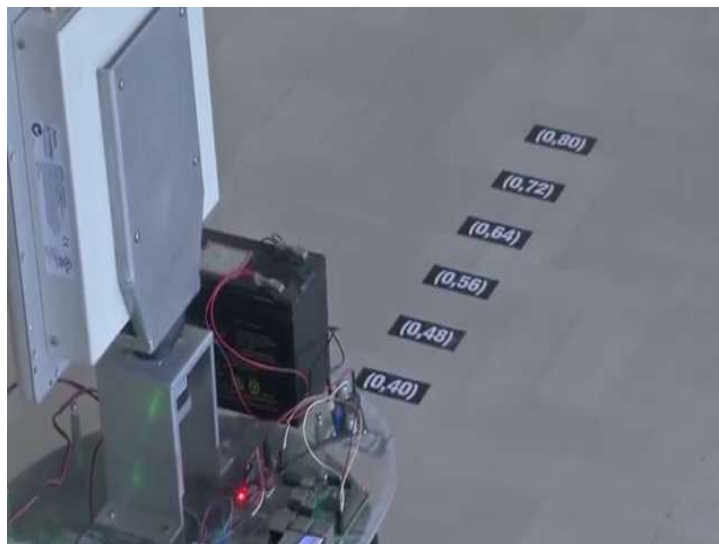
And then there is a number there (-120, 80); this indeed is the ground truth. Ok.

(Refer Slide Time: 25:24)



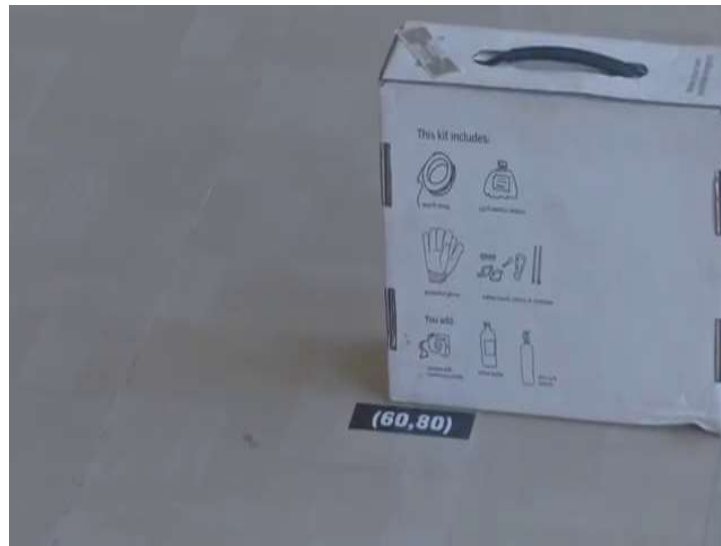
Then come towards centre left. Tag orientation is some 45 degrees; you can see a tag on top. And then this one's ground truth is $(-60, 80)$ and at exact middle is the corridor in which you are going from rack to rack.

(Refer Slide Time: 25:35)



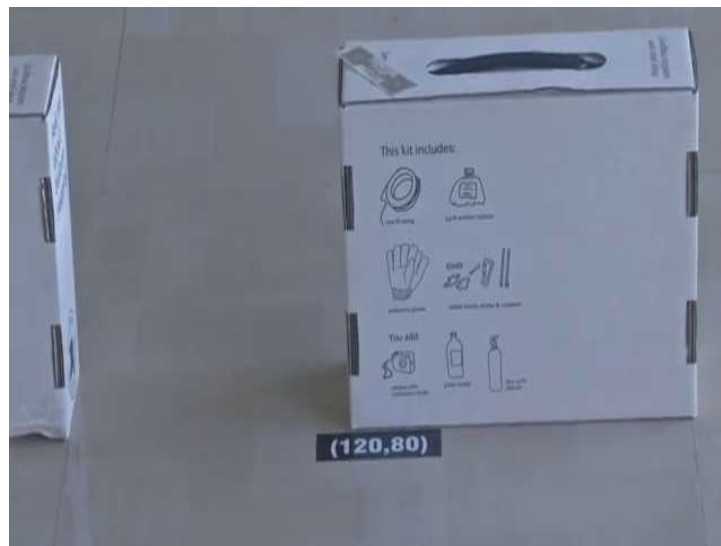
Here is another tag based on top.

