Seeing Walls And Counting People

Peijie Li, Songbo Zhu, Zhizheng Qiao, Yinchao Shi
li_paige,zuhxs,qiaozhizheng,yinchao_shi@berkeley.edu

ABSTRACT

This paper is based on the project of an energy-efficient radar solution to seeing the walls and objects in the room, initializing the environment and starting to count people when the PIR sensor detects movement.

1 INTRODUCTION

Movement detection has become a popular topic in our daily life, such as counting people, monitoring activities of people or tracking moving objects surrounded the cars during driving. Among these, human movement is what we are always caring about. Our project is focusing on counting people in one specific area and knowing their activities.

The design is based on IWR1642Boost[1], which is a single chip radar sensor from TI. This IWR1642 chip is integrated with a complete and accurate radar processing chain. It is capable of locating people in the distance of 8-10 meters and 120-degree field of view. The chip itself has already provided demo of static interference cancellation and group tracking algorithm implementations. In our project, we have added some features to the basis of original demo. Two main phases are included. The first one, named initialization, is to reconstruct the physical environment, during which the radar sensor will differentiate the walls and other stationary objects in the room. After detecting the walls and objects in the environment, Arduino micro will receive a signal from PC, then start to receive the input of PIR sensor. PIR sensor used in the project is a long distance detection type from EKMB series devices[3], which can detect the target of human bodies at most 12 meters' distance with the speed of 1m/s. Only when the Arduino receives a high input from PIR sensor, which means there's someone moving in the room, it starts to count people and displays the number of people on the screen simultaneously.

The whole system has several key features. First of all, single wall or multiple walls are automatically detected and reconstructed in high accuracy and precision. At the same time, it recognizes some furniture or other static objects as simulating the scene in the area. It's an energy saving system which only turns to counting people mode when someone enters or leaves the room. For future practical use, we can assume a situation that there are several computers on tables which offer students to use in the lab, and from the scene we can know the location of tables so that we may know how many students are in the lab and whether students are using the computers or not.

2 HARDWARE CONNECTION

This embedded system has two main boards, several auxiliary parts and a display terminal on a PC. The working state of the system

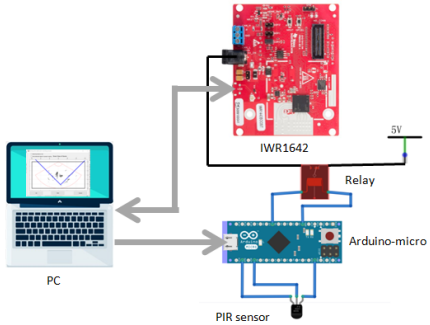

Figure 1: Hardware Connection Diagram

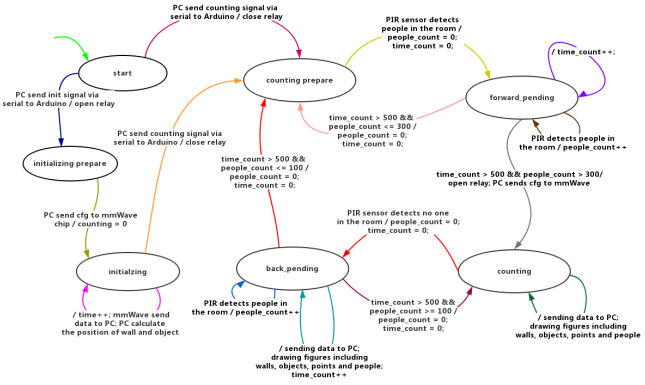

Figure 2: Finite State Machine of System

is controlled by an Arduino micro board. It can get the direction from the PC as well as analyze the signal from PIR sensor. If the current state is determined that the system needs the radar data, Arduino will power on the IWR1642 through a relay. The radar signal is collected and preliminary processed in the IWR1642. Then the processed data will be sent to the PC for further analysis.

