

AN AUGMENTED REALITY FACET MAPPING TECHNIQUE FOR RAY  
TRACING APPLICATIONS

by

Varun Kumar Siddaraju, B.E.

A thesis submitted to the Graduate Council of  
Texas State University in partial fulfillment  
of the requirements for the degree of  
Master of Science  
with a Major in Engineering  
December 2018

Committee Members:

George Koutitas, Chair

Semih Aslan

Damian Valles

**COPYRIGHT**

by

Varun Kumar Siddaraju

2018

## **FAIR USE AND AUTHOR'S PERMISSION STATEMENT**

### **Fair Use**

This work is protected by the Copyright Laws of the United States (Public Law 94-553, section 107). Consistent with fair use as defined in the Copyright Laws, brief quotations from this material are allowed with proper acknowledgement. Use of this material for financial gain without the author's express written permission is not allowed.

### **Duplication Permission**

As the copyright holder of this work I, Varun Kumar Siddaraju, authorize duplication of this work, in whole or in part, for educational or scholarly purposes only.

## **DEDICATION**

To my family, friends and my inspiration

“ Dr. B. R. Ambedkar ”



## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks to my advisor Dr. George Koutitas for giving me the opportunity to work with him, and for his guidance, support, and encouragement during my entire MS program. I am grateful to him for his trust, patience, and constructive criticisms, which helped me improve the quality of my research. Dr. Koutitas not only trained me to be a good student, but he has also taught me in many other ways that could lead me to success in my future career.

I wish to express my gratitude to all the faculty members of the Ingram School of Engineering at Texas State University for their excellent advice, constructive criticism, and helpful and critical reviews throughout my MS program. A special thank, goes to Dr. Vishu Viswanathan, Dr. Semih Aslan and Dr. Damian Valles, who kindly agreed to serve in my thesis research. Their valuable comments and enlightening suggestions have helped me to achieve a stable research path towards this thesis.

I would also like to thank all my friends Jeschel, Kiran, Vidya, Abhishek, Manju, Karan and Meghana for their continuous motivation, support and guidance. Finally, I would like to express my sincere gratitude to my parents and family, for their patience, continuous support and encouragement throughout this thesis.

## TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS .....	v
LIST OF TABLES .....	viii
LIST OF FIGURES .....	ix
LIST OF ABBREVIATIONS .....	xi
ABSTRACT .....	xii
CHAPTER	
I. INTRODUCTION .....	1
Background .....	1
Augmented reality .....	3
Indoor environment mapping .....	4
Research objectives and solutions .....	4
Thesis outline .....	5
II. LITERATURE REVIEW .....	6
III. LOW COST 2D IMAGE TO FACET MODEL .....	13
Overview .....	13
Image pre-processing .....	15
Edge and corner detection .....	16
Computation of wall dimension .....	19
Door detection .....	21
IV. SPATIAL MAPPING .....	22
Minimum-Maximum algorithm .....	23
Overview .....	23

Spatial mapping process .....	24
Spatial map data .....	25
Individual indoor wall vertices clustering.....	26
Computation of wall dimension.....	27
Spatial understanding algorithm .....	30
Overview .....	30
Topology result .....	31
Individual wall identification.....	31
Computation of wall dimension.....	33
V. THE FACET MODEL .....	35
Constructing the 3D environment .....	35
Data structure .....	36
VI. RESULTS .....	38
Graphical user interface .....	38
Augmented reality to ray tracing .....	39
Wall dimension measurement comparison .....	43
Indoor environment signal strength at different locations .....	48
VII. CONCLUSION .....	50
VIII. FUTURE WORK.....	53
APPENDIX SECTION.....	55
REFERENCES .....	99

## LIST OF TABLES

Table	Page
1. Data structure of the facet model .....	37
2. Wall dimension measurement comparison between proposed algorithms .....	43
3. Proposed algorithms individual accuracy and difference .....	45

## LIST OF FIGURES

Figure	Page
1. Indoor environment mapping using SLAM camera .....	7
2. Different type of edge detection algorithms and their comparison.....	8
3. Corner detection algorithm conditions.....	9
4. Microsoft HoloLens hardware details.....	10
5. Flowchart of 3D indoor facet mapping .....	13
6. Region of interest, pre-processed image, edge and corner detection.....	15
7. Flowchart of edge detection and corner detection .....	16
8. Corner points matching between two ROI of a wall image.....	20
9. Detected wall boundary .....	20
10. Door detection ROI, corner detection and door boundary.....	21
11. Flowchart of Minimum-Maximum algorithm .....	23
12. Spatial map of an indoor environment.....	24
13. Minimum-Maximum vertices identification.....	27
14. Spatial map of individual walls based on vertices normal.....	29
15. Spatial understanding algorithm .....	30
16. Individual wall identification using spatial understanding .....	32
17. 3D vector map and 3D indoor environment interior view .....	35
18. Individual room integration into a building .....	36
19. Graphical user interface .....	38

20. 3D vector map of a building, 3D interior view of a building and implementation of ray tracing algorithm on the facet model .....	40
21. Wall dimension measurement comparison chart .....	45
22. Indoor vector map and ray tracing result comparison .....	48
23. Signal strength of all three algorithms in different locations.....	49

## LIST OF ABBREVIATIONS

Abbreviation	Description
AR	Augmented reality
SLAM	Simultaneous localization and monocular
RGB-D	RGB Depth
3D	3 <sup>rd</sup> Dimensional
5G	5 <sup>th</sup> Generation
GUI	Graphical user interface
EM	Edge map
TL	Top left
TR	Top right
GO	Geometric Optic

## **ABSTRACT**

This research presents a novel spatial mapping technique that is capable of extracting the vector map of an indoor environment based on the images captured from a smartphone camera and the spatial maps captured from the Microsoft HoloLens. The extracted vector map follows the facet model concept and can be used as input in ray tracing algorithm. The ray tracing algorithm is used for visualizing and predicting the indoor wireless channels. The proposed solution offers three different algorithms, the first algorithm (Low cost 2D image to facet model algorithm) uses the edge and corner detection algorithms to compute the coordinates of the walls and doors of the indoor environment. The second algorithm (Minimum- maximum algorithm) computes the spatial map corner vertices by using the data processing techniques. The third algorithm (Spatial understanding algorithm) uses the Microsoft HoloLens Toolkit's "spatial understanding" feature to compute the spatial maps for detecting and measuring the individual wall dimensions. Finally, using the corner coordinates, spatial corner vertices and individual wall dimensions from all the three algorithms, a simple 3D vector map is designed. The output of all the algorithms is a facet model that can be used by ray tracing algorithms which are embedded in Augmented Reality (AR) applications. The overall process provides a better human-to-network interface and an improved user experience that is expected to provide a new way for indoor network planning of residential 5G systems.



# I. INTRODUCTION

## Background

Augmented reality (AR) is an enhanced version of a real-time reality where the computer-generated virtual images are augmented over a real-world environment. For augmenting the virtual data over the real-world environment, the system should understand the physical world environment. The indoor environment is one of the physical worlds where we spend most of our time. Computing indoor environment vector map became an active research field that encompasses the use of the real-world indoor environment in different AR applications like gaming, indoor navigation, interior designing, property advertising, indoor security, and other applications related to the Internet of Things (IoT), and ray-tracing algorithms, etc.

Commonly used indoor mapping methods are 2-dimensional (2D) mapping, where all the dimensions of a building are measured manually, which is not suitable for real-time AR applications. Research is going on to meet the need of a real-time 3D indoor vector map for various AR applications using the high-end camera like RGB depth (RGB-D), simultaneous localization, and monocular (SLAM) camera. Currently, AR glasses like Microsoft HoloLens have an integrated spatial mapping technique. Where it uses different hardware sensors, and depth camera to map the surrounding environment.

However, these cameras are complicated, and hard to use. To overcome these problems, a new solution is proposed where the individual wall images are captured and analyzed for measuring building dimensions. Different image processing techniques are used for building dimension measurements. The proposed solution is simple, and easy to

operate without any requirement of an external devices like depth, RGB-D or SLAM camera.

The proposed solution uses the individual walls of an indoor environment captured by the smart-phone camera as input. These images are enhanced by using different image processing techniques. The image noise present in the captured images are reduced using the Gaussian filter. Wall boundary of all individual walls is extracted using the edge detection algorithms. Corners of all individual walls, and their doors are detected using corner detection algorithms. The dimensions of all walls, and doors are calculated by measuring the distance between the two corners. Based on the calculated dimensions, and Cartesian coordinate system, coordinates (x, y, z) of each corner are computed. Using the calculated coordinates, the 3D indoor vector map of an indoor environment is built. After successful computation of indoor vector map, the Individual wall images were embedded over their respective walls of a 3D indoor vector map to form a 3D interior view.

Thus, the vector map developed from the proposed solution adds a new capability for mobile AR applications by making it to learn the indoor environment details like door position, room width, and breadth. This enables the augmented reality user to interact with the computed vector map with more precise details. A 3D hologram of the computed vector map can also be used in fully immersive virtual reality applications. The developed vector maps can also be used in determining the indoor environment wave propagation through the ray-tracing algorithm. The proposed solution works best for all medium-sized apartment, but it is hard to capture the complete large wall, so if we need to compute a vector map of large space area, the Microsoft HoloLens is used. The spatial

map computed from Microsoft HoloLens are sophisticated, and includes the coordinates details of all wall spaces, furniture, and other things present inside the environment.

The second proposed solution reduces their complexity of indoor vector mapping by converting the complex spatial maps into simple vector maps with only the coordinates of the room. These simple vector maps can also be used in other technologies. In this research, we are using these vector maps for indoor network planning. Thus, for small apartments with considerable wall spaces, the indoor network mapping using smart-phone can be used. The smartphone indoor vector map techniques enhance the mobile AR applications capabilities. The second proposed solution works best for large commercial spaces, where the dimension of all wall spaces is measured, and indoor network maps are computed. These indoor vector maps can be used in ray tracing algorithms to compute indoor network planning by visualizing the signal strengths of various channels.

### **Augmented reality**

Augmented reality (AR) is a combined view of both physical, and virtual world, where the graphical data are augmented on top of the real physical world. The user can see both virtual, and real world together on top of each other, which enables a new dimension of reality which will be helpful in educational, industrial, training, entertainment, and advertisement industries. For augmenting the virtual data over the real-world environment, the system should understand the physical world. For that, the indoor environment is one of the physical worlds which can be used to augment the

virtual world. Thus, our indoor environments can be mapped and used for overlaying the graphical data. So that, we can experience the virtual or digital content like computer display, games, videos, etc. by sitting in the living area without blocking our physical world.

### **Indoor environment map**

The spatial maps are the vital part of the AR, where the graphical data are augmented over the real physical world. Since most of the time we spend our time inside our house, office, school, and factory areas, etc. We will be using the AR in indoor environments compared to outdoor environment. Since we need to augment the graphical data in our indoor environment, we should efficiently map the indoor environment so that the AR applications understand the physical world and augment the virtual world. Therefore, the need for efficient indoor network mapping is very high for the AR.

### **Research objective, and solutions**

The need for indoor environment maps in the AR technology is discussed above. Currently, the AR is introduced into the market from major IT companies like Google, and Microsoft. The Microsoft took the AR into a next level by integrating the spatial mapping technique into their well sophisticated AR glass called HoloLens. At present only, few AR glasses are released into the market. All AR glasses use the same spatial mapping technique which uses sophisticated hardware sensors, and depth cameras to map the environment. These AR glasses map the complete spatial map of the indoor environment with all the furniture, and things present inside it. These spatial maps are useful in running complex AR applications. However, in certain simple applications, only

the vector map with the indoor environment coordinates are enough. These simple coordinate vector maps can be built using only the basic camera, and no sophisticated hardware sensors, and depth cameras are needed.

This feature reduces the need for expensive AR glasses for mapping the indoor environment. A simple indoor environment vector maps can also be computed using all smartphones inbuilt cameras. This feature adds on a new simple spatial mapping with indoor coordinates to mobile AR applications. Thus, we are extending the mobile AR capabilities, and simplifying the existing AR glasses spatial maps into simple coordinate systems for simple AR applications. In this research, we have designed an indoor environment vector map technique for smartphones, and 2 different algorithms to simplify the Microsoft HoloLens spatial maps into simple vector maps to implement indoor network planning through ray tracing algorithm.

### **Thesis outline**

This thesis is organized as follows. In Chapter II, a detailed literature review based on the previous work related to spatial mapping, corner detection, indoor environment vector map, and indoor network planning is explained. In Chapter III, computing indoor environment vector map from a basic camera is explained with the "Low cost 2D image to facet model" algorithm in detail. In Chapter IV, computing vector map from the spatial map with less complexity is explained through a spatial understanding features of Microsoft HoloLens in detail. In Chapter V, the results, and analysis of results are explained. Finally, Chapter VI summarizes the overall research contributions, and possible future research.

## II. LITERATURE REVIEW

The computation of indoor vector maps, and spatial mapping are active research fields for various AR (AR) applications. Characteristic examples are gaming, interior design, property advertising, indoor security, indoor navigation, that all required information of the indoor space to overlay holograms. Spatial mapping requires high-end cameras like RGB depth (RGB-D), and Simultaneous Localization, and Monocular (SLAM) cameras, and this increases the overall cost of the system [1]. Spatial mapping is the process of analyzing the 3D space and transforming it into a set of vertices coupled with other information such as vertices normal, and vertices type. In most applications, this transformation is beneficial since a user can place holograms, and avatars in the real space, and interact with them. In some occasions, other applications may require a simplified spatial mapping where the overall objective is just to create the vector map of the walls, and doors of the indoor environment without the need for indoor clutter information. This research proposes a novel technique that can provide the 3D mapping of indoor spaces utilizing the facet concept, and only requires the use of a commodity smartphone cameras.

Different types of AR algorithms, and limitations for real-time imaging are discussed in [2]. The presented applications are used in case of the military, medical, gaming, interior designing, and advertising [2]. A survey of AR technologies, and applications is also presented in [3][4]. The growth of augmented reality leads for the research, and development of sophisticated spatial mapping, and 3D indoor mapping algorithms.

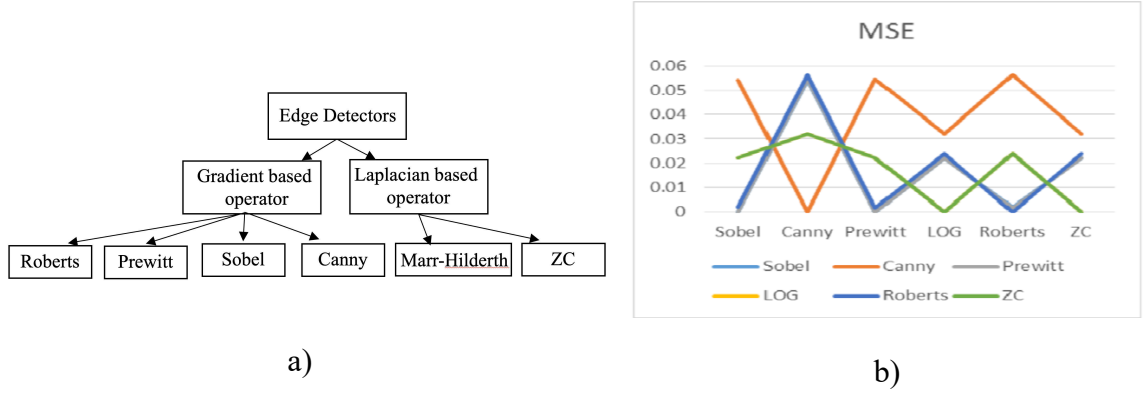
The existing spatial mapping is computed using RGB-D cameras [5][6], and SLAM cameras [7][8]. The RGB-D camera captures 3D RGB images with their depth details. SLAM cameras simultaneously map the indoor environment with localization of indoor environment features, and clutter. Both RGB-D, and SLAM cameras are integrated in expensive AR devices. The indoor environment map computed using SLAM camera, and the detected boundary is shown in Figure 1.



**Figure 1. a) Indoor environment mapping using SLAM camera b) Detected boundary of an indoor environment [29].**

The next generation of communication networks, namely, the 5G networks, are expected to create new opportunities for mobile AR applications [9]. One characteristic application is network visualization, and human-to-network interaction. For example, a user can visualize the results of ray tracing simulations that are overlaid on top of the physical space with the use of an AR application. This is very important for 5G networks where short-range communications are expected to create extensive indoor network planning challenges [9]. To perform field strength prediction, ray tracing algorithms use the facet model, where the indoor environment is represented in a vector format with facets incorporating data of the coordinates, and the material structure of

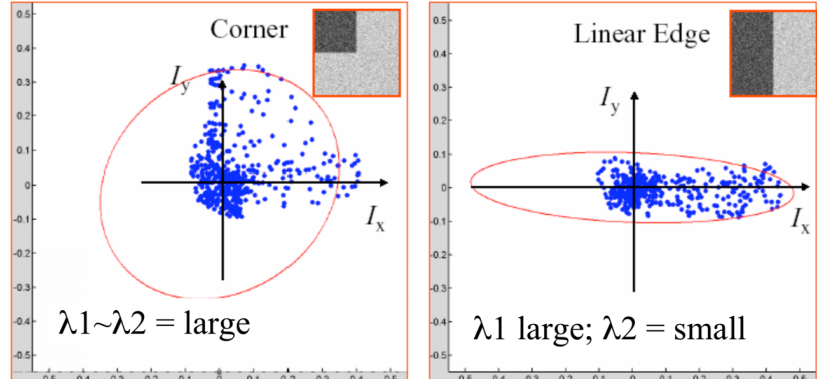
each facet [10]. An example of the use of indoor vector maps for ray tracing algorithms is given in [11].



**Figure 2. a) Different types of edge detection algorithms, b) Different edge detector performance based on the threshold value [9].**

The proposed low cost 2D image to facet model algorithm uses a simple camera of a smartphone device that captures images of individual walls and is capable of constructing a simplified 3D map of the indoor space. The 3D map is a facet model that can be used by indoor channel estimation algorithms. The AR application then overlays the ray tracing results to enable a better human to network interaction. The facet model is created by identifying the coordinates, and dimensions of the walls, and doors of indoor environment. This process incorporates image processing techniques responsible for the edge, and corner detection. Different existing types of edge detection algorithms, and their performance with respect to the Gaussian threshold value is shown in Figure 2. The proposed solution uses the Canny edge detector to extract wall, and door boundaries [12]. Corners on the found edges are detected using the concept of detect minimum eigenvectors algorithm.

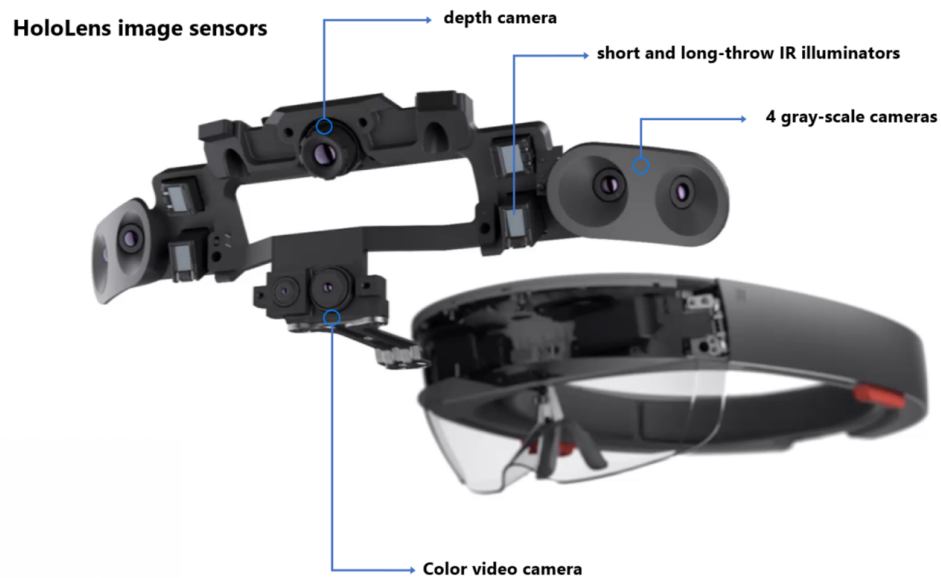




**Figure 3. Corner detection algorithms conditions for differentiating flat, linear edge, and corners [14].** The distribution of x and y deraivative can be characterized by the shape and size of the principal component ellipse.