(Refer Slide Time: 25:37)



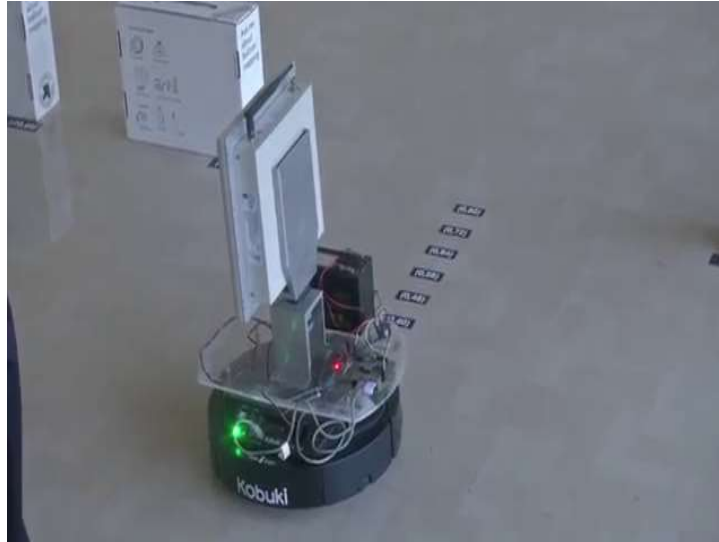
Its position ground truth is (60, 80) and the other one is (120, 80); All values are in units of centimeters.

(Refer Slide Time: 25:45)



So, the big picture is that you want to localize them; now how do you do that?

(Refer Slide Time: 26:05)



So, one idea that you can explore is: you have a single antenna system and supposing you make this antenna move spatially from one location to another location, you collect phase readings and then take the difference of successive phase values; you must be able to convert the phase difference to path difference and therefore, be able to localize based on the expressions that we saw in the class. So, let us see what exactly we plan to do here.

Thing is there is a reader here; this reader is from a company called Thing Magic and it has a beam width of 65 degrees. You can have antenna with different beam widths, and with different gain. This antenna appears to have a gain of about 8.5 dB and you can have higher gain antenna. So, this is the antenna that we have chosen here.

And it is throwing power at +18 dBm; through all these tags. If you put more power, if you throw more power; reflections from boundary sides will come; they may even power the right side tags and start reading from irrelevant location. So, you have to choose transmission power such that there are not too many reflections coming from all across the departmental store; and yet sufficient power so that the relevant tags are able to get read. Of course, there is a challenge of tag orientation; if the tag orientation is proper then it will read otherwise there can be issues there, but we keep that out of the mind.

In India, RFID frequencies are typically from 865 to 867 megahertz. There are 4 channels and there is no control on which channel the system is actually choosing; hence it exactly hops across these 4 channel. So, we will have to assume that some f (central value), it is

reading these tags. In fact, it could be reading one tag at one f and it could be reading at f_1 ; let us say. And it could be reading the other tag at f_2 , but all across this narrow 2 MHz range over which the RFID transmissions are permitted in this country.

The reader actually communicates to the tag using FM0 encoding and also there are additional settings which the reader can be set and which includes setting it to single target, session 1 and so on. So, all of this: the settings essentially are specific to RFID reader and those settings are all compiled into a nice beautiful Java code.

So, that is one part. Then there is also the requirement that the Kobuki, the robotic platform, has to move. All of these functionality essentially means that you need the nice Java code which will also take care of moving the robotic platform from one position to another spatial position so that you will be able to read the phases from both these positions.

So, the distance that it moves is less than $\lambda/2$; is actually 8 centimeter which is much less than $\lambda/2$; so that is the point. Essentially the script that is running on this computer which is a Raspberry Pi, has the ability to move the robotic platform by 8 centimeter. And also has the ability to tilt the reader to about 16 degree so that the tags are read successfully as much as possible and improve the probability of successful reading by tilting it to the certain degree.

So, this is one part of the experiment. After all of that the tags are read, the phase difference of arrival is considered, put into hyperbola expressions and ultimately you want to print the localization. The algorithm has to print some numbers, which in our hope is that should match this ground reality numbers. So, let us start the experiment and see how we will be able to start localizing.

So, now let us start and let us focus on movement of the antenna. So, a command is issued; the antenna the robotic platform is all set. Command is issued. The starting location of the robotic platform is at (0, 0). This is the reference position of the system and it starts to move from there. So, it makes one reading at (0, 0); goes to the next location (0,8), waits there for 10 seconds, makes one reading there and proceeds like this to about 11 positions before it stops at (0, 80), which is essentially the resting position for just one part of the rack.

Then you start with the second part by moving the robotic platform in reverse fashion. Ok. So, let us do this basic demo. So, you can see now that the reader antenna has shifted to angle which is roughly 16 degrees and then it is trying to read a set of tags.