3 STATE MACHINE ANALYSIS

The finite state machine is shown in figure 2, it can also be seen as a hierarchical state machine with two parent states: wall detection (aka Initialization state) state and people counting state.

For the wall detecting state, the initial state is the "start" where the system will wait for any command from host PC. If host PC sends the "initiate" signal via serial port, the system will proceed to "initializing prepare" state which will open relay to power on the IWR1642. Host PC will send configuration to the IWR1642 board and set the measurement parameters. Then the system will proceed

EECS 149/249A, December 14, 2018, UC Berkeley
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Figure 3: Sample Working Environment for Radar Sensor

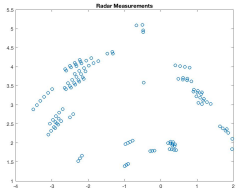

Figure 4: Radar output of measuring the physical setting shown in Figure 3.

to the "initializing" state where the board transmits the measurement data collected by radar sensor to host PC. Host PC will then apply different algorithms to analyze the received data and display the results in visual diagrams. After the system finishes reconstructing its physical surroundings, it goes to the people counting state upon receiving a signal from host PC.

For the people counting state, the initial state is "counting prepare" which is designed to wait for RIP sensor signals before proceeding to next state named "forward pending". It is necessary because the PIR sensor could produce unreliable signals due to the unpredictable and constantly changing environment. The "forward pending" state will accumulate the number of PIR signals received within a certain amount of time and proceed to the actual "counting" state only if the PIR sends enough positive detection. Otherwise it will go back to the "counting prepare" state. The prepare and pending states together help the system to avoid powering on the IWR1642 board too frequently and reduce energy cost. Once system received sufficient number of signals, Arduino will use the relay to turn on the IWR1642 board and host PC will send configuration to the board for people counting. The board will continuously send data back to host PC, which will display figures includes walls, objects, people and their tracks. During the "counting" state, if the PIR sensor detects there isn't any movement in the room for a time period, the system will go to the "back pending" state, Arduino will power off the radar board and the overall system will wait for people come in.

4 WALL DETECTION AND OBJECT RECOGNITION

Texas Instruments IWR1642 mmWave Sensor measures its physical surrounding and produces a set of measurements where each is a 2-dimensional real vector representing a location relative to the sensor. The goal of wall detection and object recognition is to reconstruct the shape and location of walls and possibly large objects such as desks and chairs from the observed measurements. For the rest of this section, we will use the physical settings shown in Figure 3 as example.

4.1 Noise Reduction and Data Clustering

Unsurprisingly, the radar measurements contain a nontrivial amount of noise. Figure 4 demonstrates one sample set of radar measurements obtained by IWR1642 from measuring the room in Figure 3. It's an imperative task to first remove noise from the measurements in order to facilitate wall detection and object recognition.

Consider a set of points to be clustered. The main purpose of noise filtering and data clustering is to identify cluster of points that are closely packed together and points whose nearest neighbors are too far away. Points that lie alone in low-density regions are more likely to due to sensor noise and should be filtered out.

It is particularly challenging to filter out noise and identify valid clusters from a set of radar measurements, due to the fact that radar signals could reflect multiple times and produce noise measurements that looks like a real object. Previously there have been many algorithms designed to solve data clustering problem, including but not limited to k-means clustering, density-based spatial clustering of applications with noise (DBSCAN). We closely examine the applicability and effectiveness of both algorithms.

K-Means Clustering: Given a set of measurements (x1, x2, ..., xn), where each measurement is a 2-dimensional real vector, k-means clustering aims to partition the n measurements into k cluster sets S = S1, S2, ..., Sk so as to minimize the within-cluster sum of squares[4]. The within-cluster sum of squares can be obtained using MatLab's built in function kmeans with pre-set number of clusters. Hence we simply run kmeans with k = 1, 2, ... until the results converge.