An interesting analysis, and comparison of corner detection techniques are given in [13]-[15]. Corner point determination on the basis of lambda values are explained in Figure 3. Based on the detected corners of the walls, and doors of individual rooms, the entire indoor environment is synthesized to create a full 3D vector representation. The overall objective of the proposed solution is to create the necessary foundations for the efficient network planning, and positioning of femtocell stations with the use of a typical smartphone device, and AR applications. The above low cost 2D image to facet model algorithm discussed can measure the dimensions of individual wall efficiently and compute the indoor environment map. However, to get efficient results the user needs to capture the complete wall in a single picture [21]. This can be achieved easily in most of the medium sized apartments. If the user wants to map the indoor environment of big spaces like office, college or other buildings with large wall spaces. Capturing the complete wall into a single picture will be hard. In these cases, we can use the spatial mapping technique.

The spatial mapping technique is an active research area in all the AR devices, and applications. Spatial mapping technique is an ability to understand, and map all the indoor spaces including the objects present inside it. The Microsoft HoloLens have the efficient inbuilt spatial mapping feature which maps the indoor environment. The Microsoft HoloLens uses the spatial mapping technique in various mixed reality applications [18]. The sensors, and cameras present inside the Microsoft HoloLens are explained in Figure 4. Expanding the capabilities of the spatial mapping technique might become useful in several different technologies [20]. Out of the different technologies, we are aiming to implement the spatial mapping technique on indoor network planning [12]. For that, the indoor vector maps are extracted, and computed from the spatial maps. These vector maps are used as input to the ray tracing algorithm [11].



**Figure 4. Microsoft HoloLens hardware details [19].**

The ray tracing algorithm models the propagation of electromagnetic signals. The signal interaction, and losses concerning to the physical world like walls, buildings, etc. can be visualized through the ray tracing algorithm. The geometric optics in ray tracing algorithm decomposes the total signal strength from the source into a different number of rays. These rays carry a different value of phase, and amplitude which can be visualized in the ray tracing output. This technique of visualizing the signals, and signal strength enables the user to understand the signal propagation, and interaction in a better, and easier way [11].

To understand the physical world that interacts with different signals, we are using the indoor spatial maps. These spatial maps captured from the Microsoft HoloLens contains a lot of unwanted indoor object details. For the indoor network planning, and ray tracing algorithm, we need only the 3D-coordinates of indoor space, and their wall dimensions [21]. The Microsoft HoloLens uses the depth sensor to measure the distance between the Microsoft HoloLens, and the target object [26]. The depth sensor can measure a distance of up to 3.1 meters, and an area of up to 4.25 square meters [26]. If the user wants to measure a distance above that, the user has to move forward, and map the environment. The depth sensor considers each initial point of mapping as the origin. If the user changes the position of the Microsoft HoloLens, the depth sensor measures the coordinate distance from the changed positions, and results in the complex coordinate system [21]. Currently, The Microsoft HoloLens is not providing access to depth sensor data. Therefore, we are not able to convert the multiple complex coordinates into a single

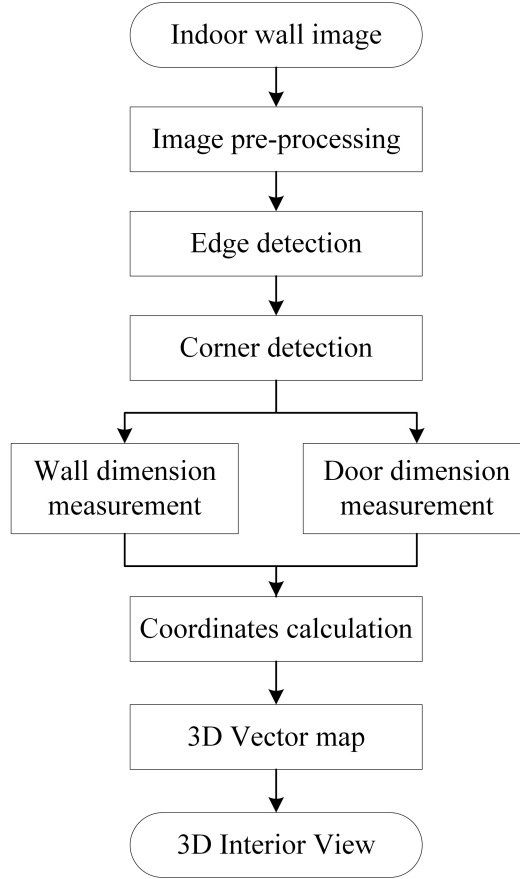
coordinate system [25]. Thus, to have a single coordinate system, the user should not change the position of the Microsoft HoloLens and have to map the environment from the same position. Fortunately, Microsoft unconstrained a Mixed Reality HoloToolkit software development kit [22]. This software development kit provides the different functionalities to relate, and understand the spatial maps for developing different mixed reality applications [12][24][25].

In this proposed research, based on the Microsoft HoloLens depth sensor data, we have designed two different algorithms. These algorithms work on two different methodologies. The primary objective of the two algorithms is to remove unwanted spatial map data, and to extract only the useful information like wall width, breadth, and room area. The two different algorithms are:

1. Minimum-Maximum algorithm
2. Spatial understanding algorithm

Using the extracted data from the above two algorithms a simple vector map of a building is built. These indoor vector maps are computed based on the facet model, which are easier to understand for the ray tracing algorithm [12]. This ray tracing algorithm displays the signal strength, and its propagation. From which the users can have a better scientific understanding of signal strength, and signal interaction with the physical world. This process opens a new door in indoor network planning that can be performed by any non-technical users, and non-experts in the field [21].

### III. LOW COST 2D IMAGE TO FACET MODEL



**Figure 5. Flowchart of 3D indoor facet mapping.**

#### Overview

The algorithm processes images of the indoor environment to identify the walls, and doors positions, computes the coordinates, and creates the facet model for each wall, and door. For the purpose of this investigation, window detection was omitted. This process is performed for each room of the indoor space, and the found facets are combined in a data structure to represent the entire indoor environment. This process can be considered as a simplified spatial mapping technique that neglects the detailed furniture clutter since it is not significantly affecting signal propagation. The input of the algorithm are the images

of every wall but also the height of the ceiling. The images can be captured using a standard camera of a typical smart phone device, without the need of using an expensive depth camera. The input images are then pre-processed to enable an efficient edge, and corner detection process which is important for the identification of the vertices, and coordinates of the walls, and doors. The 3D Cartesian coordinates of a room are calculated using the length, width, and height of the room which is computed once the wall, and door vertices are detected. Using these coordinates, the 3D vector map or else the facet model can be constructed, and become available to third party applications such as ray tracing, and AR.

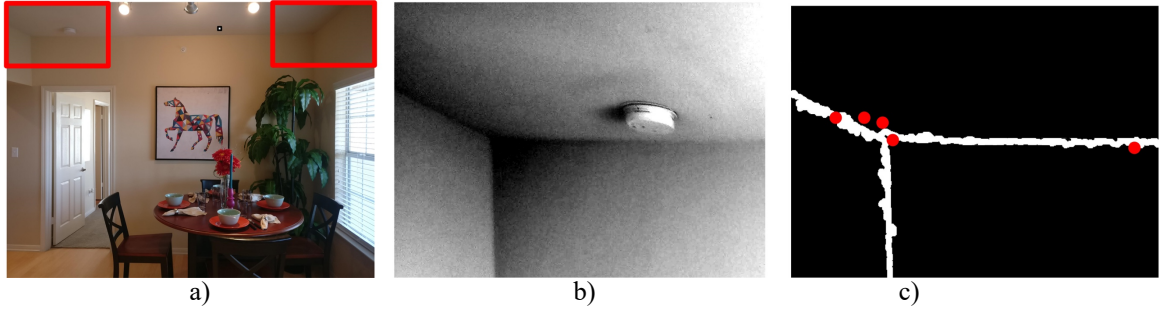
The detailed overview of the proposed solution is presented in Figure 5 and is analyzed in the following sections. For efficient performance of the algorithm, the following assumptions should stand:

- Capture photo of the wall from the center of the room by standing parallel to the wall
- The captured image must be clear without any clutter near the top corners of the walls
- If the wall is large, the user can use the panorama function of the smartphone device to capture the entire wall in a single image file

In practice, the aforementioned conditions are usually met in most typical residential units. It should be noted that the proposed technique cannot be used for large corporate offices, since a wall is usually large enough, and cannot fit in one photo screen.

## Image pre-processing

The image pre-processing is the first step of the overall technique and prepares the images of the room for the edge, and corner detection phase. For the efficient edge, and corner detection, the input image is converted into a grayscale image [5]. The second step of the pre-processing phase is to crop selected regions of interest from the gray scale image. For the purpose of our investigation these are the top, and left corners of the wall as shown in Figure 6. The regions of interest are used to minimize unwanted edge, and corner detection, and reduce the computational demands of the algorithm.



**Figure 6. a) The two regions of interest on the original wall image, b) Pre-processed image of top left region, c) detected edges, and candidate corners.**

The last part of the pre-processing phase corresponds to a down sampling of the image pixel size procedure on the cropped images that further reduces the computational demands of the process. Usually, the image can be convolved with a Gaussian filter to reduce the number of unwanted edges [9]. The smoothing process [9] is given in the following formula:

$$S[i, j] = G[i, j; \sigma] * I[i, j] \quad (1)$$

, where  $I[i, j]$  denotes the input image of pixel size  $i \times j$ ,  $G[i, j; \sigma]$  denotes Gaussian smoothing filter, and  $S[i, j]$  denotes the array of smoothed data, and  $\sigma$  is the gradient level of the filter.

Image is down-sampled to different resolutions like 1280x768, 960x720, 640x480, and experimented for best corner detection results. Images with low resolution 640x480 help to reduce the number of false corner detection compared to higher resolution images. A 5x5 size Gaussian filter is used for efficient edge, and corner detection [13]. The overall process of the image pre-processing is demonstrated in Figure 6 a), and Figure 6 b). The next phase of the proposed solution is to process those images for edge and corner detection.

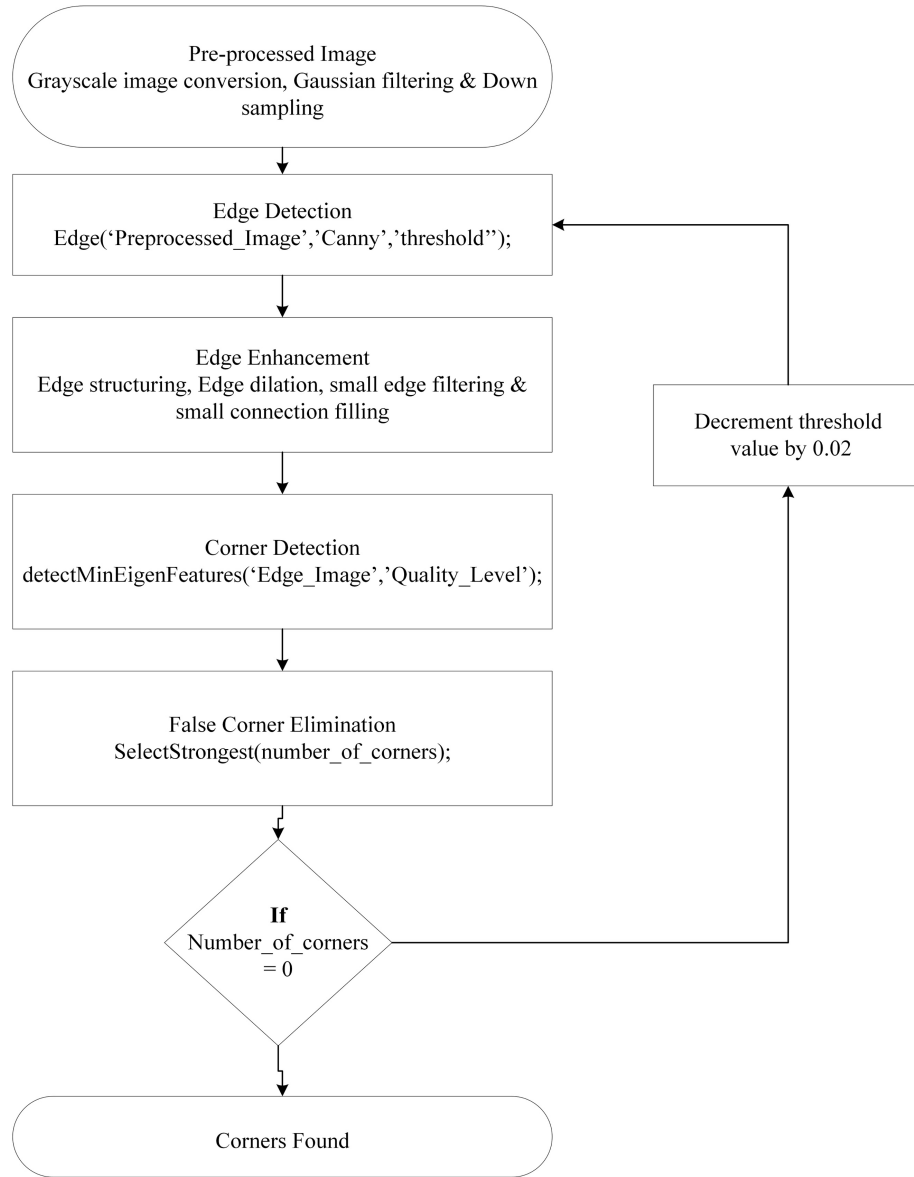
### Edge and corner detection

The edge, and corner detection of the wall image is the most crucial part of the algorithm. This is because, corner detection is directly related to the coordinates of the wall of the room, and thus the development of the facet model. The edge, and corner detection flowchart is given in Figure 7. The first part of the algorithm is to perform edge detection upon the preprocessed input wall images by implementing the edge *Canny* method [9]. The *Canny* method calculates the gradient using the derivative of a *Gaussian* filter, and uses two thresholds to identify strong, and weak edges. With this approach, the edge detection of unwanted noisy parts of the image is minimized. The *Canny* method uses a threshold to distinguish between strong, and weak edges. For the purpose of our investigation, the edge is detected according to the following function.

$$EM = edge(S, Canny, \delta, \sigma)(2)$$

, where  $S$  denotes the pre-processed image,  $Canny$  denotes the edge detection algorithm,  $\delta$  is the threshold used, and is a two-element vector,  $\sigma$  is a scalar of the standard deviation of the *Gaussian* filter, and  $EM$  denotes the edge map of the wall image.





**Figure 7. Flowchart of edge detection, and corner detection.**

The *EM* image is a binary matrix with *I*s representing the points where an edge is detected. The threshold value is a sensitivity value and is used to ignore all edges that are not stronger than the selected threshold. The initial threshold was set to  $\delta=0.4$ , and if no corners found, it decrements by 0.02. The standard deviation was set to  $\sigma=sqrt(2)$ .

The corner detection is the second step of the process during which the edge map of the image is processed for the identification of the candidate corners. The output of the

low cost 2D image to facet model algorithm is a set of potential points that can be considered corners of the walls, as shown in Figure 6. c). The red mark corresponds to the set of potential points. It is observed that the corner point that falls on the intersection of the three edges is the preferred wall corner. The identification of the final corner is described in the next section. For the purpose of our investigation, the *detectMinEigenFeatures* low cost 2D image to facet model algorithm [14] was used. This is a MATLAB function, and has the following structure:

$$Corner = detectMinEigenFeatures (EM, q, G); \quad (3)$$

, where  $EM$  denotes the edge map in gray scale (binary),  $q$  is a scalar value between  $[0, 1]$ , and denotes the corner strength, and quality. Larger values of  $q$  are used to eliminate erroneous corner points. For the purpose of our investigation, the value was set  $q=0.5$  because the pre-processing phase of the image eliminates the majority of erroneous points. The function returns an object file called *Corner* that incorporates location of corners in pixel coordinates  $i, j$ , and the corner metric value,  $C_{metric}$ . Larger corner metric indicates a strongest candidate for a corner [13]. Parameter  $G$  is the Gaussian filter dimension, and is an odd integer value in the range  $[3, \infty]$ . For the purpose of our investigation, we set  $G=3$ . The Gaussian filter is used to smooth the gradient of the input image. The minimum Eigen values of the low cost 2D image to facet model algorithm is computed using the

following formula [14]:

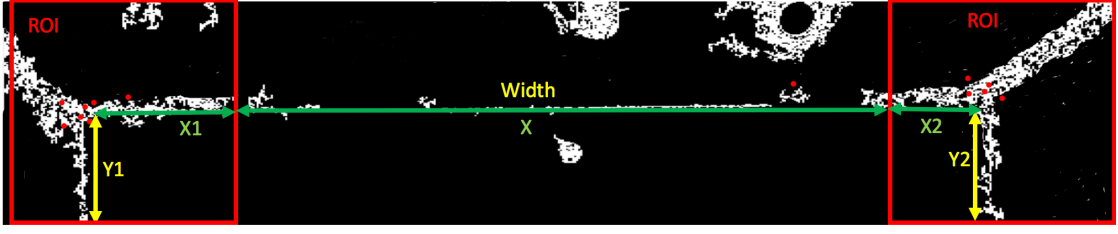
$$C_{metric} = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad (4)$$

, where  $I_x$  denotes the horizontal gradients of the edge map,  $I_y$  denotes the vertical gradients of the edge map,  $I_x I_y$  denotes the edges on diagonal.  $C_{metric}$  denotes the matrix with two *Eigen* values  $\lambda_1, \lambda_2$  characterized by their shape, and size of the principal

component ellipse inside each filter of an image were computed. According to the used parameter  $q$  the output of the low cost 2D image to facet model algorithm may not provide any candidate corner points. In that case, the algorithm reduces the corner quality parameter  $q$  with a step of 0.05 until corner points are detected. This process is also presented in Figure 7. The corner detection phase ends with the detection of at least one or more strong candidate corner points with a corner metric value above the quality level. The same procedure is performed for the bottom corners of the wall. Thus, the output of the corner detection process is a set of corner points for each region of interest of the wall. For the top left region of interest (ROI) of the wall, the output is a set of points  $(x_i, y_i)$ ,  $i \in TL$  where  $TL$  indicates the number of found corners for this region of the wall. Respectively, for the top right part of the wall the potential corner points are  $(x_j, y_j)$ ,  $j \in TR$ . The bottom left part includes the candidate corner points  $(x_m, y_m)$ ,  $m \in BL$ . Finally, the bottom right part of the image includes the candidate corner points  $(x_n, y_n)$ ,  $n \in BR$ .

### Computation of wall dimension

This part of the algorithm provides an estimation of the wall width according to the detected candidate corners. These corner points may include both good candidate corners but also erroneous corners. In order to avoid the negative effects of the erroneous corners in wall width measurements, the best candidates should be determined. For that reason, each corner point in all regions of interest are compared with each other. The corner points from the top part of the wall that have the same  $y$  value  $y_i \sim y_j$ , where  $i$ , and  $j$  are the two candidate corner points from the top right, and top left part of the wall, are preferred. In addition, the corner points from the top left, and bottom left part of the wall that have the same  $x$  value  $x_i \sim x_m$ .