(Refer Slide Time: 31:51)



And now it moves by 8 centimeter, very small movement, waits there for 10 seconds; makes a reading, hopefully reads these tags with a high success probability, gets the phase information, goes to the next reader location, reads the tags here in this location and then makes the next movement to another location and therefore, successfully gets phase readings one after the other.

Now essentially you should be able to start creating hyperbola; start forming hyperbolas. In this case a hyperbola is typically formed exactly in the middle between $(0,0)$ and $(0,8)$; at $(0,4)$ the hyperbola is formed.

And once these hyperbola starts intersecting; you realize that the intersection point indeed is the actual location of the tag. So, you have to solve the hyperbola equations and therefore and arrive at the decisions.

(Refer Slide Time: 33:09)

```
entered for loop..
start time:1.5247665805172
entered isweep..
write set to 1
write before!
Tag-ID : GEN2:000000000000000000000000A2 Phase : 14 time : 2307300 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 14 time : 2307430 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 11 time : 2300070 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 14 time : 2300110 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 5 time : 2300000 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 11 time : 2300010 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 23 time : 2309400 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 14 time : 2309130 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 19 time : 2310100 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 11 time : 2310230 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 10 time : 2310050 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 11 time : 2310930 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 14 time : 2311530 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 11 time : 2311640 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 11 time : 2312270 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 17 time : 2312350 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 14 time : 2312090 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 2 time : 2313070 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 22 time : 2313570 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 14 time : 2313790 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 19 time : 2314250 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 14 time : 2314490 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 10 time : 2314940 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 8 time : 2315221 RSSI : -47 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A1 Phase : 10 time : 2315010 RSSI : -50 distance : 63.699999999999996
Tag-ID : GEN2:000000000000000000000000A2 Phase : 11 time : 2315920 RSSI : -47 distance : 63.699999999999996
```

So, what at each antenna position exactly is being read is also important. So, you can see that you get Tag ID, the protocol which is EPC GEN 2. There is a Tag ID. It has to read the phase otherwise how will you know phase difference. So, it is reading the phase. It is reading a time at which the phase is the tag is being read. It also throws the RSSI which we said is not so useful for any localization purpose and this distance is essentially the antenna location from where the reading was made.

So, this is essentially what you get from a basically from a tag and somehow you have to construct the hyperbola equations using this phase information and ultimately arrive at the final location of the tag.

(Refer Slide Time: 34:01)

```
X array values [132.05104068859464, 112.30570963336058, 98.07192518368211, 49.70273296559511
402189, 37.603379906716995, 31.825245869245357, 36.061038165637505, 91.73114458185549, 78.00
, 60.830299774421565]
Y array values [133.80701116337062, 114.39722540669847, 100.40532494668184, 67.8247343929866
9267642, 71.92853662123179, 63.94919950548398, 69.79863367679474, 101.50035401606226, 90.498
81.6291335342429]
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Name A4
X array values [53.60422329833587, 74.0103397858526, 147.0081652430697, 83.54799855798863, 1
777456, 102.6644012691516, 139.3970068975936, 140.90212064049058, 95.90753268642788, 147.586
74.23012920349129, 88.93146244058445, 147.26227061038867]
Y array values [41.05164716804508, 55.15651021134507, 105.61316281168814, 61.74901255332687,
088, 67.6405779035434, 87.54838800912464, 88.36410794154601, 65.5581498746185, 90.1069356389
9869628899, 74.95148777668318, 90.00553944717899]
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Obtained xy locations of tags processed using DBSCAN clustering technique
0 A1 -120 80 -112.156271053 80
1 A2 -60 80 -68.0134332165 80
2 A3 60 80 61.8781580722 80
3 A4 120 80 106.757806104 80
root@raspberrypi:/opt/pl4j/examples#
```

After reading the phases from each antenna position; you will get multiple intersecting hyperbolas. And so you can see take this A4; this is essentially the right side of the rack; the last tag on the right side. And you will see many intersecting points. Now how do you make out anything from this where exactly the tag is.

So, what do you do? You essentially apply a clustering algorithm and we described that a good clustering algorithm indeed is the DB-SCAN algorithm. So, after applying DB-SCAN clustering algorithm, many of these intersecting points are clustered and you arrive at some location which is one (x,y) location. You can see the results here; A2 whose ground truth is (-60, 80); you ended up with estimated position of (-68, 80).

And A4 is (120, 80); we ended up with estimated position of (106, 80). So, it is fairly accurate. So we have actually shown that you could use spatial domain based phase difference of arrival and actually localize RFID tags.