DBSCAN: Consider a set of measurement points, DBSCAN classifies them into core points where at least minPts points are within distance ϵ of it, density-reachable points that locates within ϵ distance neighborhood of some core points, and outliers that are not reachable from any other point. Two inter-reachable core points, along with their density-reachable neighbors, became part of the same cluster[2]. The pseudo-code for DBSCAN can be found in figure 5.

We ran a few experimental tests on the radar measurements on the meeting room shown in figure 3 in order to compare the effectiveness of K-means clustering and DBSCAN. Results are shown in Figure 6.

Compared with k-means clustering, DBSCAN does a better job at filtering noise. K-means clustering is more likely to produce false positive errors, where some noise data points are evaluated to a valid cluster. This is likely due to the large size of radar measurements. In contrast, DBSCAN offers more flexibility in allowing the users to define custom ϵ and MinPts so that the algorithm is able to detect clustered outliers as noise and filter them out. Overall DBSCAN is able to find arbitrarily sized and arbitrarily shaped clusters quite well.

We extensively tried a wide range of combinations of ϵ and MinPts values. figure 7 shows the results for varying ϵ and MinPts.

```
DBSCAN(measurements, epsilon, minPts) {
   ClusterCount = 0
   for each measurement m in measurements {
      /* m is already processed */
      if getClusterID(m) != NO_CLUSTER
         continue
      /* get epsilon-distance neighbors */
      neighbors = getNeighbors(measurements, m, epsilon)

      /* If m locates in a low-density region */
      if |neighbors| < minPts {
         m.UpdateCluster(NOISE_CLUSTER)
         continue
      }

      /* Found a new cluster */
      ClusterCount = ClusterCount + 1
      m.UpdateCluster(ClusterCount)

      /* Include m's neighbors in the cluster */
      for each measurement M in neighbors {
         /* Add "noise" neighbor to cluster */
         if getCluster(M) == NOISE_CLUSTER
            M.UpdateCluster(ClusterCount)

         /* If M is not precessed yet */
         if M.getCluster() == NO_CLUSTER {
            M.UpdateCluster(ClusterCount)

            /* Look for second-degree reachable points */
            newNeighbors = getNeighbors(
                              measurements, M, epsilon)
            if |newNeighbors| ≥ minPts
               neighbors = union(neighbors, newNeighbors)
         }
      }
   }
}


getNeighbors(measurements, curr, epsilon) {
   neighbors = []

   /* Iterate through all neighbors of m */
   for each measurement m in measurements {
      if distance(curr, m) ≤ epsilon then {
         neighbors.add(m)
      }
   }
   return neighbors
}
```

Figure 5: Pseudocode for DBSCAN algorithm

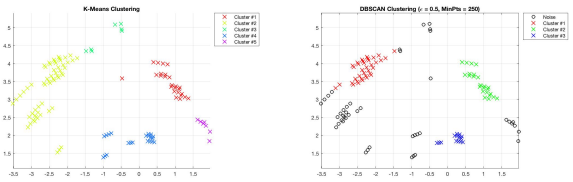

Figure 6: Performance of K-means Clustering and DBSCAN.

With ϵ = 0.5 and MinPts = 150, DBSCAN is surprisingly accurate in identifying data clusters as walls and desks and filtering out noises. We use these values in our project.

4.2 Wall Detection and Object Recognition

After noise measurements is filtered out, we are left over with clustered measurements where each cluster represents either a wall or an object such as desks or chairs. The next task is to differentiate walls from other static objects.

Two scenarios of potential wall locations are considered in this project: when the sensor is placed facing a single wall and when the sensor is placed at one corner of a rectangular room. Both cases are visualized in figure 8 where the green dash line represents the x-axis and y-axis of visual diagram produced by the sensor.

Notice that in the first case, the sensor should detect the wall that's horizontal to the x-axis. The representation of the wall is modeled by a line y = b where b is a constant real number. In the second scenario, the sensor is expected to detect two walls, both of which are linear lines that intersect the x-axis (or y-axis) forming a 45 degrees' angle. The line representations of the walls are of model y = x + b or y = -x + b where b is some real number.