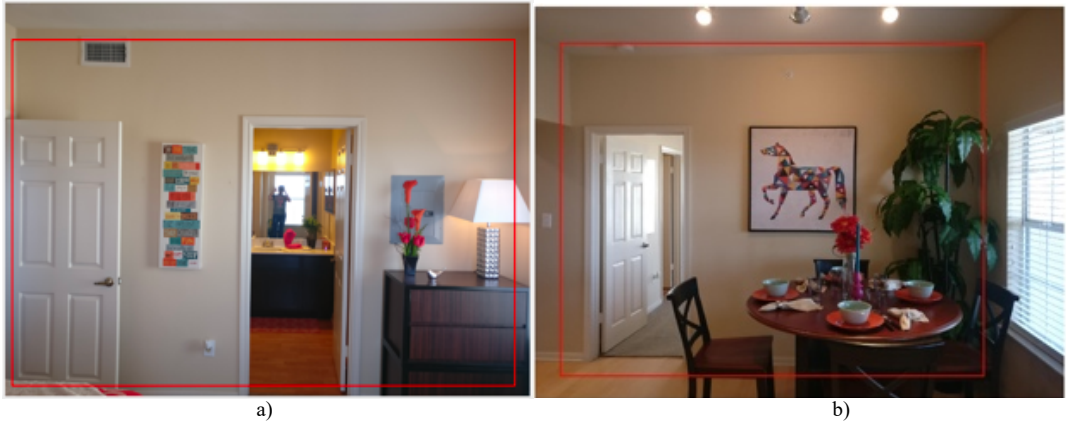
**Figure 8. Corner points matching between two regions of interest of a wall image.**

$$i^*, j^*, m^*, n^* = \min_{i,j,m,n} [|x_i - x_m| \cdot |y_i - y_j| \cdot |y_m - y_n| \cdot |x_j - x_n|] \quad (5)$$

, where  $i$ , and  $m$  are the two candidate corner points from the bottom, and top left part of the wall, are preferred. Similarly, the same procedure occurs for the bottom left, and bottom right, and also for the top, and bottom right part of the walls. The final corner detection is computed according to the overall process is shown in Figure 8. The width,  $w$ , of a room wall is calculated by measuring the pixel distance between the two final corners.

$$w = \frac{h}{|y_{i^*} - y_{m^*}|} \cdot |x_{i^*} - x_{j^*}| \quad (6)$$

, where  $h/|y_{i^*} - y_{m^*}|$  is the pixel resolution  $r_p$  measured in meters/pixel. The pixel resolution can be computed according to the height of the wall,  $h$ , which is defined by the user, and the number of pixels between the two corners. In a mathematical form, this is presented in (6). The detected wall boundary is demonstrated in below Figure 9.



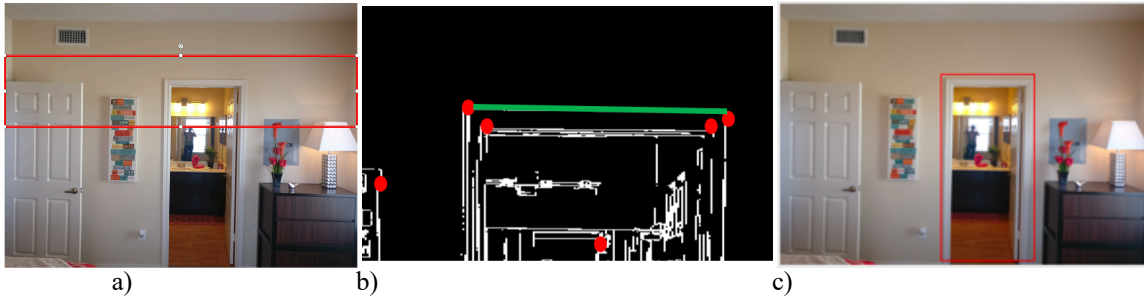
**Figure 9. Detected wall boundary.**

## Door detection

The door detection process follows a similar approach where an edge, and corner detection algorithm is used to find the location of the boundaries of the door [17]. An illustration of the overall process is given in in Figure 7. For the door detection, the region of interest is focused above the half of the wall, and below the third quarter of a wall. This is because, most doors found in typical residential units have these height values. To increase the efficiency of the door corner detection algorithm, the following conditions were assumed:

- The preferred door corner should have a y-axis value equal to a standard door height of 2.1 meter. Thus,  $r_p \cdot |y_i - y_m| = 2.1\text{m}$ .
- Two corner points should have the same y-axis values. Thus,  $|y_i - y_j| \sim 0$ .
- Two corner points should be separated by a standard door width 0.9 meters. Thus,  $r_p \cdot |x_i - x_j| = 0.9\text{m}$ .

Similar to the wall detection process, the algorithm first identifies the position of the door boundaries, computes the door dimension, and defines the coordinate values of its corners. The detected door boundary is as shown in Figure 10.



**Figure 10. (a) Input image with region of interest (ROI) selected. (b) Detected door corners over edge map. (c) Detected door on a wall**

#### IV. SPATIAL MAPPING

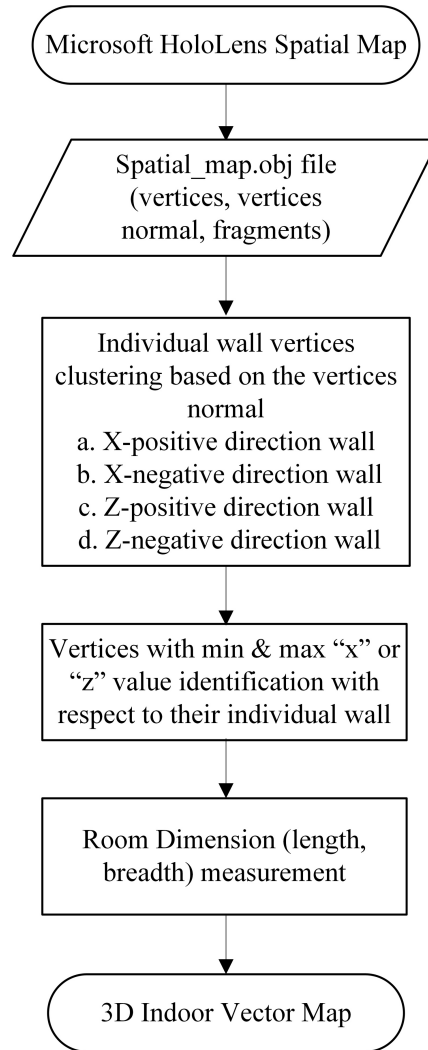
Spatial mapping is a mapping technique mapped using a depth sensor where it computes the vertices, and vertices normal of the indoor environment, and all the furniture, objects or things present inside it. The Microsoft HoloLens uses the depth sensor, and a laser ray cast to measure the vertices or depth distance between the HoloLens, and the wall, ceiling, floor, and different object or things distance. Based on this depth measurement, the vertices of all spaces are computed into an object file. These spatial maps are sophisticated and can be used in varieties of complex AR applications. However, for our proposed problem, we need a simple vector map with only the room coordinates. So, we are using the spatial map from the Microsoft HoloLens, and we will convert the spatial map into a vector map. Since Microsoft HoloLens is using a well-sophisticated depth sensor, it can map large commercial room wall spaces. This feature of mapping large wall spaces will clear the limitation of our previous algorithm which is aimed at residential spaces.

In the following proposed solution, two different algorithms are discussed, they are:

1. Minimum-Maximum algorithm
2. Spatial understanding algorithm

The two algorithms are explained briefly in next chapters. The minimum-maximum algorithm is designed to work in MATLAB. It uses the spatial map object file of the HoloLens as input and converts the spatial map into a simple vector map. The second algorithm i.e. spatial understanding algorithm uses the inbuilt HoloLens algorithm in C#, and Unity,

## Minimum – Maximum Algorithm



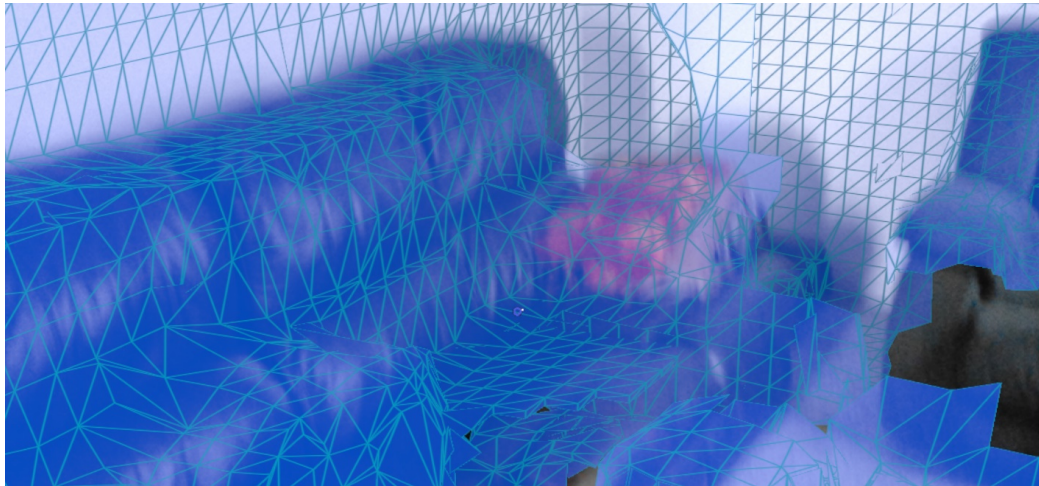
**Figure 11. Flowchart of Minimum – Maximum Algorithm**

### Overview

The algorithm processes the spatial maps of an indoor environment captured from the Microsoft HoloLens. These spatial maps consist of vertices, vertex normals, and fragments. Based on the vertex normals, vertices of an individual wall are identified, and

named after their directions. Vertices with minimum, and maximum “x” or “z” values of all the walls are identified. Thus, for each individual wall, 2 vertices of minimum, and maximum values both “x” axis, “z” axis or each from “x”, and “z” axis is identified. The dimension of each wall will be the distance between the 2 vertices points. Using the dimensions of all 4 walls of a room i.e. length, and breadth, the 3D indoor vector map will be computed, and used as an input to ray tracing algorithm to visualize signal strength, and signal propagation. The overview of the minimum-maximum algorithm with individual steps involved in it is as shown in Figure 11.

### **Spatial mapping process**



**Figure 12. Spatial map of an indoor environment.**

Before, working on the spatial coordinates of the spatial map. We need to understand, how the spatial map is captured inside the Microsoft HoloLens. The spatial map of an indoor environment captured from the Microsoft HoloLens is as shown in Figure 12. The spatial mapping technique involves both hardware, and inbuilt software functionalities of the Microsoft HoloLens. Depth sensor is used in the HoloLens to measure the distance between the HoloLens, and the target object or the target surface.



The depth sensor computes 3D cartesian coordinate system in terms of triangles of multiple vertices. It has the capacity to measure a distance of up to 3.1 meters, and If we want to measure a distance above that, the user wearing the HoloLens should step forward till the sensor reaches the target and have to map the surrounding environment. Thus, using a depth sensor in the HoloLens standing at one point we can only build a spatial map with a single coordinate system of area 4.25 square meters.

If the user wants to map the area above that, the user needs to step forward, and the HoloLens computes multiple different coordinate systems with different origins, resulting in complex, and invalid coordinate system. Thus, in order to measure the wall dimensions of a room, the user needs to stay in the same position, and have to map the indoor environment. Using this algorithm, the user can map a room of an area up to 4.25 square meters, which works best in common residential apartments.

### **Spatial map data**

The indoor environment is scanned through Microsoft HoloLens, and a detailed spatial map for the same is generated with 3D (x, y, z) spatial coordinates of all the spaces, and objects present inside it. The spatial map data can be saved or exported from HoloLens to a computer in a .obj file format. The spatial map data will consist of 3 main information i.e. vertices, vertices normal, and fragments. The Microsoft HoloLens maps the data in terms of small triangles where the vertices represent each point of a triangle. Vertices normal represents the direction of the object or the wall direction. Fragments data represents the graphical representation of the spatial map. Figure 12 demonstrates the spatial map of an indoor environment mapped through Microsoft HoloLens.

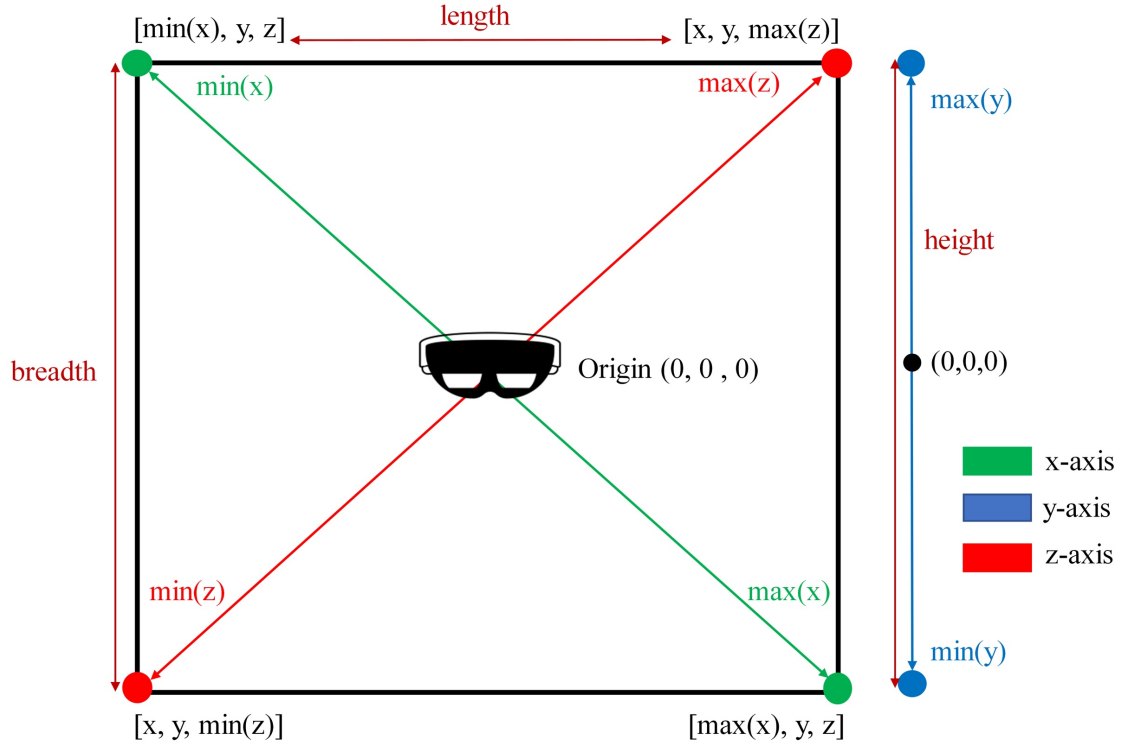
### **Individual indoor wall vertices clustering**

The extracted spatial map data is then imported into MATLAB for data analyzing, and processing of useful information. The main objective of the "Minimum & Maximum" algorithm is to find the precise dimensions of a room using the vertices & vertices normal data of an indoor environment spatial map. Since fragment data is used for graphical representation, we excluded its data, and clustered only the vertices, and vertices normal data from the spatial map .obj file data. Vertices normal is the key variable which can be used to identify the direction of an indoor space, objects, and walls so, based on the vertices normal direction, vertices of all 4 different walls i.e. wall1, wall2, wall3 & wall4 are grouped separately. Where:

- wall1 = X positive direction
- wall2 = Z positive direction
- wall3 = X negative direction
- wall4 = Z negative direction
- ceiling = Y positive direction
- floor = Y negative direction

Now, we are having 6 different vertices groups with the right hand (x, y, z) 3D-coordinate system. Each unit in the coordinate system is measured in terms of meter unit i.e. (x=1m, y=2m, z=3m). Since the spatial map is done at the same position, and the coordinates distance values are measured from the HoloLens. We can consider HoloLens position as an origin.

### Computation of wall dimension



**Figure 13. Minimum-Maximum vertices Identification**

Now, we will find the maximum, and minimum values of the x-axis, and z-axis as shown in Figure 13. Since we are measuring the coordinate distance from the same position or the origin, the maximum, and minimum vertices x-axis, and z-axis values are always will be near the corner of the mapped environment, and the orientation of the coordinate system will always remain same as in Figure 13. Identify the respective coordinates of the  $\max(x)$ ,  $\min(x)$ ,  $\max(z)$ ,  $\min(z)$ . Using the 3D cartesian distance formula, find the distance between the two corner point vertices, and these distances will be the width of the individual wall. Using these dimension, length, and breadth of a room are calculated i.e. length is the average of 2 parallel walls wall1, wall3, and breadth is the average of other 2 parallel walls wall2, and wall4. Similarly, the height of the room is

also computed by measuring, and adding the max(y), and min(y). Thus, of the room is also computed by measuring, and adding the max(y), and min(y). Thus, (length, height, breadth) i.e. (x, y, z) is measured, and using this 3D coordinate values, simple, and clear vector map suitable for ray tracing algorithm is built.

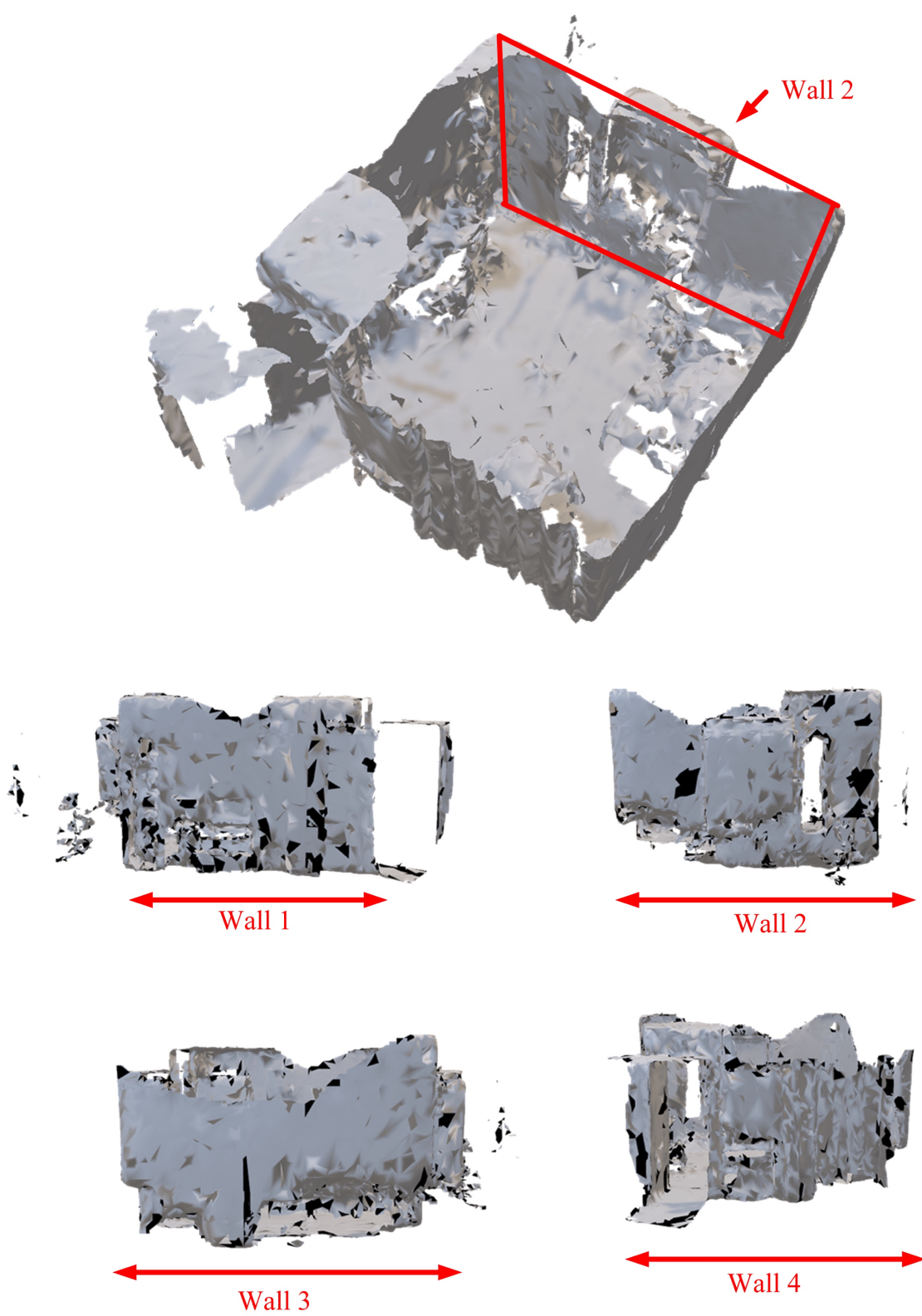
$$Wall1 = dist\{ Vert[max(x), y, z], Vert[x, y, min(z)] \} \quad (1)$$

$$Wall2 = dist\{ Vert[max(x), y, z], Vert[x, y, max(z)] \} \quad (2)$$

$$Wall3 = dist\{ Vert[min(x), y, z], Vert[x, y, max(z)] \} \quad (3)$$

$$Wall4 = dist\{ Vert[min(x), y, z], Vert[x, y, min(z)] \} \quad (4)$$

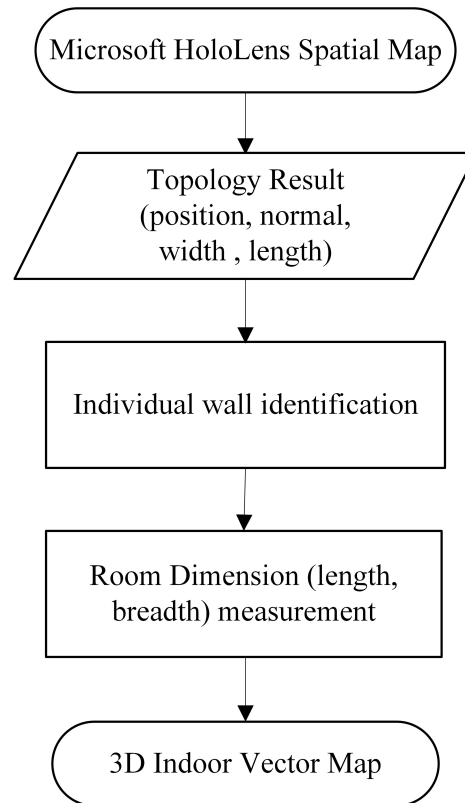
The spatial map of an indoor environment is as observed in below Figure 14 a). The spatial information is complex, and hard to visualize. These spatial maps are used as input to the minimum-maximum algorithm, and all 8 corners of an indoor environment are identified. Based on the corners, individual walls, i.e. wall1, wall2, wall3, and wall4 are identified. Figure 14 b), c), d), e) shows the individual walls of an indoor environment. The dimensions of these wall samples are calculated, and the coordinates of the room are recorded in the data structure.