The wall detection task is then reduced to the problem of looking for data clusters whose best fit line is of specified model. Namely, for each data cluster we construct a best fitting line through the data points. If the coefficient (slope) of the best fitting line is close enough to models we've discussed, the conclude that this data cluster is likely representing a wall. We then reconstruct a best fitting line with slope (0, 1, or -1) through the data points in the cluster and use it to represent the wall.

Our algorithm, however, could give false positive errors where objects are evaluated to be walls because the data cluster produces a line model similar to the models of walls. For example, given radar measurements shown in figure 9 we could likely get two line models (from both the red- and pink-colored clusters) for walls but the pink cluster is actually an object. A simple approach to correct this false positive error is to always use the cluster that's farthest away from the sensor. In other words, we always treat line models with largest y-intercept as walls and all other clusters as objects because walls are always the farthest object detected by mmWave sensors.

A cluster where the best fitting line differs too much from the models of walls is classified as object. Objects are demonstrated using a black circle, with center being the centroid of the cluster and radius being the within-cluster average of point-to-centroid distances. Figure 10 shows the result after applying wall detection and object recognition.

4.3 Measuring Distance

When the sensor is placed facing a single wall, it is possible to calculate the distance between the sensor and the wall after we obtain the line representation of the wall. The distance is simply the y-intercept of the line representation. figure 11 shows a demo of measuring distance from the sensor to the wall it is facing.

5 COUNTING PEOPLE

After radar signals have been received and the room environment is successfully reconstructed, we now proceed to the next state, named counting state, to count the number of people. At this state our project will only count moving objects in the region, ignoring all stationary walls and objects. The system will report the number

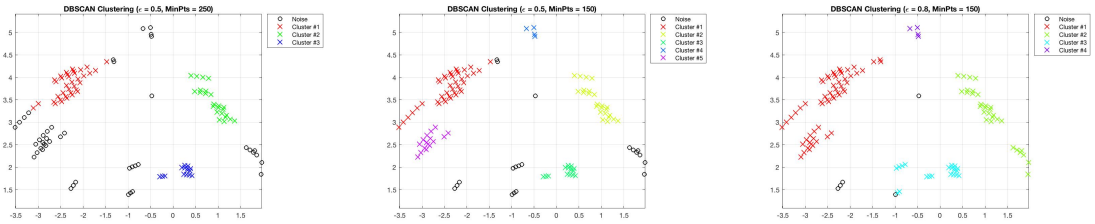

Figure 7: Noise Filtering and Data Clustering with DBSCAN with varying epsilon and MinPts.

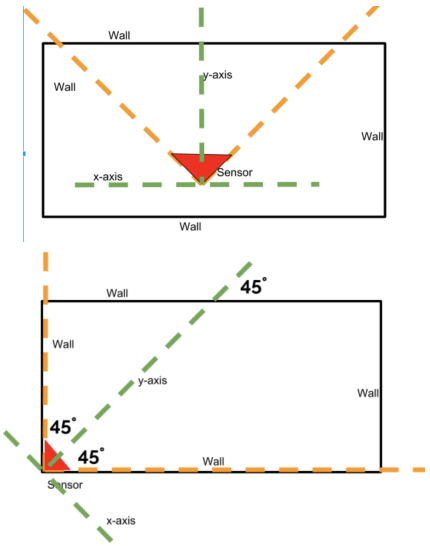

Figure 8: Two Wall-Sensor Scenarios

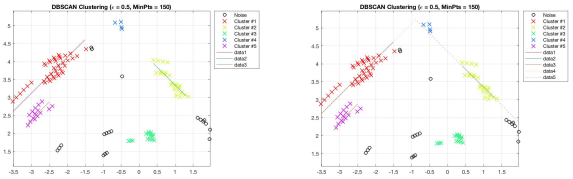

Figure 9: False Positive Errors in Wall Detection Before/After applying fix.