**Figure 14. a) spatial map of an indoor environment. b), c), d), & e) Individual walls of an indoor environment.**

## Spatial Understanding Algorithm

### Overview



**Figure 15. Spatial understanding algorithm**

The main limitation of the Minimum-Maximum algorithm is that we have to scan the room from the same position, and we can only scan a room of area only up to 4.25 square meter. This limitation can be overcome by Microsoft's inbuilt spatial mapping functions. Where the functions integrate the different coordinate, systems computed from different origins into a single coordinate system during the scanning process. Thus, using these functions, the user can scan the indoor environment, so all usable surfaces like the wall, floor, ceiling, etc. irrespective of the initial positions. The user can also move

around in any directions or any orientation and maps the indoor environment. The overview flowchart of spatial understanding algorithm is shown in Figure 15.

### **Topology result**

The single coordinate system computed from multiple different coordinate systems of different origins are stored in Topology result structure object by the spatial mapping function. The Topology result object consist the spatial map data of position or vertices, vertices normal, width, and length. The spatial map of an entire indoor environment is a combination of multiple small spatial maps mapped in a random shape, and order. The spatial mapping functions combine all the individual spatial maps together. It stores individual spatial map data i.e. mentioned in the result topology in a descending order.

```
struct TopologyResult
{
    DirectX::XMFLOAT3 position;
    DirectX::XMFLOAT3 normal;
    float width;
    float length;
};
```

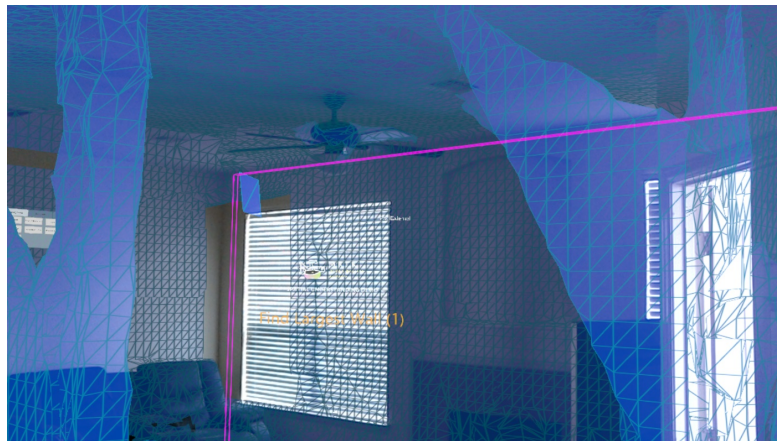
### **Individual wall identification**

The Spatial understanding is another inbuilt function of the Microsoft HoloLens. It has different topology queries to understand the spatial map data stored in the result topology structure. Using spatial understanding functions, the spatial maps of an indoor environment can be understood and used in different varieties of mixed reality

applications. Different objects, and spaces like the chair, table, couch, floor, ceiling, wall, and large open spaces, etc. can be identified. Using the existing topology queries, we can specify the conditions to find the large walls of a room as shown in this code snippet.

```
EXTERN_C __declspec(dllexport) int QueryTopology_FindLargestWalls(  
    _In_ float minHeightOfWallSpace,  
    _In_ float minWidthOfWallSpace,  
    _In_ float minHeightAboveFloor,  
    _In_ float minFacingClearance,  
    _In_ int locationCount,  
    _Inout_ Dll_Interface::TopologyResult* locationData)
```

The above function identifies the large wall spatial maps and stores it in the topology result structure. The largest spaces are stored on the top in a descending order. The first 4 largest spatial map areas identified are the spatial maps of 4 individual walls.



**Figure 16. Individual wall identification using spatial understanding.**



### **Computation of wall dimension**

The spatial understanding function identifies the large spatial map data in the topology result. Computes the position, and normal, and calculates the length, and width of the individual spatial map spaces. The first two large space areas are stored in the result topology will be wall1, and wall3. The next 3rd, and 4th spaces in the result topology will be wall2, and wall4. Using these dimension, length, and breadth of a room are calculated i.e. length is the average of 2 parallel walls wall1, wall3, and breadth is the average of other 2 parallel walls wall2, and wall4. The large wall detection with the spatial understanding is shown in Figure 16 with the pink line boundary of the wall. Thus, we identified the 4 different walls of a room. These dimension values will be used in building a vector map of the building. Using the same algorithm, we tried inserting suitable conditions for the detection of the door in the indoor space. since the Microsoft HoloLens considers the doors same as walls, and there is no much considerable space difference between wall, and a door like the difference between a floor, and a table. We weren't able to identify the doors inside the indoor spaces.

After successful execution of the Minimum–Maximum algorithm, the spatial map data is been processed, and structured into different vertices groups based on their vertices normal. Each wall group coordinates are plotted to see the specific spatial map of its own wall group. The spatial map of 4 different wall groups are shown in Figure 14. Using these walls, we can visually know the width, and length of the walls. Also, we can understand how the spatial mapping is done from the Microsoft HoloLens. We can also observe from the plots the maximum distance measured from the Microsoft HoloLens i.e.

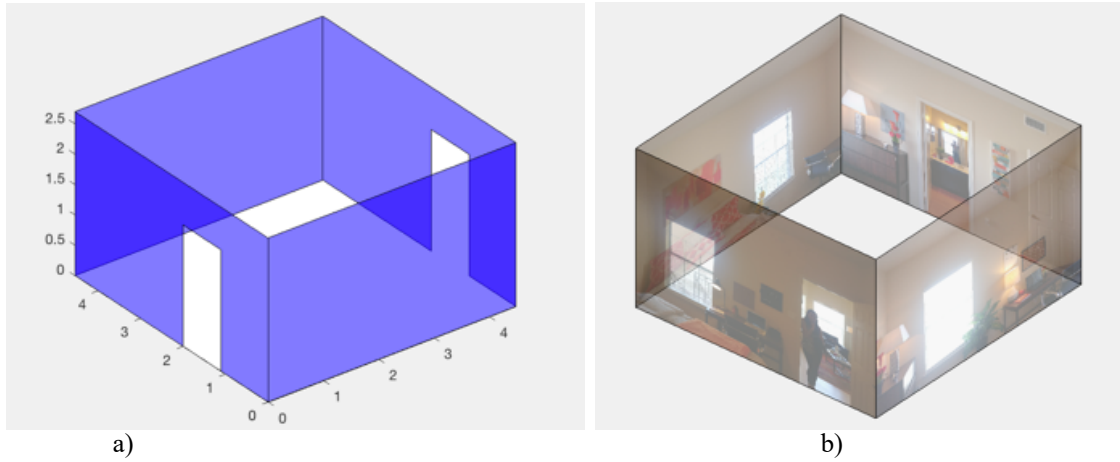
origin is less than 3.1mt, and the total area will be less than 4.25 square meter. The detected large wall in the indoor space are as shown in Figure 16, where the Spatial Understanding function understands the spatial map of an indoor space. It also separates all the spatial map areas into different individual areas based on their directions, and their length, width, and area. The spaces with maximum areas are stored on top of the result topology array. Using these result topology data, we can extract the dimensions or spatial map space of 4 individual walls of a room.

Using the above 2 algorithms, we can measure the length, and breadth of an individual room with 8 corners of a room. These coordinates can be stored in a facet model format. Each facet represents an individual wall with four corner coordinates (x, y, z). These coordinates indicate the respective corners. Using the same process, the facet model of all other rooms is computed, and combined together to form a clear vector map of an entire building without any unwanted indoor objects data. The vector map of a 3-bedroom apartment is computed. The finally designed vector map is used as an input to a ray tracing algorithm. The ray tracing algorithm models the electromagnetic wave propagation using the Geometric Optic (GO), and sum of the individual rays carrying different amplitude, and phase. The indoor space details like wall length, breadth area, etc. been loaded into ray tracing algorithm, and the resulting electromagnetic signal variations or channel condition can be visualized inside the modeled indoor vector map. This process opens up a new set of functionalities, and options for non-technical users, and non-experts to perform indoor network planning.

## V. THE FACET MODEL

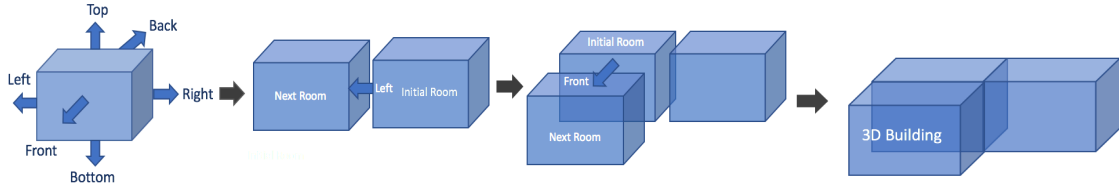
### Constructing the 3D environment

After the successful wall, and door width detection, the final coordinates of the room can be stored in a facet model format. The vector map is represented by its facet where each wall, and door is defined by four coordinate points  $x,y,z$ . These coordinates indicate the respective corners. The facet representation of a single room is presented in Figure 17 a. For more enhanced experience, it is possible to overlay the picture as texture on the facet as presented in Figure 17 b.



**Figure 17. a) 3D vector map. (b) 3D indoor environment interior view.**

Using the same method, and principles, the 3D vector map of the remaining rooms of the indoor environment can be constructed. One difficulty for this case is the positioning of the rooms to form a realistic indoor environment, close to the real one. For the purpose of our investigation, we assume that the user takes four pictures per room to cover the 360 space and takes the pictures in a clockwise manner. Once the user completes this process for one room, then the user takes the pictures of the adjacent room, starting from the wall that is shared with the previous room.



**Figure 18. Integrating individual room blocks into a building based on the direction of next room with respect to initial room.**

In that way, there is always a “calibration” or orientation point that allows the algorithm to reconstruct, and attach the facet of each room, and form a realistic indoor environment. This process is presented in Figure 18. where the first room is marked as initial and attached to the adjacent room according to the shared wall of the two rooms. In the next iteration, the second room becomes the reference room, and the third room is attached according to their shared wall. This process is followed until the user captures images of all rooms of the indoor environment, and the indoor environment is fully constructed.

### **Data structure**

The data structure of the facet model is presented in Table I. The indoor environment is composed by a set of individual rooms. Each room has a number of walls, and each wall may have a number of doors. The elements of the room structure store all the details of the indoor environment like wall width, room number, room position, wall image, wall coordinates, and door coordinates. The room position field is used to determine the position of the room according to the previous one. The wall image is used as a texture and is overlaid on the facet model to enhance the user experience. The pixel size is used for the computation of the dimensions of the walls, and doors length, and width, and may also be used for future applications. The wall coordinates, and door

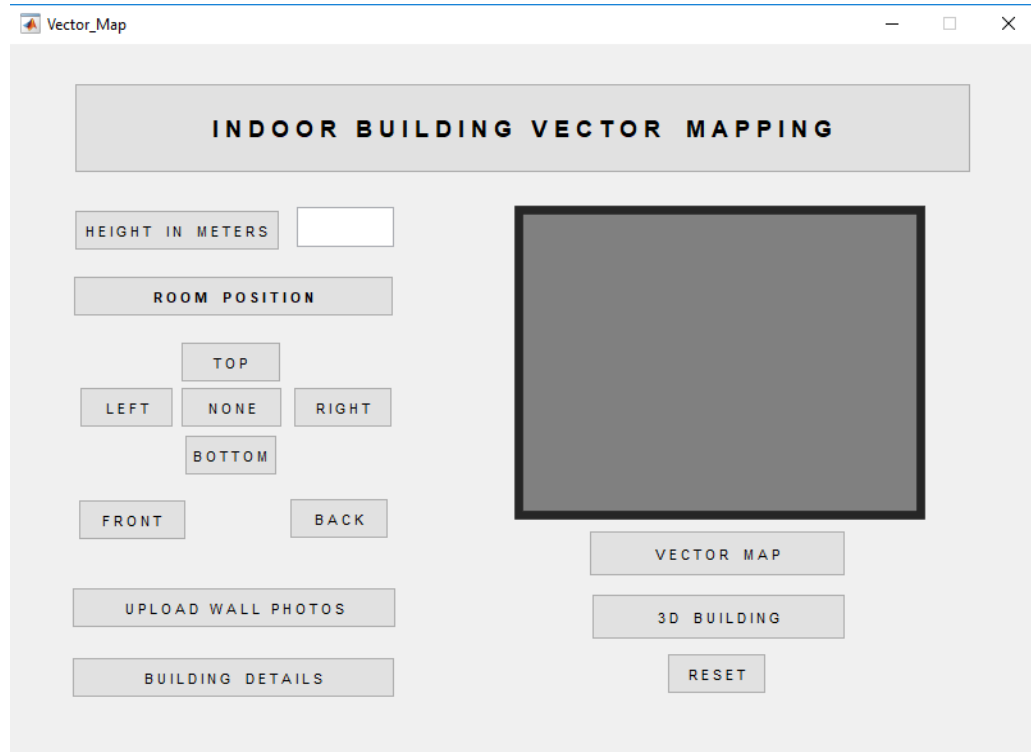
coordinates represent the vector format of the facet, and is the most valuable element of the structure, used by the ray tracing algorithm.

<b>Table 1. Data Structure of the facet model.</b>		
<b>Room Wall Position</b>	<b>Room Properties</b>	<b>Description</b>
Room(i).Wall(j)	Room_Position	Top, down, left, right, front & back position
-----  -----	Wall_Image	Respective room wall image
-----  -----	Width_Pixel	Wall width in pixel size
-----  -----	Width	Wall width in meter
-----  -----	Height	Wall height in meter
-----  -----	Coordinates	Wall (x,y,z) coordinates as a set of four corners
-----  -----	Door	Door (x,y,z) coordinates as a set of four corners

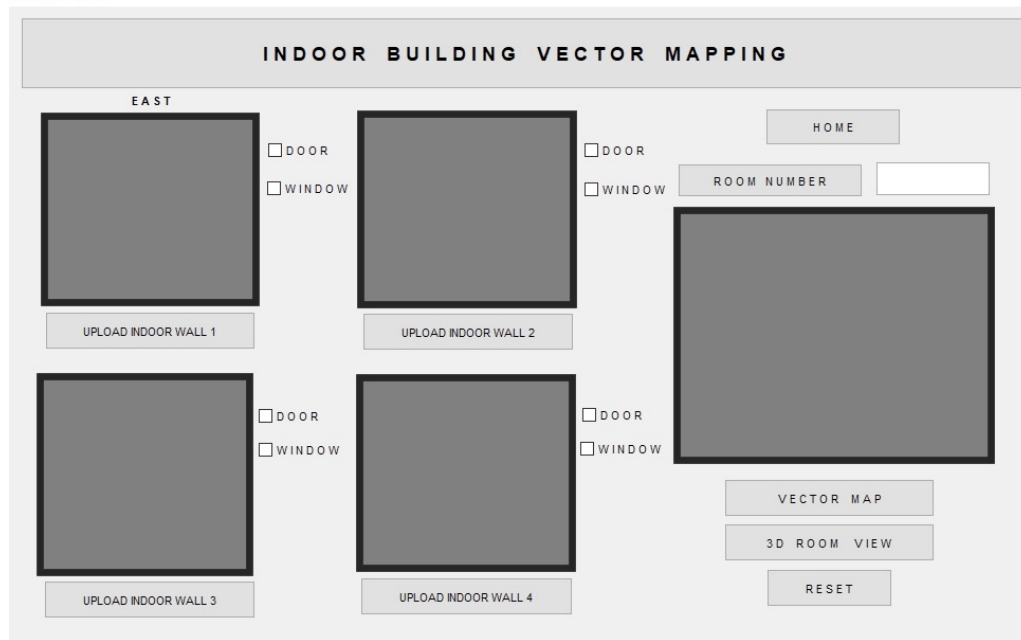
Finally, each facet incorporates its constitutive parameters that are used for the computation of the diffraction, reflection, and transmission coefficients of the ray tracing model. For the purpose of our investigation, the wall was assumed to be made by brick material, and the doors by wood material. The constitutive parameters of these materials can be found in [12]. The details of the indoor environment can be fetched using the ‘Building Details’ button of the main GUI, as described in the following section of the report.

## VI. RESULTS

### Graphical user interface



a)



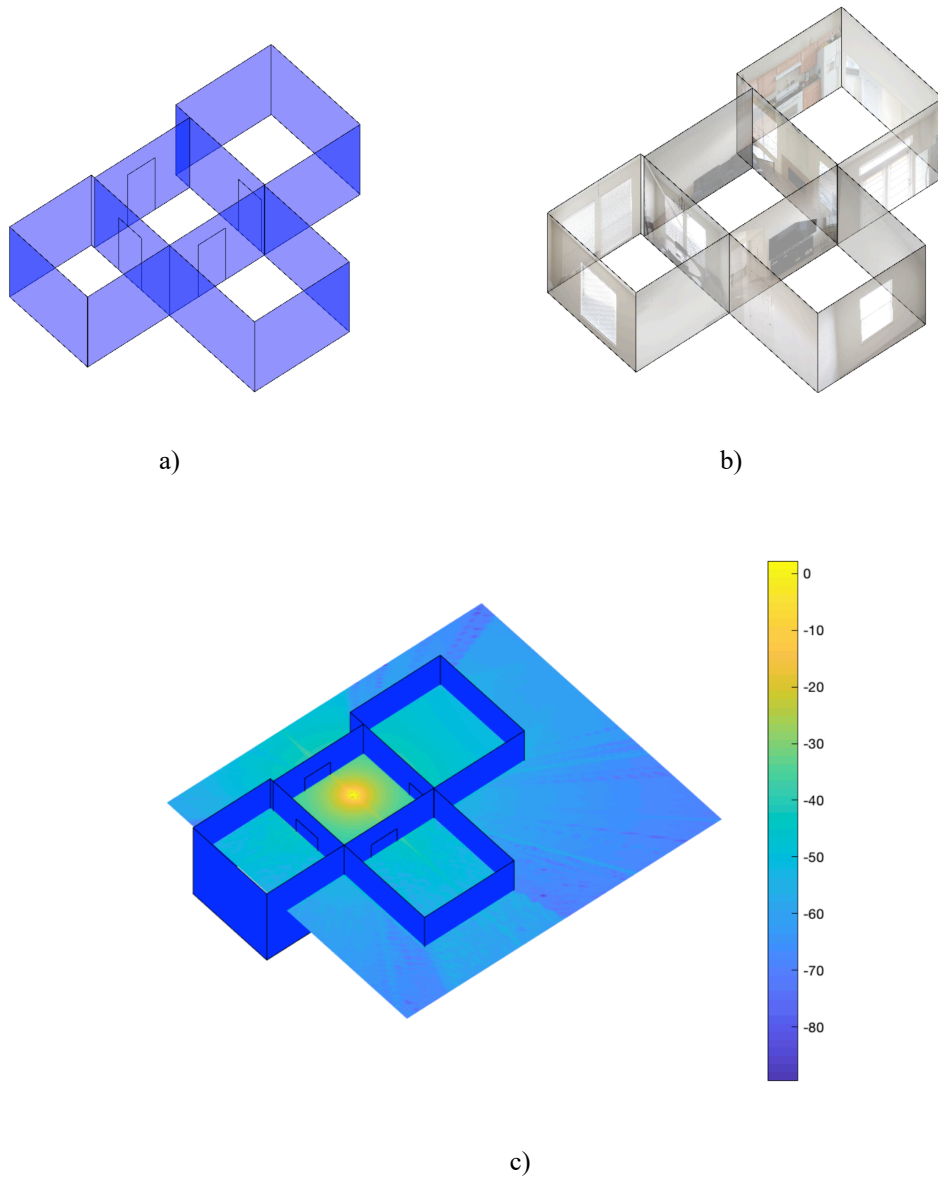
b)

**Figure 19. a) Main GUI of indoor building vector mapping, b) Secondary GUI for uploading images of a room and mapping individual rooms.**

A GUI was designed to make the use of the developed app easy, and user friendly. The user can enter the standard height of the ceiling that is used as reference for the pixel resolution definition. The user also enters the room position that is used as a reference point for the construction of the 3D space. Finally, the user uploads the images for each wall of the indoor environment by using a secondary GUI as indicated in Figure 19. The user can upload four individual wall images per room and indicate if there is a door in the room. The door checkbox was used to reduce the computational cost by eliminating unwanted door detection processes. Once the user uploads the data to the system, the facet model is computed. Within the GUI, there is a button to indicate if there is a window in a wall. For the purpose of our investigation, windows were not incorporated in the facet model, and is something that will be integrated in future versions of the algorithm.