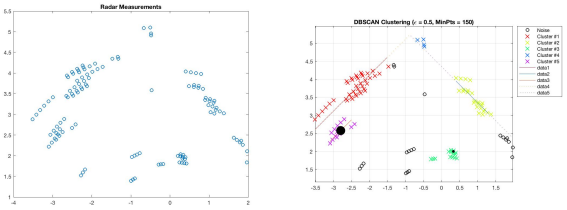

Figure 10: Object Recognition in the presence of walls.

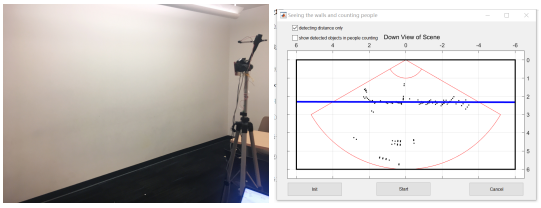

Figure 11: Measuring Distance between the sensor and the wall

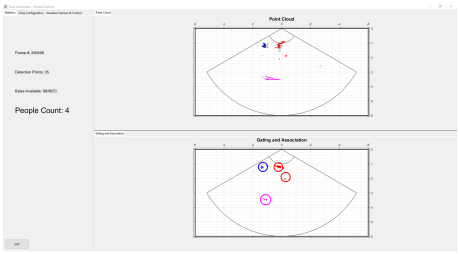

Figure 12: Actual working state of terminal

of moving people at real time as it receives radar measurement and analyzes the gathered information. In addition, the position and velocity of each moving person are also reported and displayed at real time.

It is possible for the radar sensor to detect some movement behind the wall, possibly due to the reflections of radar signals. Using the linear models of walls, our system will automatically ignore any moving people behind the walls and ensure a correct count of people in the room.

TI provides demo code that counts the number of moving people. We added a few modifications and improvements such as filtering moving targets outside a given range, etc. Figure 12 shows that the system detects four moving people.

6 EVALUATION

This project is a great success. It is made up of two phases: the initiation (aka setup) phase and the counting phase. In the initiation phase the project obtains radar measurements and reconstructs walls and objects. In the counting phrase the program outputs and updates the headcount of people in real time.

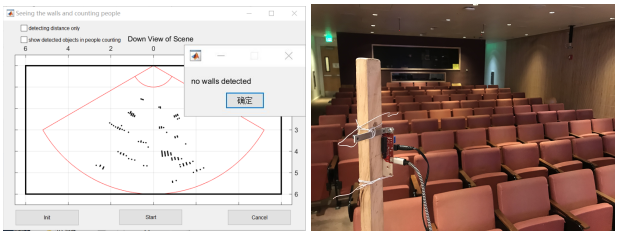

Figure 13: No walls are present; chairs line up in rows

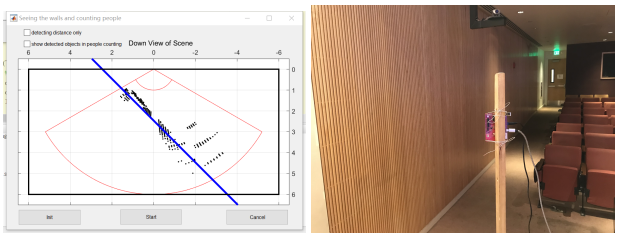

Figure 14: This simulates the scenario where only one wall is present.