### **Augmented reality to Ray tracing**

The scenario under investigation is presented in Figure 20. A two-bedroom student dorm apartment was examined that has three main rooms. The facet model of the apartment was successfully reconstructed when the user uploads the twelve images of the walls of the three rooms. A commodity smart phone device was used to capture the images. The user spent 3 minutes to take the photos, and upload into the system using the GUI. When the user uploads the images to the system, the algorithm performed the pre-processing phase by down sampling, and applying Gaussian filters. The input images were down sampled to different resolutions, and the best performance was met when the resolution was set to 640x480.



**Figure 20. a) 3D Vector map of a building, b) 3D Interior view of a building, c) Implementation of a Ray Tracing algorithm on the facet model.**

It was found that the most suitable corner detection technique was ‘*DetectMinEigenFeatures*’ of MATLAB since it provided the most accurate results and is widely used by the research community. The found coordinates of all rooms were integrated together to form the facet model of the entire indoor environment. The



processed images are then embedded on to their respective walls to form a 3D interior view which is as demonstrated in Figure 20 b). It should be noted that the user inputs at the GUI, such as the height of the ceilings, the position of different rooms, and the existence of doors on walls, reduced the computational cost by 35%-40%. This is because, the algorithm did not search for doors in case there was no door at the room and made a more efficient positioning of the rooms to form the entire indoor space.

The final step of the proposed system is to use the facet model of the indoor space as input to a ray tracing algorithm [11]. The ray tracing algorithm models the propagation of the electromagnetic waves using the Geometric Optic (GO) technique and decomposes the total field strength as sum of individual rays carrying a different amplitude, and phase. The amplitude was computed as a combination of multiple reflection, transmission, and diffraction coefficients. For the purpose of our investigation the used frequency was assumed to be of the order of the 6GHz band of 5G systems. The results are presented in Figure 20 c). It is observed that the walls, and doors of the environment interact with the electromagnetic waves and change their signal strengths. With the use of the proposed system, the user is able to take 12 images of the walls of the house, and with just a few clicks be able to visualize the signal variation, and channel condition of the 5G femtocell station inside the house. This process opens new frontiers in indoor network planning that can be performed by non-technical users, and non-experts in the field. In addition, it creates new opportunities for the education of indoor channel modelling with the use of AR (AR) devices, and applications.

After successful execution of the Minimum–Maximum algorithm, the spatial map data is been processed, and structured into different vertices groups based on their vertices normal. Each wall group coordinates are plotted to see the specific spatial map of its own wall group. The spatial map of 4 different wall groups are as shown in Figure 14. Using these plots, we can visually know the width, and length of the walls. Also, we can understand how the spatial mapping is done from the Microsoft HoloLens. We can also observe from the plots the maximum distance measured from the Microsoft HoloLens i.e. origin is less than 3.1mt, and the total area will be less than 4.25 square meter. The detected large wall in the indoor space is as shown in Figure 16, where the Spatial Understanding function understands the spatial map of an indoor space. It also separates all the spatial map areas into different individual areas based on their directions, and their length, width, and area. The spaces with maximum areas are stored on top of the result topology array. Using these result topology data, we can extract the dimensions or spatial map space of 4 individual walls of a room.

Using the above 2 algorithms, we can measure the length, and breadth of an individual room with 8 corners of a room. These coordinates can be stored in a facet model format. Each facet represents an individual wall with four corner coordinates (x, y, z). These coordinates indicate the respective corners. Using the same process, the facet model of all other rooms is computed, and combined together to form a clear vector map of an entire building without any unwanted indoor objects data. The vector map of a 3-bedroom apartment is computed and is as shown in Figure 20. a. The finally designed vector map is used as an input to a ray tracing algorithm. The ray tracing algorithm models the electromagnetic wave propagation using the Geometric Optic (GO), and sum

of the individual rays carrying different amplitude, and phase. The indoor space details like wall length, breadth area, etc. been loaded into ray tracing algorithm, and the resulting electromagnetic signal variations or channel condition can be visualized inside the modeled indoor vector map. This process opens up a new set of functionalities, and options for non-technical users, and non-experts to perform indoor network planning.

### Wall dimension measurement comparison

<b>Table 2. Wall dimension measurement comparison between the proposed algorithms.</b>				
<b>Wall Samples</b>	<b>Actual Measurements</b>	<b>Low cost 2D image to facet model algorithm</b>	<b>Min-Max Algorithm</b>	<b>Spatial Understanding Algorithm</b>
<b>1</b>	3	3.15	3.15	2.9
<b>2</b>	3.4	2.95	3.2	3.5
<b>3</b>	5.3	3.15	5.29	5.2
<b>4</b>	3.4	2.95	3.2	3.4
<b>5</b>	1.8	3.06	2.12	1.7
<b>6</b>	4.2	3.5	3.59	4.2
<b>7</b>	2.3	3.06	2.17	2.3
<b>8</b>	3.1	3.5	4.56	3.1
<b>9</b>	2.6	2.71	2.77	2.6
<b>10</b>	2.6	2.71	2.67	2.6
<b>11</b>	2.2	3.07	3.5	2.1
<b>12</b>	3.7	3.07	3.5	3.7
<b>13</b>	3.2	3.35	3.86	3.2
<b>14</b>	2.9	3.13	2.98	2.8
<b>15</b>	3.1	3.35	2.8	3.2
<b>16</b>	2.9	3.13	2.7	2.9

For checking the efficiency, and wall measurement accuracy of all three proposed algorithms, a 2-Bed apartment is considered. Actual measurements of all individual walls are measured using a laser beam measurement device and recorded in column 2. The low cost 2D image to facet model algorithm is applied over the images of all individual walls captured from a smart-phone. The resulting wall dimension measurements are recorded in column 3. The low cost 2D image to facet model algorithm results are compared with the actual dimensions, and the low cost 2D image to facet model algorithm achieved 93% accuracy. The minimum- maximum algorithm is implemented over the Microsoft HoloLens spatial map object file. The algorithm computes the individual wall dimensions, and the data has been recorded in column 3. The minimum- maximum algorithm shows an improved accuracy compared to low cost 2D image to facet model algorithm.

The minimum-maximum algorithm shows a 2.4% increased accuracy with a total accuracy of 95.4% which is good for efficient indoor vector mapping. Even though the minimum-maximum algorithm achieved good accuracy, it has a limitation of measuring a room of only area less than 4.25 square meters. So, the spatial understanding algorithm is implemented, and the resulting wall dimensions are recorded in column 4. Since the spatial algorithm uses the inbuilt algorithms, and functions from Microsoft HoloLens software development kit, it achieves higher accuracy than the other two algorithms. The minimum-maximum algorithm shows a 3.6% improved accuracy than low cost 2D image to facet model algorithm, and 1.2% improved accuracy than a minimum-maximum algorithm. Thus, the spatial understanding algorithm achieves the highest 96.6% accuracy compared to the actual wall dimensions.

Table 3. Proposed algorithms individual accuracy, and difference		
Algorithm	Accuracy	Difference
Low cost 2D image to facet model algorithm	93%	-
Minimum-Maximum Algorithm	95.4%	+ 2.4%
Spatial Understanding Algorithm	96.6%	+ 1.2%

The accuracy of all three different algorithms concerning the actual dimensions of all individual walls are shown in Table 3. The difference between the accuracy of the algorithms is also listed in the table. The proposed algorithms show an improvisation from one algorithm to another. The difference between all 3 individual algorithms is not much significant for use in AR applications. Whereas in Indoor network planning, the accuracy of indoor space wall dimension is essential to get the best ray tracing, signal strength, and signal propagation results. So, based on the application, and requirement of accuracy, and availability of hardware any one of the above three algorithms can be used.

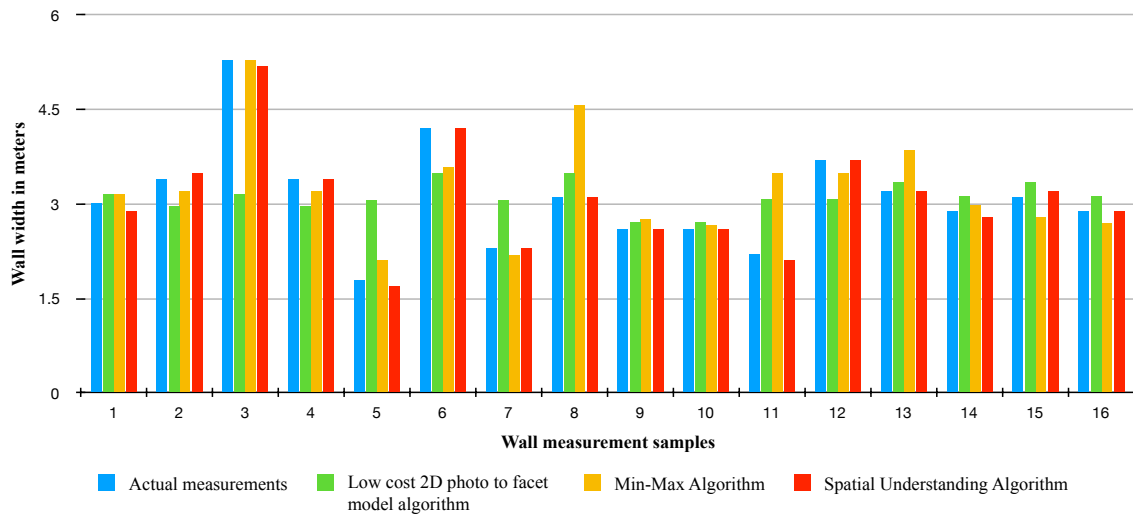
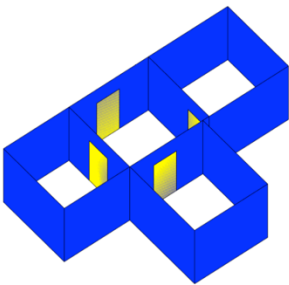
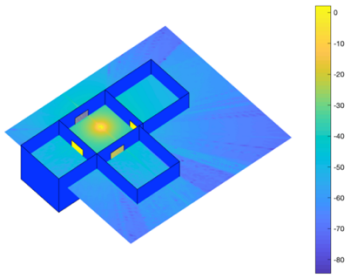
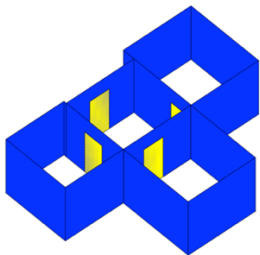
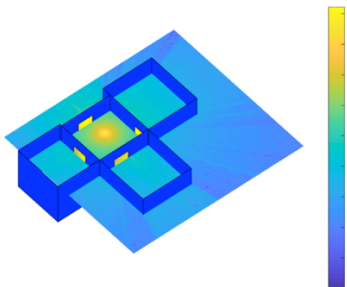
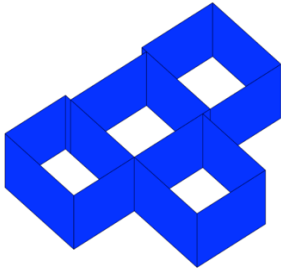
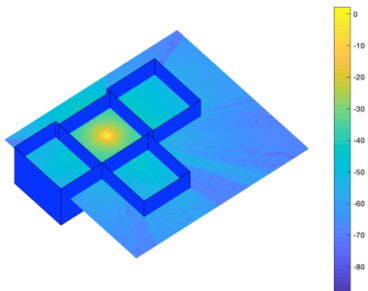


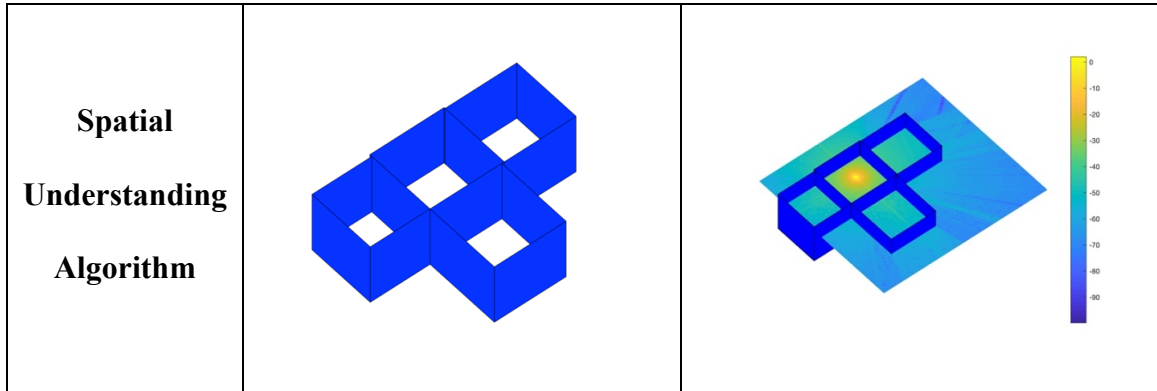
Figure 21. Wall dimension measurement comparison chart.

The graphical representation of wall dimensions of sixteen different wall samples are as shown in the Figure 21. Different colors in the histogram chart represent the actual dimension, and individual wall dimension measurement algorithms. The chart provides the simple comparison understanding of all three algorithms compared to the actual dimensions. Out of 16 walls, all three algorithms work best for 9 walls. The spatial algorithm matches the actual dimension in all 16 wall samples, so it's the most reliable algorithm out of all the three algorithms. The low cost 2D image to facet model algorithm accuracy was dropped in 4 wall dimension measurements with reduced accuracy. The minimum-maximum algorithm accuracy was dropped in only 2 walls with different wall dimension results. But these mis-calculation of dimension will be corrected in the vector map building function, where it compares the individual wall with their parallel walls and matches up with the right dimension.

The ray tracing results, and indoor vector map builds based on all three algorithms are as shown in Figure 22. In column 2, we can observe the 3D indoor vector maps of a 2-bedroom apartment. Observe the changes in the room area or dimensions of the individual algorithm concerning actual dimension vector map. All the three algorithms show a slight difference in individual room area, but the changes are not significant. The changes are less, and don't affect the ray tracing results. Thus, all the three algorithms can be used for indoor network planning. The ray tracing results of all three algorithms are shown in column 3. Thus, using these algorithms, the user can build a 3D indoor environment vector map, and use them to visualize the signal strength for Indoor network planning using ray tracing algorithms.

	Indoor Vector Map	Ray Tracing Results
<b>Actual Dimensions</b>		
<b>Low cost 2D image to facet model algorithm</b>		
<b>Min-Max Algorithm</b>		

**Figure 22. Indoor vector map, and ray tracing results of all three algorithms.**

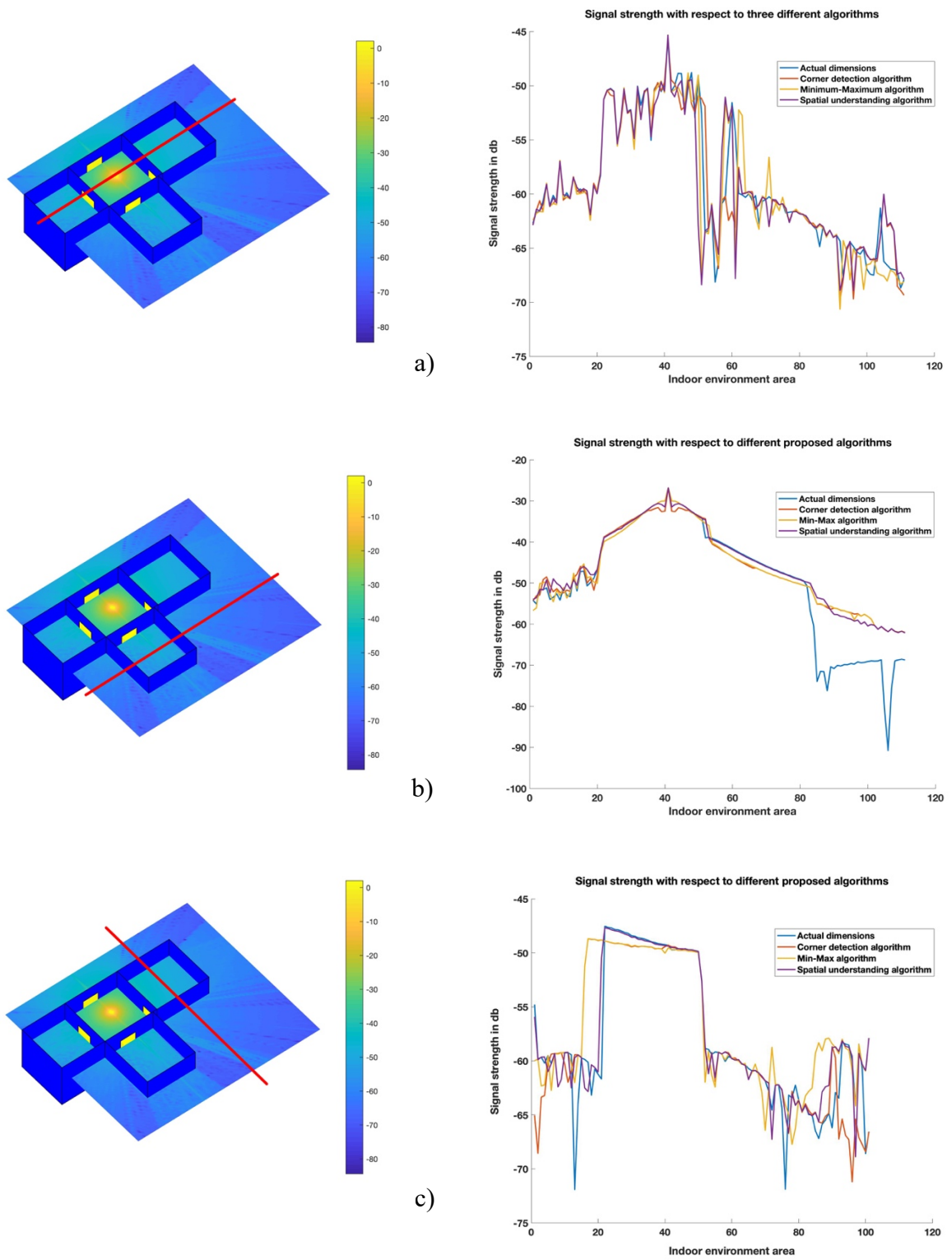


**Figure 22. Continued.**

### **Indoor environment signal strength at different locations.**

Indoor environment signal strength at different regions with respect to all three algorithms are observed. The signal strength of an indoor environment in the region marked in a red line is shown in Figure 23 a). In this case, the region closer to the signal source is selected. The changes in signal strength with respect to the walls, and doors can be observed. Since there is a door parallel to the source in the left-side room the signal loss is less compared to the right-side room where the signal is blocked more. The signal strength of the indoor environment away from the signal source is shown in Figure 23 b). Only one room is considered, so the signal strength reduces once it crosses the walls of the room, which can be observed in Figure 23 b). The signal strength of the indoor environment away from the signal source is shown in Figure 23 c). Only one room is considered, so the signal strength reduces once it crosses the walls of the room. This can also be observed in Figure 23 c). Thus, from the three cases, we can observe, and compare the variations of all three algorithms indoor vector maps. Since there is no much difference in the dimension measurement of all three algorithms, the signal strength variations of the algorithms are less.





**Figure 23. Signal strength of all three algorithms at the region of red line, i.e. a) at the center of source. b) and c), away from the source.**

## VII. CONCLUSION

This research presents a novel image processing algorithm that is capable of creating the facet model of an indoor environment based on images captured by typical smart phone cameras. The algorithm can be considered as a simplified spatial mapping technique that leverages AR (AR) technologies, and principles. The application of the algorithm was focused on ray tracing, and wireless indoor channel prediction. With the evolution of 5G networks, and AR application, it is expected that there will be a great need for integrating network planning, and visualization algorithms with AR technologies. It was found that the proposed solution could be used for standard indoor residential houses, but it is not efficient for large or complex indoor spaces. The proposed solution applies edge, and corner detection algorithms on the images of the walls, and identifies the coordinates, and dimensions of the basic electromagnetic clutter, which are walls, and doors. The coordinate system was based on the facet model that is used by most of the ray tracing, and channel estimation algorithms. It was found that in less than 3 minutes a user could obtain signal strength estimations in a 3-bedroom house just by uploading *.jpg* images of the walls of all rooms.

The proposed solution works best for all medium-sized apartment but if we need to compute a vector map of some large space area like office, school or other large commercial spaces. It is hard to capture the complete large wall space inside a single picture. So, to overcome this limitation, and to reduce the complexity of existing spatial maps, Microsoft HoloLens is used. The spatial map computed from Microsoft HoloLens is sophisticated, and it includes coordinates details of all the furniture, and things present inside the environment, which will not be used in many different AR applications.

The spatial maps mapped through the Microsoft HoloLens are used for the minimum-maximum, and spatial understanding algorithms. These spatial maps are efficient in mapping indoor environment. These maps will contain the spatial data of all the objects inside the mapped space i.e. wall, empty space, furniture's, etc., which is very complex. For efficient indoor network planning, and ray tracing algorithm, we need clear, simple vector map with only the building dimensions data. Thus, we designed 2 other algorithms. In algorithm1, we processed the spatial map vertices of individual walls based on vertices normal and measured the dimensions of a room. But due to the limitation of the depth sensor capability of measuring maximum of 3.1 meter, and not able to access the direct real-time sensor data from the HoloLens. we are only able to map an area less than 4.25 square meter.

In order to overcome this limitation, we have used the inbuilt functionalities available in the HoloLens like plane finding, spatial understanding, and different topology queries. In which the Microsoft HoloLens understands the spatial map, and stores different individual spatial maps into an array. These arrays consist of multiple small spatial maps of an indoor environment, and are arranged in a descending order of spatial area. These spatial maps also consist of their respective width, and breadth of the individual spaces. Since the wall spaces are the biggest spaces compared to other small spaces, table space or & couch space , etc., we can easily consider the first 2 largest spatial maps as the 2 parallel walls of a room, and next 2 largest spatial maps as the other 2 parallel walls of a room. Thus, we got the spatial maps of all the 4 walls, and extracted their dimensions. In the end, the dimensions of a room from the 2 algorithms are used to build a 3D

coordinate system. From which the vector maps are build and used as an input for the ray tracing algorithm to visualize signal, and for indoor network planning. The above 2 algorithms are efficient and can measure the indoor walls dimensions precisely, but they don't have a feature of identifying the doors inside the indoor environment, which is an important factor for ray tracing, and indoor network planning so, a new algorithm or technique needs to be designed for identifying doors from the indoor spatial map data.

## VIII. FUTURE WORK

The proposed solution offers three different algorithms to compute indoor environment vector map. All three individual algorithms achieve the research objective of computing vector map. The low cost 2D image to facet model algorithm creates a novel vector mapping technique using image processing algorithms. The other two algorithms i.e. minimum-maximum algorithm and spatial understanding algorithms converts the existing complex spatial maps into simple vector maps. Though, all three algorithms successfully compute the indoor environment vector map, each algorithm will have their own advantages and disadvantages. The low cost 2D image to facet model algorithm is best suitable for residential apartments and are cost efficient. The minimum-maximum algorithm is best suitable for small commercial spaces. Both low cost 2D image to facet model algorithm and minimum-maximum algorithm can measure an indoor environment of area less than 4.25 square meters. The spatial understanding algorithm works best for all spaces like residential and large commercial spaces irrespective of their dimensions. Both minimum-maximum algorithms require existing AR devices like Microsoft HoloLens to compute the spatial maps and cost more than the low cost 2D image to facet algorithms. Since, all things presents inside the indoor environment like wall, windows, doors, furniture, etc. affects the ray tracing results, the more precise details of these things will increase the accuracy of ray tracing results. In our proposed solution. Since we are using the image processing techniques only in low cost 2D image to facet model algorithms, we are able to identify the doors. Thus, to have an efficient solution from all the three designed algorithms, it is best to integrate the low cost 2D image to facet model algorithm and spatial understanding algorithms. So that the integrated algorithm will

work best in all residential and commercial spaces with door, window and furniture detection. This feature of object identification and all the things present in the environment will increase the ray tracing results and provides a new way of indoor network planning.

## APPENDIX SECTION

### 1. Vector map

```
function varargout = Vector_Map(varargin) % VECTOR_MAP MATLAB code for
Vector_Map.fig
%   VECTOR_MAP, by itself, creates a new VECTOR_MAP or raises the
existing
%   singleton*.
%
%   H = VECTOR_MAP returns the handle to a new VECTOR_MAP or the handle
to
%   the existing singleton*.
%
%   VECTOR_MAP('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in VECTOR_MAP.M with the given input
arguments.
%
%   VECTOR_MAP('Property','Value',...) creates a new VECTOR_MAP or
raises the
%   existing singleton*. Starting from the left, property value pairs
are
%   applied to the GUI before Vector_Map_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to Vector_Map_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Begin initialization code
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @Vector_Map_OpeningFcn, ...
    'gui_OutputFcn', @Vector_Map_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Vector_Map is made visible.
function Vector_Map_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% varargin command line arguments to Vector_Map (see VARARGIN)

% Choose default command line output for Vector_Map
handles.output = hObject;
% Room.Wall=struct('Coordinates',[],'Width',[],'Image',[]);
% handles.Room = Room;
% % Update handles structure
% setappdata(0,'Room',handles.Room);

guidata(hObject, handles);
%
%     textIn = 'Welcome to Indoor building vector mapping';
%     ha = actxserver('SAPI.SpVoice');
%     invoke(ha,'speak',textIn)
% UIWAIT makes Vector_Map wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Vector_Map_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in Height.
function Height_Callback(hObject, eventdata, handles)
% hObject handle to Height (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
%     textIn = 'please enter wall height';
%     ha = actxserver('SAPI.SpVoice');
%     invoke(ha,'speak',textIn)

function HeightEdit_Callback(hObject, eventdata, handles)
% hObject handle to HeightEdit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of HeightEdit as text
%     str2double(get(hObject,'String')) returns contents of HeightEdit
% as a double
str=str2double(get(handles.HeightEdit,'String'));
if isempty(str2double(str))
    str=2.7;
end
setappdata(0,'height',str)

```



```

% --- Executes during object creation, after setting all properties.
function HeightEdit_CreateFcn(hObject, eventdata, handles)
% hObject handle to HeightEdit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in UploadPhotos.
function UploadPhotos_Callback(hObject, eventdata, handles)
% hObject handle to UploadPhotos (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% height=handles.Height1;
% % textIn = 'please upload indoor wall photos';
% % ha = actxserver('SAPI.SpVoice');
% % invoke(ha,'speak',textIn)
height=getappdata(0,'height');
Indoor_Wall_Photo(height);

% --- Executes on button press in RoomPosition.
function RoomPosition_Callback(hObject, eventdata, handles)
% hObject handle to RoomPosition (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'please enter room position';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)

% --- Executes on button press in Left.
function Left_Callback(hObject, eventdata, handles)
% hObject handle to Left (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'left';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','left');

% --- Executes on button press in Top.
function Top_Callback(hObject, eventdata, handles)
% hObject handle to Top (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
setappdata(0,'Room_Position','top');

% --- Executes on button press in Right.
function Right_Callback(hObject, eventdata, handles)
% hObject handle to Right (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB

```

```

% handles structure with handles, and user data (see GUIDATA)
% textIn = 'right';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','right');

% --- Executes on button press in None.
function None_Callback(hObject, eventdata, handles)
% hObject handle to None (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'none';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','none');

% --- Executes on button press in Bottom.
function Bottom_Callback(hObject, eventdata, handles)
% hObject handle to Bottom (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'bottom';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','bottom');

% --- Executes on button press in Front.
function Front_Callback(hObject, eventdata, handles)
% hObject handle to Front (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'front';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','front');

% --- Executes on button press in Back.
function Back_Callback(hObject, eventdata, handles)
% hObject handle to Back (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'back';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
setappdata(0,'Room_Position','back');

% --- Executes on button press in BuildingDetails.
function BuildingDetails_Callback(hObject, eventdata, handles)
% hObject handle to BuildingDetails (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'building details';
% ha = actxserver('SAPI.SpVoice');

```

```

%      invoke(ha,'speak',textIn)
a=getappdata(0,'Room1');
assignin('base','Room',a);
openvar('Room');

% --- Executes on button press in BuildingVectorMap.
function BuildingVectorMap_Callback(hObject, eventdata, handles)
% hObject handle to BuildingVectorMap (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'building vector map';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
Building_Vector_Map(getappdata(0,'Room1'));
view(3);
rotate3d on;

% --- Executes on button press in ThreeDBuilding.
function ThreeDBuilding_Callback(hObject, eventdata, handles)
% hObject handle to ThreeDBuilding (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = '3d view of building';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
% % if isfield(handles,'Room')==0
% Room.Wall=struct('Coordinates',[],'Width',[],'Image',[]);
% handles.Room = Room;
% endr
cla
ThreeD_Building(getappdata(0,'Room1'));
view(3);
rotate3d on;

% --- Executes on mouse press over axes background.
function axes1_ButtonDownFcn(hObject, eventdata, handles)
% hObject handle to axes1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject handle to Reset (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'reset';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
close(gcf);
Vector_Map;

```

## 2. Indoor wall photo

```
function varargout = Indoor_Wall_Photo(varargin)
```

```

% INDOOR_WALL_PHOTO MATLAB code for Indoor_Wall_Photo.fig
% INDOOR_WALL_PHOTO, by itself, creates a new INDOOR_WALL_PHOTO or
raises the existing
% singleton*.
%
% H = INDOOR_WALL_PHOTO returns the handle to a new INDOOR_WALL_PHOTO
or the handle to
% the existing singleton*.
%
% INDOOR_WALL_PHOTO('CALLBACK',hObject,eventData,handles,...) calls
the local
% function named CALLBACK in INDOOR_WALL_PHOTO.M with the given input
arguments.
%
% INDOOR_WALL_PHOTO('Property','Value',...) creates a new
INDOOR_WALL_PHOTO or raises the
% existing singleton*. Starting from the left, property value pairs
are
% applied to the GUI before Indoor_Wall_Photo_OpeningFcn gets called.
An
% unrecognized property name or invalid value makes property
application
% stop. All inputs are passed to Indoor_Wall_Photo_OpeningFcn via
varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Indoor_Wall_Photo

% Last Modified by GUIDE v2.5 11-May-2017 19:43:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @Indoor_Wall_Photo_OpeningFcn, ...
    'gui_OutputFcn', @Indoor_Wall_Photo_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Indoor_Wall_Photo is made visible.
function Indoor_Wall_Photo_OpeningFcn(hObject, eventdata, handles,
varargin)

```

```

% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% varargin command line arguments to Indoor_Wall_Photo (see VARARGIN)

% Choose default command line output for Indoor_Wall_Photo
handles.output = hObject;
    % Room=getappdata(0,'Room'); handles.Room = Room;
guidata(hObject,handles);
% Update handles structure
%cla(handles.axes1);
% axes(handles.axes1);
% hold off;
% cla reset;
guidata(hObject, handles);
if isfield(handles, 'Room')==0
Room.Wall=struct('Coordinates',[], 'Width',[], 'Image',[], 'door',[], 'window',[]);
    handles.Room = Room;
end
close(gcf);
setappdata(0, 'door1', 0);
setappdata(0, 'door2', 0);
setappdata(0, 'door3', 0);
setappdata(0, 'door4', 0);

% UIWAIT makes Indoor_Wall_Photo wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = Indoor_Wall_Photo_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in UploadIndoorWall1.
function UploadIndoorWall1_Callback(hObject, eventdata, handles)
% hObject handle to UploadIndoorWall1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
%     Wall=guidata(gcbo);
%     textIn = 'please upload first wall photo';
%     ha = actxserver('SAPI.SpVoice');
%     invoke(ha,'speak',textIn)
[basefilename,path]= uigetfile({'*.jpg'}, 'Open jpeg Image File');
filename= fullfile(path, basefilename);
I = imread (filename);
size(I)
% Wall_Images=[];
IndoorWall1=I;

```

```

handles.Wall1=IndoorWall1;
guidata(hObject, handles);
imshow(I, 'Parent', handles.axes1);

% --- Executes on button press in UploadIndoorWall2.
function UploadIndoorWall2_Callback(hObject, eventdata, handles)
% hObject handle to UploadIndoorWall2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'please upload second wall photo';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
[basefilename,path]= uigetfile({'*.jpg'}, 'Open jpeg Image File');
filename= fullfile(path, basefilename);
I = imread (filename);
size(I)
IndoorWall2=I;
handles.Wall2=IndoorWall2;
guidata(hObject, handles);
imshow(IndoorWall2, 'Parent', handles.axes2);

% --- Executes on button press in UploadIndoorWall3.
function UploadIndoorWall3_Callback(hObject, eventdata, handles)
% hObject handle to UploadIndoorWall3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'please upload third wall photo';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
[basefilename,path]= uigetfile({'*.jpg'}, 'Open jpeg Image File');
filename= fullfile(path, basefilename);
I = imread (filename);
size(I);
IndoorWall3=I;
handles.Wall3=IndoorWall3;
guidata(hObject, handles);
imshow(IndoorWall3, 'Parent', handles.axes3);

% --- Executes on button press in UploadIndoorWall4.
function UploadIndoorWall4_Callback(hObject, eventdata, handles)
% hObject handle to UploadIndoorWall4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'please upload fourth wall photo';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
[basefilename,path]= uigetfile({'*.jpg'}, 'Open jpeg Image File');
filename= fullfile(path, basefilename);
I = imread (filename);
size(I);
IndoorWall4=I;
handles.Wall4=IndoorWall4;
guidata(hObject, handles);
imshow(IndoorWall4, 'Parent', handles.axes4);

```

```

% --- Executes on button press in VectorMap.
function VectorMap_Callback(hObject, eventdata, handles)
% hObject handle to VectorMap (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'vector map of a room';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
Room=getappdata(0,'Room');
RoomNumber=getappdata(0,'Room_No');
Room_Position=getappdata(0,'Room_Position');
if RoomNumber>1
    Room=getappdata(0,'Room1');
end
Room(RoomNumber).Wall(1).Image=handles.Wall1;
Room(RoomNumber).Wall(2).Image=handles.Wall2;
Room(RoomNumber).Wall(3).Image=handles.Wall3;
Room(RoomNumber).Wall(4).Image=handles.Wall4;
height=getappdata(0,'height');
Room1=Single_Room_Vector_Map(height,RoomNumber,Room_Position,Room,getappdata(0,'door1'),getappdata(0,'door2'),getappdata(0,'door3'),getappdata(0,'door4'));
% handles.Room = Room;
guidata(hObject,handles);
setappdata(0,'Room1',Room1);
view(3);
rotate3d on;

% --- Executes on button press in ThreeDRoomView.
function ThreeDRoomView_Callback(hObject, eventdata, handles)
% hObject handle to ThreeDRoomView (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = '3d view of a room';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
cla reset
IndoorWall1=handles.Wall1;
IndoorWall2=handles.Wall2;
IndoorWall3=handles.Wall3;
IndoorWall4=handles.Wall4;
height=getappdata(0,'height');
Room=getappdata(0,'Room1');
Single_Room_ThreeD(IndoorWall1,IndoorWall2,IndoorWall3,IndoorWall4,height,Room);
rotate3d on;

% --- Executes on button press in home.
function home_Callback(hObject, eventdata, handles)
% hObject handle to home (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'home';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)

```

```

close(gcf);
Vector_Map;

% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject handle to Reset (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'reset';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)
close(gcf);
Indoor_Wall_Photo;

% --- Executes on button press in RoomNumber.
function RoomNumber_Callback(hObject, eventdata, handles)
% hObject handle to RoomNumber (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% textIn = 'please enter room number';
% ha = actxserver('SAPI.SpVoice');
% invoke(ha,'speak',textIn)

function Room_No_Callback(hObject, eventdata, handles)
% hObject handle to Room_No (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of Room_No as text
% str2double(get(hObject,'String')) returns contents of Room_No as a
double
str=str2double(get(handles.Room_No,'String'));
if isempty(str2double(str))
    set(handles.Room_No,'string','0');
    warndlg('Input must be numerical');
end
setappdata(0,'Room_No',str)

% --- Executes during object creation, after setting all properties.
function Room_No_CreateFcn(hObject, eventdata, handles)
% hObject handle to Room_No (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
% See ISPC, and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in door1.
function door1_Callback(hObject, eventdata, handles)
% hObject handle to door1 (see GCBO)

```



```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of door1
door=get(handles.door1,'Value');
setappdata(0,'door1',door);

% --- Executes on button press in door2.
function door2_Callback(hObject, eventdata, handles)
% hObject handle to door2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of door2
door2=get(handles.door2,'Value');
setappdata(0,'door2',door2);

% --- Executes on button press in door3.
function door3_Callback(hObject, eventdata, handles)
% hObject handle to door3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
door3=get(handles.door3,'Value');
setappdata(0,'door3',door3);
% Hint: get(hObject,'Value') returns toggle state of door3

% --- Executes on button press in door4.
function door4_Callback(hObject, eventdata, handles)
% hObject handle to door4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles, and user data (see GUIDATA)
door4=get(handles.door4,'Value');
setappdata(0,'door4',door4);
% Hint: get(hObject,'Value') returns toggle state of door4

```

### 3. Building Vector Map

```

function [ output_args ] = Building_Vector_Map( Room )
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here
hold on;
for i=1:numel(Room)
for j=1:4
    w=Room(i).Wall(j).Coordinates;
    patch(w(:,1),w(:,2),w(:,3),'b');
    axis equal;
    axis tight;
    alpha(0.5);
end
end
end

```

### 4. Corner detection

```

function [ a ] = corner_detection( top_right_corner)
%J=imsharpen(top_right_corner);
J=top_right_corner;