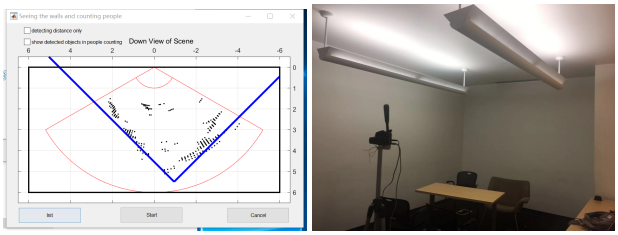

Figure 15: Detecting two perpendicular walls

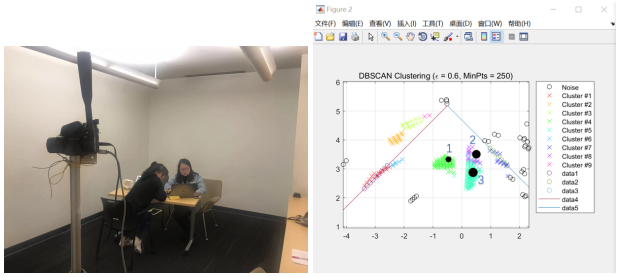

Figure 16: Evaluation of Object Recognition

6.1 Evaluation of Initiation Phase

The initiation phase takes up two 1 minute for the chip to measure its physical surroundings and reconstruct walls and objects, if present, from radar measurements. We extensively tested our project at a variety of places including study rooms, classrooms in Soda and Cory Hall as well as the large HP auditorium. Results have shown that our projects exhibit accuracy and precision in reconstructing its physical surroundings.

6.1.1 Accuracy in Wall Detection.

We tested our project in a few different environments to see if it can accurately detect the presence of walls and objects and reconstruct them correctly. Namely, we'd like our project to perform correctly in an environment where there is no wall, a single wall and two walls. It should also be able to differentiate static objects such as desks and chairs from walls.

We did our evaluation tests in Soda HP auditorium. Figure 13 shows the performance of our project when there is no wall present in the surrounding area but chairs lined up in rows. Although from radar measurements the chairs look very alike a wall, our algorithm was able to correctly differentiate the chairs and reported an absence of walls.

We then test our project to see if it can accurately detect a single wall in the presence of noise. Figure 14 shows that our project works perfectly in reconstruction one single wall from radar measurements. Again, the lined up chairs are not considered as walls. In this example they are marked as noise due to insufficient measurement data points.

Lastly, we tested our project in a much smaller study room to evaluate its ability of reconstructing two walls. As shown in Figure 15, our algorithm is effective in filtering out noise and reconstructing two perpendicular walls.

Our wall detection algorithm works really nice even with multiple reproduction. When detecting a same room for several times, results are almost always the same only with subtle bias, the number and the direction of the walls are always correct.

6.1.2 Accuracy in Object Recognition.

Another evaluation metric is if objects can be recognized accurately. It is challenging for our radar sensor to collect measurements of objects that are made by/covered by certain materials. For example, our radar sensor is not able to detect the presence of a wood/plastic chair unless part of it, such as chair legs, is made of metal. The limitation of radar sensor hurts the accuracy of our performance of object recognition.

Although we could find enough metal objects, for testing purpose we have a person sitting still so that we'd expect our project will recognize that person as a stationary object. As seen in 16, our project successfully reports two stationary persons and also recognizes those metal desk legs as the third object. For some objects like iron trash can, the object detection works really well.

6.1.3 Precision.

The precision of wall detection and reconstruction is measured by the calculating the difference between the physical and measured distance from the sensor to the wall.

Distance, or perpendicular distance, from the sensor to the wall is measured using the coefficients of the line that models the wall. In Figure 11 we were testing the accuracy of wall detection in one of the Soda study rooms. We placed our sensor 2.2 meters away from the wall. Our project was able to detect the wall and report a distance ranging from [2.19 - 2.24] meters (Figure 17) in several measuring attempts. The deviation is less than 0.04 meters, and the real distance calculated by a tape measure is 2.22m.