```

```

no_of_corners=0;
% quality_level=0.9;
threshold=0.08;
k=imresize(J,0.5);
G = fspecial('gaussian',[4 4],0.6);
Ig = imadjust(imfilter(k,G,'same'));

while no_of_corners<3

threshold=threshold-0.02; % threshold level for edge detection.
% quality_level=quality_level-0.1; % quality level for corner
detection.
L=edge(Ig,'canny',threshold); % edge detection function.

% edge dilation.
se=strel('disk',3); % small disc formation on edges.
S=imdilate(L,se);

% edge filling.
X=bwareaopen(S,400);
N=bwpropfilt(X,'MajorAxisLength',100);
M=imfill(N,'holes');

% corner detection.
points = detectMinEigenFeatures(M,'FilterSize',115);
no_of_corners=points.Count;

end

```

## 5.Corners Match

```

Function [width,Pixel_Meter,height,rect_x1,rect_x2]=corners_match(top_le
ft_corner_position,top_right_corner_position,no_of_columns,b,no_of_rows
,Wall_Width,height)

x2=top_right_corner_position.Location(1,1)+(0.75*no_of_columns);
width_pixel=sqrt((top_left_corner_position.Location(1,1)-
x2).^2+(top_left_corner_position.Location(1,2)-
top_right_corner_position.Location(1,2)).^2);

actual_height_pixel=2*((no_of_rows/2)-
min(top_left_corner_position.Location(1,2),top_right_corner_position.Lo
cation(1,2)));
width=(height*width_pixel)/(actual_height_pixel);

rect_x2=top_left_corner_position.Location(1,2);
rect_x1=top_left_corner_position.Location(1,1);
rect_x3=width_pixel;
rect_x4=actual_height_pixel;

Pixel_Meter=width_pixel./width;

End

```

## 6. Door detection

```
function [ door_found,x_meter ] = Door_Detection(a,pixel,Length)
%UNTITLED2 Summary of this function goes here
% Detailed explanation goes here

b=rgb2gray(a);
no_of_rows=size(b,1);
no_of_columns=size(b,2);

ROI=(b(0.20*no_of_rows:0.5*no_of_rows,1:no_of_columns));
% imshow(ROI);
I=ROI;
L = imresize(I,0.5);
n = 1;

Idouble = im2double(I);
avg = mean2(Idouble);
sigma = std2(Idouble);
J = imadjust(L,[avg-n*sigma avg+n*sigma],[]);
Iblur1 = imgaussfilt(J,1,'FilterSize',5);

points=detectHarrisFeatures(Iblur1,'MinQuality',0.1,'FilterSize',5);
number_of_corners=points.Count;

for i=1:number_of_corners
    for j=2:number_of_corners

        if (abs(points.Location(i,1)-points.Location(j,1))>400 &&
abs(points.Location(i,1)-points.Location(j,1))<520)

            if (abs(points.Location(i,2)-points.Location(j,2))>0 &&
abs(points.Location(i,2)-points.Location(j,2))<15)

                x1=points.Location(i,1);
                y1=points.Location(i,2);
                x2=points.Location(j,1);
                y2=points.Location(j,2);

                door_found=1;

                x_val=min(x1,x2);
                Pixel_Per_Meter=pixel/Length;
                x_meter=x_val/Pixel_Per_Meter;

                return
            end
        end
    end
end

door_found=0;
x_meter=0;
end
```

## 7. Individual or First room

```
function [ Room ] = First_Room(
length,breadth,height,Room,First_Wall_Direction)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
Room_No=1;

p1=flip(Room(Room_No).Wall(1).Image,1);
p2=flip(Room(Room_No).Wall(2).Image,1);
p3=flip(Room(Room_No).Wall(3).Image,1);
p4=flip(Room(Room_No).Wall(4).Image,1);

if strcmp(First_Wall_Direction,'North')==1
    s1=p1;s2=p2;s3=p3;s4=p4;
elseif strcmp(First_Wall_Direction,'West')==1
    s1=p4;s2=p1;s3=p2;s4=p3;
elseif strcmp(First_Wall_Direction,'South')==1
    s1=p3;s2=p4;s3=p1;s4=p2;
elseif strcmp(First_Wall_Direction,'East')==1
    s1=p2;s2=p3;s3=p4;s4=p1;
else
    disp('Please enter valid direction');
end

% front
surface([0 length;0 length], [breadth breadth;breadth breadth], [0
0;height height], ...
'FaceColor', 'texturemap', 'CData', s1 );

% back
surface([0 length;0 length], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s3) );

% left
surface([0 0; 0 0], [breadth 0;breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s4) );

% right
surface([length length;length length], [0 breadth;0 breadth], [0 0;
height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s2) );

alpha 0.5;
view(3);
axis equal;
axis tight;
end
```

## 8. Next room

```
function [ Room ] = Next_Room(Room,Room_No)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here

i=Room_No;

height=Room(i).Wall(4).height;
% NR_Wall_One=Wall_Class;
% NR_Wall_Two=Wall_Class;
% NR_Wall_Three=Wall_Class;
% NR_Wall_Four=Wall_Class;

[Enhanced_Wall1,s1,Pixel_Meter,height,W1_rect_x1,W1_rect_x2]=Wall_Enhancement(Room(i).Wall(1).Image,0,height);
[Enhanced_Wall2,s2,d,c,W2_rect_x1,W2_rect_x2]=Wall_Enhancement(Room(i).Wall(2).Image,0,height);
[Enhanced_Wall3,s3,e,f,W3_rect_x1,W3_rect_x2]=Wall_Enhancement(Room(i).Wall(3).Image,0,height);
[Enhanced_Wall4,s4,g,h,W4_rect_x1,W4_rect_x2]=Wall_Enhancement(Room(i).Wall(4).Image,0,height);

[Room]=Wall_Measurement(Enhanced_Wall1,Enhanced_Wall2,Enhanced_Wall3,Enhanced_Wall4,s1,s2,s3,s4,Pixel_Meter,height,Room_No,Room);

wall_width_pixel=Room(i).Wall(1).Width*Pixel_Meter;
rect=[W1_rect_x1,W1_rect_x2,wall_width_pixel,height*Pixel_Meter];
Room(i).Wall(1).Image=imcrop(s1,rect);

wall_width_pixel=Room(i).Wall(2).Width*Pixel_Meter;
rect=[W2_rect_x1,W2_rect_x2,wall_width_pixel,height*Pixel_Meter];
Room(i).Wall(2).Image=imcrop(s2,rect);

wall_width_pixel=Room(i).Wall(3).Width*Pixel_Meter;
rect=[W3_rect_x1,W3_rect_x2,wall_width_pixel,height*Pixel_Meter];
Room(i).Wall(3).Image=imcrop(s3,rect);

wall_width_pixel=Room(i).Wall(4).Width*Pixel_Meter;
rect=[W4_rect_x1,W4_rect_x2,wall_width_pixel,height*Pixel_Meter];
Room(i).Wall(4).Image=imcrop(s4,rect);

% Useful link for 3d cube -
https://www.mathworks.com/help/matlab/ref/primitivesurface-properties.html

end
```

## 9. Plain 3D

```
function [ ] = Plain_ThreeD(
Room,H_Length,H_Breadth,length,breadth,height,NR_Length,NR_Breadth,Hall
_Position )
%UNTITLED5 Summary of this function goes here
% Detailed explanation goes here

FaceColor= 'y';
if strcmp(Hall_Position,'left')==1

%NR_front
surface([-NR_Length -(H_Length+NR_Length);-NR_Length -
(H_Length+NR_Length)], [H_Breadth H_Breadth;H_Breadth H_Breadth], [0
0;height height], ...
'FaceColor', 'texturemap' );

%NR_back
surface([-NR_Length -(H_Length+NR_Length);-NR_Length -
(H_Length+NR_Length)], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap');

%NR_left
surface([-NR_Length -NR_Length;-NR_Length -NR_Length], [0 H_Breadth;0
H_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

%NR_right
surface([- (H_Length+NR_Length) -(H_Length+NR_Length);-
(H_Length+NR_Length) -(H_Length+NR_Length)], [0 H_Breadth;0 H_Breadth],
[0 0; height height], ...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'right')==1
%IF ROOM IS IN LEFT
surface([0 H_Length;0 H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap' );

% back
surface([0 H_Length;0 H_Length], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [0 0; height height],
...
'FaceColor', 'texturemap' );
% right
surface([H_Length H_Length;H_Length H_Length], [0 H_Breadth;0
H_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );
```

```

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position, 'top')==1

% font
surface([0 -H_Length;0 -H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [height height; height+height height+height], ...
'FaceColor', 'texturemap');

% back
surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap');

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 H_Breadth;0
H_Breadth], [height height; height+height height+height], ...
'FaceColor', 'texturemap');
alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position, 'bottom')==1

% font
surface([0 -H_Length;0 -H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [0 0;-height -height], ...
'FaceColor', 'texturemap');

% back
surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [0 0; -height -height],
...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [0 0; -height -height],
...
'FaceColor', 'texturemap');

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 H_Breadth;0
H_Breadth], [0 0; -height -height], ...
'FaceColor', 'texturemap');
alpha 0.5;
view(3);
axis equal;
axis tight;

```

```

elseif strcmp(Hall_Position,'back')==1

surface([0 -H_Length;0 -H_Length], [-H_Breadth -H_Breadth;-H_Breadth -
H_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap' );

% back
surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap' );

% left
surface([0 0; 0 0], [-H_Breadth 0;-H_Breadth 0], [0 0; height height],
...
'FaceColor', 'texturemap' );

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 -H_Breadth;0 -
H_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'front')==1

surface([0 -H_Length;0 -H_Length], [H_Breadth+NR_Breadth
H_Breadth+NR_Breadth;H_Breadth+NR_Breadth H_Breadth+NR_Breadth], [0
0;height height], ...
'FaceColor', 'texturemap' );

% back
surface([0 -H_Length;0 -H_Length], [NR_Breadth NR_Breadth; NR_Breadth
NR_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

% left
surface([0 0; 0 0], [H_Breadth+NR_Breadth
NR_Breadth;H_Breadth+NR_Breadth NR_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [NR_Breadth
H_Breadth+NR_Breadth;NR_Breadth H_Breadth+NR_Breadth], [0 0; height
height], ...
'FaceColor', 'texturemap' );
alpha 0.5;
view(3);
axis equal;
axis tight;

else
disp('please enter proper input for Next_Room_Position');
end
end

```



## 10. Plain 3D vector map

```
function [ output_args ] =
Plain_ThreeD_Vector_Map(length,breadth,height,Room,First_Wall_Direction
)
% UNTITLED Summary of this function goes here
% Detailed explanation goes here

% front
surface([0 length;0 length], [breadth breadth;breadth breadth], [0
0;height height], ...
'FaceColor', 'texturemap' );

% back
surface([0 length;0 length], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [breadth 0;breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap');

% right
surface([length length;length length], [0 breadth;0 breadth], [0 0;
height height], ...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

if strcmp(Next_Room_Position,'right')==1
%IF ROOM IS IN RIGHT

%NR_front
surface([length length+NR_Length;length length+NR_Length], [NR_Breadth
NR_Breadth;NR_Breadth NR_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap' );

%NR_back
surface([length length+NR_Length;length length+NR_Length], [0 0; 0 0],
[0 0; height height], ...
'FaceColor', 'texturemap' );

%NR_left
surface([length length;length length], [0 NR_Breadth;0 NR_Breadth], [0
0; height height], ...
'FaceColor', 'texturemap' );

%NR_right
surface([length+NR_Length length+NR_Length;length+NR_Length
length+NR_Length], [0 NR_Breadth;0 NR_Breadth], [0 0; height height],
...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
```

```

axis equal;
axis tight;

elseif strcmp(Next_Room_Position, 'left')==1
%IF ROOM IS IN LEFT

surface([0 -NR_Length;0 -NR_Length], [NR_Breadth NR_Breadth;NR_Breadth
NR_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap' );

% back
surface([0 -NR_Length;0 -NR_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap' );

% left
surface([0 0; 0 0], [NR_Breadth 0;NR_Breadth 0], [0 0; height height],
...
'FaceColor', 'texturemap' );
% right
surface([-NR_Length -NR_Length;-NR_Length -NR_Length], [0 NR_Breadth;0
NR_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position, 'top')==1
%IF ROOM IS IN TOP

% font
surface([0 NR_Length;0 NR_Length], [breadth breadth;breadth breadth],
[height height;height+height height+height], ...
'FaceColor', 'texturemap' );

% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap' );

% left
surface([0 0; 0 0], [breadth 0;breadth 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap' );
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 breadth;0
breadth], [height height; height+height height+height], ...
'FaceColor', 'texturemap' );
alpha 0.5;
view(3);
axis equal;
axis tight;

```

```

elseif strcmp(Next_Room_Position,'bottom')==1
%IF ROOM IS IN BOTTOM

% front
surface([0 NR_Length;0 NR_Length], [breadth breadth;breadth breadth],
[0 0;-height -height], ...
'FaceColor', 'texturemap', 'CData' );

% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; -height -height],
...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [breadth 0;breadth 0], [0 0; -height -height], ...
'FaceColor', 'texturemap');
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 breadth;0
breadth], [0 0; -height -height], ...
'FaceColor', 'texturemap' );
alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'back')==1
%IF ROOM IS IN front

surface([0 NR_Length;0 NR_Length], [-breadth -breadth;-breadth -
breadth], [0 0;height height], ...
'FaceColor', 'texturemap');

% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap');

% left
surface([0 0; 0 0], [-breadth 0;-breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap');

% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 -breadth;0 -
breadth], [0 0; height height], ...
'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'front')==1
%IF ROOM IS IN front

```

```

surface([0 NR_Length;0 NR_Length], [breadth+NR_Breadth
breadth+NR_Breadth;breadth+NR_Breadth breadth+NR_Breadth], [0 0;height
height], ...
    'FaceColor', 'texturemap' );

% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; height height],
...
    'FaceColor', 'texturemap' );

% left
surface([0 0; 0 0], [breadth+NR_Breadth 0;breadth+NR_Breadth 0], [0 0;
height height], ...
    'FaceColor', 'texturemap' );

% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0
breadth+NR_Breadth;0 breadth+NR_Breadth], [0 0; height height], ...
    'FaceColor', 'texturemap' );

alpha 0.5;
view(3);
axis equal;
axis tight;

else

disp('please enter proper input for Next_Room_Position');

end

Plain_ThreeD( Room,length,breadth,height )
view(3);
axis equal;
axis tight;

end

```

## 11. Single room 3d

```
function [ output_args ] = Single_Room_ThreeD(
wall1,wall2,wall3,wall4,height,Room )
%UNTITLED7 Summary of this function goes here
% Detailed explanation goes here

% height=2.27; %height in meters
Room_No=Room(1).Wall(1).RoomNumber;

Room(Room_No).Wall(1).Image=wall1; % wall 1
Room(Room_No).Wall(2).Image=wall2; % wall 2
Room(Room_No).Wall(3).Image=wall3; % wall 3
Room(Room_No).Wall(4).Image=wall4; % wall 4

i=1;
for j=1:4
Room(i).Wall(j).height=height;
end

s1=flip(wall1,1);
s2=flip(wall2,1);
s3=flip(wall3,1);
s4=flip(wall4,1);

length=Room(Room_No).Wall(1).Width;
breadth=Room(Room_No).Wall(2).Width;

% font
surface([0 length;0 length], [breadth breadth;breadth breadth], [0
0;height height], ...
'FaceColor', 'texturemap', 'CData', s1 );

% back
surface([0 length;0 length], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', s3 );

% left
surface([0 0; 0 0], [breadth 0;breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', s2 );

% right
surface([length length;length length], [0 breadth;0 breadth], [0 0;
height height], ...
'FaceColor', 'texturemap', 'CData', s4 );

alpha 0.5;
view(3);
axis off;
axis tight;
axis equal;
end
```

## 12. Single room 3D vector map

```
function [ Room1 ] =  
Single_Room_Vector_Map(height,RoomNumber,Room_Position,Room,door1,door2  
,door3,door4 )  
  
i=RoomNumber;  
Room_No=RoomNumber;  
Room(i).Wall(1).RoomNumber=RoomNumber;  
  
for j=2:4  
    Room(i).Wall(j).RoomNumber=RoomNumber;  
end  
  
Room(i).Wall(1).Room_Position=Room_Position;  
for j=2:4  
    Room(i).Wall(j).Room_Position=Room_Position;  
end  
  
for j=1:4  
    Room(i).Wall(j).height=height;  
end  
  
[Room]=Next_Room(Room,RoomNumber);  
if door1==1  
    [door_found,x_meter]=Door_Detection(Room(i).Wall(1).Image,Room(i).Wall(  
1).Pixel,Room(i).Wall(1).Width);  
  
    if door_found==1  
        d=[0 x_meter 0; 0 x_meter 2;0 x_meter+0.9 2; 0 x_meter+0.9 0;0 0  
0;0 0 0;0 0 0;0 0 0];  
        Room(i).Wall(1).Coordinates=[Room(i).Wall(1).Coordinates;d];  
        Room(Room_No).Wall(1).door=d;  
  
    elseif door_found==0  
        d=[0 1.1 0; 0 1.1 2;0 2 2; 0 2 0;0 0 0;0 0 0;0 0 0;0 0 0];  
        Room(i).Wall(1).Coordinates=[Room(i).Wall(1).Coordinates;d];  
  
        Room(Room_No).Wall(1).door=d;  
    end  
end  
  
if door2==1  
    [door_found,x_meter]=Door_Detection(Room(i).Wall(2).Image,Room(i).Wall(  
2).Pixel,Room(i).Wall(2).Width);  
    z=Room(Room_No).Wall(2).Width;  
    if door_found==1  
        d=[x_meter+0.9 z 0;x_meter+0.9 z 2; x_meter z 2;x_meter z 0];  
        Room(Room_No).Wall(2).door=d;  
        Room(i).Wall(2).Coordinates=[Room(i).Wall(2).Coordinates;d];  
  
    elseif door_found==0  
        d=[2 z 0;2 z 2; 1.1 z 2;1.1 z 0]  
        Room(Room_No).Wall(2).door=d;  
        Room(i).Wall(2).Coordinates=[Room(i).Wall(2).Coordinates;d];  
    end  
end  
end
```

```

if door3==1
[door_found,x_meter]=Door_Detection(Room(i).Wall(3).Image,Room(i).Wall(
3).Pixel,Room(i).Wall(3).Width);
    x=Room(Room_No).Wall(3).Width;
    if door_found==1
        d=[x x_meter 0; x x_meter 2;x x_meter+0.9 2; x x_meter+0.9 0];
        Room(i).Wall(3).Coordinates=[Room(i).Wall(3).Coordinates;d];
        Room(Room_No).Wall(3).door=d;

    elseif door_found==0

        d=[x 1.1 0; x 1.1 2;x 2 2; x 2 0];

        Room(i).Wall(3).Coordinates=[Room(i).Wall(3).Coordinates;d];

        Room(Room_No).Wall(3).door=d;
    end
end

if door4==1
[door_found,x_meter]=Door_Detection(Room(i).Wall(4).Image,Room(i).Wall(
4).Pixel,Room(i).Wall(4).Width);

    if door_found==1
        d=[x_meter+0.9 0 0;x_meter+0.9 0 2;x_meter 0 2;x_meter 0 0];
        Room(i).Wall(4).Coordinates=[Room(i).Wall(4).Coordinates;d];
        Room(Room_No).Wall(4).door=d;

    elseif door_found==0

        d=[2 0 0;2 0 2;1.1 0 2;1.1 0 0];

        Room(i).Wall(4).Coordinates=[Room(i).Wall(4).Coordinates;d];
        Room(Room_No).Wall(4).door=d;
    end
else
end

if RoomNumber==1
    Room1=Room;
else
    Room1=Patch(Room,RoomNumber,door1,door2,door3,door4);
end

i=RoomNumber;
for j=1:4
w=Room(i).Wall(j).Coordinates;
patch(w(:,1),w(:,2),w(:,3),'b');
end

axis off;
axis equal;
axis tight;
alpha(0.5);

end

```

### 13. 3D Building

```
function [ Room ] = Threed_Building( Room )
%UNTITLED5 Summary of this function goes here
% Detailed explanation goes here

First_Wall_Direction='East'; % For reference initial wall direction is
set to East.
height=Room(1).Wall(1).height;

for Room_No=1:numel(Room)
length=Room(Room_No).Wall(1).Width;
breadth=Room(Room_No).Wall(2).Width;
Room=Threed_Vector_Map(length,breadth,height,Room,First_Wall_Direction,
Room_No);

alpha(0.5);
axis off;
hold on;
end

end
```