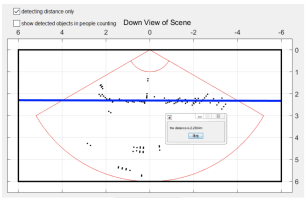

Figure 17: Measured Distance to the Wall placed 2.2 meter away

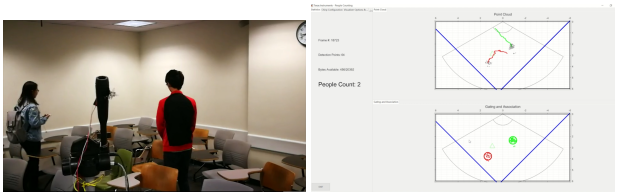

Figure 18: Evaluation of People counting

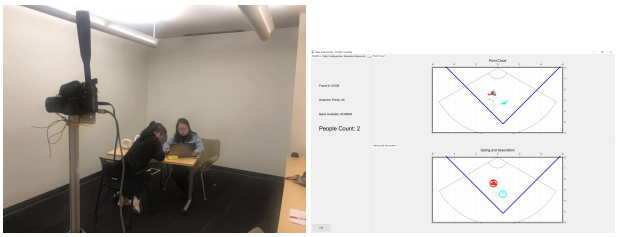

Figure 19: Evaluation of People counting

6.2 Evaluation of people counting

For this part, we used the demo code from TI, the counting is processed on the mmWave Radar chip. The chip uses an aggregation algorithm to tell the number of people and the position of each one. It will also return the speed and the acceleration of every individual so that we can draw the moving trace on our GUI. As in the following figure, the number and the position of people are accurate.

For Figure 18 and Figure 19, the blue line are the walls we detected. In the right picture, the upper figure represents the moving trace, another figure shows the position of each person. For instance, in Figure 18, people are moving, so the trace are long. Similarly, the trace in Figure 19 are short since people are siting still in the chair.

6.3 Integrated work

link for demo video: Integrated working demo [1]

7 DISCUSSION AND FUTURE WORK

Our project does a great job in analyzing received radar measurements as well as reconstructing the physical environment. However, there are still limitations due to the radar sensor we are using. We also see a potential improvement in how we could better represent the results to the future users of our project.

[1]https://www.youtube.com/watch?v=LYKIYf1EUGo

The IWR1642Boost chip by default filters out static interference and hence makes it difficult for us to obtain radar measurements of walls and objects in initiation phase. In order to obtain these radar measurements, we have to shake the radar sensor slightly in order to add a little vibration to it and get measurements of stationary walls and objects. In the future we plan on connecting the sensor to a motor that vibrates automatically during the initialization phase.

In addition, certain materials are difficult for radar sensors to detect. For example, chairs and desks made of wood and plastics absorb radar signals and they could be invisible from radar sensors. In addition, radar signals could go though materials like glass. As a result, radar sensors fail to detect the presence of glass wall and instead report objects it.

Furthermore, we plan to upgrade our system to transmit radar measurements from sensor to host PC via internet or Bluetooth. This will help us to better modulate different components of the project. We are also going to improve the representation of the information received from radar to make it more straightforward and understandable to future users.

As we mentioned early in the paper, this system can be a useful tool to monitor human activities within a restricted area. To illustrate an example, it can monitor activities in a lab. The feature of object recognition can be used for monitor use of certain sensitive resources, such as laptops or cabinet safes, etc. The use of PIR makes the entire project energy-efficient so that the system could be ready-to-work at all times.

REFERENCES

[1] TEXAS INSTRUMENT (Ed.). [n. d.]. IWR1642 EVM (IWR1642BOOST) Single-Chip mmWave Sensing Solution User's Guide (Rev. B). ([n. d.]). http://www.ti.com/tool/IWR1642BOOST

[2] JÃűrg Sander Martin Ester, Hans-Peter Kriegel and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *Second International Conference on Knowledge Discovery and Data Mining* (1996), 226–231.

[3] Panasonic (Ed.). [n. d.]. EKMB standard profile(1,2,6uA). ([n. d.]). https://na.industrial.panasonic.com/products/sensors/sensors-automotive-industrial-applications/pir-motion-sensor-papirs/series/ekmb-wl-series/2480

[4] N. S. Netanyahu C. D. Piatko R. Silverman T. Kanungo, D. M. Mount and A. Y. Wu. 2002. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 7 (July 2002), 881–892. https://doi.org/10.1109/TPAMI.2002.1017616