### 14. 3D vector map

```
function [Room] =
Threed_Vector_Map(length,breadth,height,Room,First_Wall_Direction,Room_
No)
%UNTITLED Summary of this function goes here
% Detailed explanation goes her
i=Room_No-1;
next_room=Room_No;

if Room_No==1
i=Room_No; %i=1 is room number

p1=flip(Room(Room_No).Wall(1).Image,1);
p2=flip(Room(Room_No).Wall(2).Image,1);
p3=flip(Room(Room_No).Wall(3).Image,1);
p4=flip(Room(Room_No).Wall(4).Image,1);

if strcmp(First_Wall_Direction,'North')==1
s1=p1;s2=p2;s3=p3;s4=p4;
elseif strcmp(First_Wall_Direction,'West')==1
s1=p4;s2=p1;s3=p2;s4=p3;
elseif strcmp(First_Wall_Direction,'South')==1
s1=p3;s2=p4;s3=p1;s4=p2;
elseif strcmp(First_Wall_Direction,'East')==1
s1=p2;s2=p3;s3=p4;s4=p1;
else
disp('Please enter valid direction');
end
hold on;

length=(Room(1).Wall(1).Width+Room(1).Wall(3).Width)/2;
breadth=(Room(1).Wall(2).Width+Room(1).Wall(4).Width)/2;

% front
```



```

surf([0 length;0 length], [breadth breadth;breadth breadth], [0
0;height height], ...
'FaceColor', 'texturemap', 'CData', s1 );
% % back
surf([0 length;0 length], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s3) );
% left
surf([0 0; 0 0], [breadth 0;breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s4) );
% right
surf([length length;length length], [0 breadth;0 breadth], [0 0; height
height], ...
'FaceColor', 'texturemap', 'CData', fliplr(s2) );
%alpha 0.5;
view(3);
axis equal;
axis tight;
hold on;

elseif next_room<=numel(Room)

q1=flip(Room(next_room).Wall(1).Image,1);
q2=flip(Room(next_room).Wall(2).Image,1);
q3=flip(Room(next_room).Wall(3).Image,1);
q4=flip(Room(next_room).Wall(4).Image,1);

length=(Room(i).Wall(1).Width+Room(i).Wall(3).Width)/2;
breadth=(Room(i).Wall(2).Width+Room(i).Wall(4).Width)/2;

NR_Length=(Room(next_room).Wall(1).Width+Room(next_room).Wall(3).Width)
/2;
NR_Breadth=(Room(next_room).Wall(2).Width+Room(next_room).Wall(4).Width
)/2;

Next_Room_Position=Room(next_room).Wall(1).Room_Position;

if strcmp(Next_Room_Position,'right')==1 %If room position is right.

r1=q2;r2=q3;r3=q4;r4=q1;
Room(next_room).Wall(2).Reference_Wall=Room(i).Wall(1);

%NR_front
surface([length length+NR_Length;length length+NR_Length], [NR_Breadth
NR_Breadth;NR_Breadth NR_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap', 'CData', r1 );
%NR_back
surface([length length+NR_Length;length length+NR_Length], [0 0; 0 0],
[0 0; height height], ...
'FaceColor', 'texturemap', 'CData', r3 );
%NR_left
surface([length length;length length], [0 NR_Breadth;0 NR_Breadth], [0
0; height height], ...
'FaceColor', 'texturemap', 'CData', r2 );
%NR_right
surface([length+NR_Length length+NR_Length;length+NR_Length
length+NR_Length], [0 NR_Breadth;0 NR_Breadth], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', r4 );

```

```

%alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'left')==1 %If room position is left.

r1=q4;r2=q1;r3=q2;r4=q3;
Room(next_room).Wall(2).Reference_Wall=[1 3];

%front
surface([0 -NR_Length;0 -NR_Length], [NR_Breadth NR_Breadth;NR_Breadth
NR_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(r1) );
% back
surface([0 -NR_Length;0 -NR_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', r3 );
% left
surface([0 0; 0 0], [NR_Breadth 0;NR_Breadth 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData',fliplr(r2) );
%right
surface([-NR_Length -NR_Length;-NR_Length -NR_Length], [0 NR_Breadth;0
NR_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(r4) );
%alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'top')==1 %If room position is top.

r1=q1;r2=q2;r3=q3;r4=q4;

% font
surface([0 NR_Length;0 NR_Length], [breadth breadth;breadth breadth],
[height height;height+height height+height], ...
'FaceColor', 'texturemap', 'CData', r1 );
% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap', 'CData', r3 );
% left
surface([0 0; 0 0], [breadth 0;breadth 0], [height height;
height+height height+height], ...
'FaceColor', 'texturemap', 'CData', r2 );
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 breadth;0
breadth], [height height; height+height height+height], ...
'FaceColor', 'texturemap', 'CData', r4 );
%alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'bottom')==1 %If room position is
bottom.

```

```

r1=q1;r2=q2;r3=q3;r4=q4;

% front
surface([0 NR_Length;0 NR_Length], [breadth breadth;breadth breadth],
[0 0;-height -height], ...
'FaceColor', 'texturemap', 'CData', r1 );
% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; -height -height],
...
'FaceColor', 'texturemap', 'CData', r3 );
% left
surface([0 0; 0 0], [breadth 0;breadth 0], [0 0; -height -height], ...
'FaceColor', 'texturemap', 'CData', r2 );
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 breadth;0
breadth], [0 0; -height -height], ...
'FaceColor', 'texturemap', 'CData', r4 );
%alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'front')==1 %If room position is
front.

r1=q3;r2=q2;r3=q1;r4=q4;
Room(i+1).Wall(2).Reference_Wall=Room(i).Wall(3);

%front
surface([0 NR_Length;0 NR_Length], [-breadth -breadth;-breadth -
breadth], [0 0;height height], ...
'FaceColor', 'texturemap', 'CData', r1 );
% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', r3 );
% left
surface([0 0; 0 0], [-breadth 0;-breadth 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', r2 );
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0 -breadth;0 -
breadth], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', r4 );
%alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Next_Room_Position,'back')==1 %If room position is back.

r1=q3;r2=q2;r3=q1;r4=q4;
Room(i+1).Wall(2).Reference_Wall=Room(i).Wall(1);

% front
surface([0 NR_Length;0 NR_Length], [breadth+NR_Breadth
breadth+NR_Breadth;breadth+NR_Breadth breadth+NR_Breadth], [0 0;height
height], ...
'FaceColor', 'texturemap', 'CData', r1 );

```

```

% back
surface([0 NR_Length;0 NR_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', r3 );
% left
surface([0 0; 0 0], [breadth+NR_Breadth 0;breadth+NR_Breadth 0], [0 0;
height height], ...
'FaceColor', 'texturemap', 'CData', r2 );
% right
surface([NR_Length NR_Length;NR_Length NR_Length], [0
breadth+NR_Breadth;0 breadth+NR_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', r4 );
%alpha 0.5;
%view(3);
axis equal;
axis tight;

else
disp('please enter proper input for Next_Room_Position');
end
else
end
end

```

## 15. 3D view

```
function [Room] = ThreeD(
Room,H_Length,H_Breadth,length,breadth,height,NR_Length,NR_Breadth,Hall
_Position )
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
i=Room(2).Wall(1).Room_No;
t1=flip(Room(3).Wall(1).Image,1);
t2=flip(Room(3).Wall(2).Image,1);
t3=flip(Room(3).Wall(3).Image,1);
t4=flip(Room(3).Wall(4).Image,1);

if strcmp(Hall_Position,'left')==1

%IF ROOM IS IN RIGHT
h1=t4;h2=t1;h3=t2;h4=t3;
Room(i+1).Wall(2).Reference_Wall=Room(i).Wall(3);
%NR_front
surface([-NR_Length -(H_Length+NR_Length);-NR_Length -
(H_Length+NR_Length)], [H_Breadth H_Breadth;H_Breadth H_Breadth], [0
0;height height], ...
'FaceColor', 'texturemap', 'CData', h1 );

%NR_back
surface([-NR_Length -(H_Length+NR_Length);-NR_Length -
(H_Length+NR_Length)], [0 0; 0 0], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', h3 );

%NR_left
surface([-NR_Length -NR_Length;-NR_Length -NR_Length], [0 H_Breadth;0
H_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', h2 );

%NR_right
surface([- (H_Length+NR_Length) -(H_Length+NR_Length);-
(H_Length+NR_Length) -(H_Length+NR_Length)], [0 H_Breadth;0 H_Breadth],
[0 0; height height], ...
'FaceColor', 'texturemap', 'CData', h4 );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'right')==1
%IF ROOM IS IN LEFT
h1=t2;h2=t3;h3=t4;h4=t1;
Room(i+1).Wall(2).Reference_Wall=Room(i).Wall(1);
```

```

surface([0 H_Length;0 H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [0 0;height height], ...
    'FaceColor', 'texturemap', 'CData', h1 );

% back
surface([0 H_Length;0 H_Length], [0 0; 0 0], [0 0; height height], ...
    'FaceColor', 'texturemap', 'CData', h3 );

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [0 0; height height],
...
    'FaceColor', 'texturemap', 'CData', h2 );
% right
surface([H_Length H_Length;H_Length H_Length], [0 H_Breadth;0
H_Breadth], [0 0; height height], ...
    'FaceColor', 'texturemap', 'CData', h4 );

alpha 0.5;
view(3);
axis equal;
axis tight;
elseif strcmp(Hall_Position,'top')==1
%IF ROOM IS IN TOP
h1=t3;h2=t2;h3=t1;h4=t4;
% font
surface([0 -H_Length;0 -H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [height height;height+height height+height], ...
    'FaceColor', 'texturemap', 'CData', h1 );

% back
surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [height height;
height+height height+height], ...
    'FaceColor', 'texturemap', 'CData', h3 );

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [height height;
height+height height+height], ...
    'FaceColor', 'texturemap', 'CData', h2 );
% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 H_Breadth;0
H_Breadth], [height height; height+height height+height], ...
    'FaceColor', 'texturemap', 'CData', h4 );
alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'bottom')==1
%IF ROOM IS IN BOTTOM
h1=t1;h2=t2;h3=t3;h4=t4;
% font
surface([0 -H_Length;0 -H_Length], [H_Breadth H_Breadth;H_Breadth
H_Breadth], [0 0;-height -height], ...
    'FaceColor', 'texturemap', 'CData', h1 );

% back

```

```

surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [0 0; -height -height],
...
'FaceColor', 'texturemap', 'CData', h3 );

% left
surface([0 0; 0 0], [H_Breadth 0;H_Breadth 0], [0 0; -height -height],
...
'FaceColor', 'texturemap', 'CData', h2 );
% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 H_Breadth;0
H_Breadth], [0 0; -height -height], ...
'FaceColor', 'texturemap', 'CData', h4 );
alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'back')==1
%IF ROOM IS IN front
h1=t3;h2=t2;h3=t1;h4=t4;
Room(i+1).Wall(2).Reference_Wall=[1 2];
surface([0 -H_Length;0 -H_Length], [-H_Breadth -H_Breadth;-H_Breadth -
H_Breadth], [0 0;height height], ...
'FaceColor', 'texturemap', 'CData', h1 );

% back
surface([0 -H_Length;0 -H_Length], [0 0; 0 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', fliplr(h3) );

% left
surface([0 0; 0 0], [-H_Breadth 0;-H_Breadth 0], [0 0; height height],
...
'FaceColor', 'texturemap', 'CData', fliplr(h2) );

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [0 -H_Breadth;0 -
H_Breadth], [0 0; height height], ...
'FaceColor', 'texturemap', 'CData', fliplr(h4) );

alpha 0.5;
view(3);
axis equal;
axis tight;

elseif strcmp(Hall_Position,'front')==1
%IF ROOM IS IN front
h1=t3;h2=t2;h3=t1;h4=t4;
Room(i+1).Wall(2).Reference_Wall=Room(i).Wall(4);
surface([0 -H_Length;0 -H_Length], [H_Breadth+NR_Breadth
H_Breadth+NR_Breadth;H_Breadth+NR_Breadth H_Breadth+NR_Breadth], [0
0;height height], ...
'FaceColor', 'texturemap', 'CData', h1 );

% back

```

```

surface([0 -H_Length;0 -H_Length], [NR_Breadth NR_Breadth; NR_Breadth
NR_Breadth], [0 0; height height], ...
    'FaceColor', 'texturemap', 'CData', h3 );

% left
surface([0 0; 0 0], [H_Breadth+NR_Breadth
NR_Breadth;H_Breadth+NR_Breadth NR_Breadth], [0 0; height height], ...
    'FaceColor', 'texturemap', 'CData', h2 );

% right
surface([-H_Length -H_Length;-H_Length -H_Length], [NR_Breadth
H_Breadth+NR_Breadth;NR_Breadth H_Breadth+NR_Breadth], [0 0; height
height], ...
    'FaceColor', 'texturemap', 'CData', h4 );
alpha 0.5;
view(3);
axis equal;
axis tight;

else
disp('please enter proper input for Next_Room_Position');
end
end

```



## 16. wall enhancement

```
function [test,b,Pixel_Meter,height,rect_x1,rect_x2] =  
Wall_Enhancement( Wall,Wall_Width,height)  
%UNTITLED6 Summary of this function goes here  
% Detailed explanation goes her  
% b=imread('s1.jpg');  
b=Wall;  
% figure  
                % imshow(b);  
                a=rgb2gray(b);  
  
% figure  
% imshow(a);  
  
no_of_rows=size(a,1); % to find number of rows in a image matrix.  
%disp(0.5*no_of_rows);  
  
no_of_columns=length(a); % to find number of columns in a mimage  
matrix.  
%disp(0.5*no_of_columns);  
  
% figure  
top_left_corner_image=a(1:0.25*no_of_rows,1:0.25*no_of_columns); % to  
crop only top lIMG_20170113_113504left corner from image.  
% imshow(top_left_corner_image);  
  
% figure  
top_right_corner_image=a(1:0.25*no_of_rows,0.75*no_of_columns:no_of_col  
umns); % to crop only right corner from image.  
% imshow(top_right_corner_image);  
  
top_left_corner_position=corner_detection(top_left_corner_image);  
top_right_corner_position=corner_detection(top_right_corner_image);  
  
[test,Pixel_Meter,height,rect_x1,rect_x2]=corners_match(top_left_corner  
_position,top_right_corner_position,no_of_columns,b,no_of_rows,Wall_Wid  
th,height);  
% disp('width');  
% disp(test);  
  
end
```

## 17. wall measurement

```
function [Room,length,breadth,height] = Wall_Measurement(
Enhanced_Wall1,Enhanced_Wall2,Enhanced_Wall3,Enhanced_Wall4,s1,s2,s3,s4
,Pixel_Meter,height,Room_No,Room)
%UNTITLED4 Summary of this function goes here
% Detailed explanation goes here

if Enhanced_Wall1>Enhanced_Wall3
    length=Enhanced_Wall1;
else
    length=Enhanced_Wall3;
end

i=Room_No;
Room(i).Wall(1).Width=length;
Room(i).Wall(1).Pixel=length*Pixel_Meter;
Room(i).Wall(3).Width=length;
Room(i).Wall(3).Pixel=length*Pixel_Meter;

if Enhanced_Wall2>Enhanced_Wall4
    breadth=Enhanced_Wall2;
else
    breadth=Enhanced_Wall4;
end

Room(i).Wall(2).Width=breadth;
Room(i).Wall(2).Pixel=breadth*Pixel_Meter;
Room(i).Wall(4).Width=breadth;
Room(i).Wall(4).Pixel=breadth*Pixel_Meter;

disp('Width of wall 2 & wall 4 in meters:');
disp(breadth);

% coordinates are in x,z,y format
Room(i).Wall(1).Coordinates(1:4,1:3)=[0 0 0;0 0 height ;0 breadth
height ; 0 breadth 0 ];

Room(i).Wall(2).Coordinates(1:4,1:3)=[ 0 breadth 0;0 breadth height;
length breadth height ;length breadth 0];

Room(i).Wall(3).Coordinates(1:4,1:3)=[length breadth 0; length breadth
height ; length 0 height ;length 0 0 ];

Room(i).Wall(4).Coordinates(1:4,1:3)=[length 0 0; length 0 height ;0 0
height ;0 0 0];

end
```

## 18. Spatial mapping

```
x1=min(VRoom1.Object1);
x2=max(VRoom1.Object1);
y1=max(VRoom1.VarName3);
y2=min(VRoom1.VarName3);
z1=min(VRoom1.VarName4);
z2=max(VRoom1.VarName4);

l1=sqrt(x1^2+z1^2);
l2=sqrt(x2.^2+z2.^2);

b1=sqrt(x1.^2+z2.^2);
b2=sqrt(x2.^2+z1.^2);

disp('Length');
length=(abs(l1)+abs(l2))/2;
disp(length);

disp('height');
height=abs(y1)+abs(y2);
disp(height);

disp('breadth');
breadth=(abs(b1)+abs(b2))/2;
disp(breadth);

Room(1).Wall(1).Coordinates(1:4,1:3)=[0 0 0;0 0 height ;0 breadth
height ; 0 breadth 0 ];
Room(1).Wall(2).Coordinates(1:4,1:3)=[ 0 breadth 0;0 breadth height;
length breadth height ;length breadth 0];
Room(1).Wall(3).Coordinates(1:4,1:3)=[length breadth 0; length breadth
height ; length 0 height ;length 0 0 ];
Room(1).Wall(4).Coordinates(1:4,1:3)=[length 0 0; length 0 height ;0 0
height ;0 0 0];

figure;
surf(X,Y,Z,ReceivePower);
shading interp
colorbar

hold on;

for j=1:12
w=facet(j).position;
patch(w(:,1),w(:,2),w(:,3),'b');
axis equal;
axis tight;
axis off;
end

end
```

## 19. Minimum – Maximum Algorithm

```

a=RoomB;

j=1;
Object=struct([]);
for i=1:size(a,1)
    if a.o(i,1)~='f'
        if a.o(i,1)=='v'

            Object(j).Vertex(i,:)=(a(i:i,2:4));

        elseif a.o(i,1)=='vn'
            k=i-size( Object(j).Vertex,1);
            Object(j).normal(k,:)=(a(i:i,2:4));

        end
    else
        if i<size(a,1)
            if a.o(i+1,1)=='o'
                i=i+1;
                j=j+1;
            else
                continue;
            end
        end
    end
end

temp_object=struct([]);
for m=1:size(Object,2)
    Object(m).Vertex(~any(table2array(Object(m).Vertex),2),:)=[];
    temp_object(m).Vertex=Object(m).Vertex;
    for i=1:size(Object(m).normal,1)

        if lt(table2array(Object(m).normal(i,1)),0)
            temp_object(m).X_negative(i,:)=Object(m).normal(i,1);
%           temp_object(m).X_Vert_Negative(i,:)=Object(m).Vertex;
%           tem=max(temp_object(m).X_Vert_Positive),;
        else
            temp_object(m).X_positive(i,:)=Object(m).normal(i,1);
%           temp_object(m).X_Vert_Positive(i,:)=Object(m).Vertex(i:i,1:3);
        end

        if lt(table2array(Object(m).normal(i,2)),0)
            temp_object(m).Y_negative(i,:)=Object(m).normal(i,2);
%           temp_object(m).Y_Vert_Negative(i,:)=Object(m).Vertex(i:i,1:3);
        else
            temp_object(m).Y_positive(i,:)=Object(m).normal(i,2);
%           temp_object(m).Y_Vert_Positive(i,:)=Object(m).Vertex(i:i,1:3);
        end

        if lt(table2array(Object(m).normal(i,3)),0)
            temp_object(m).Z_negative(i,:)=Object(m).normal(i,3);
%           temp_object(m).Z_Vert_Negative(i,:)=Object(m).Vertex(i:i,1:3);

```

```

        else
            temp_object(m).Z_positive(i,:)=Object(m).normal(i,3);
%            temp_object(m).Z_Vert_Positive(i,:)=Object(m).Vertex(i:i,1:3);
        end

    end

    if isempty(temp_object(m).X_positive)==0 &&
        isempty(temp_object(m).X_negative)==0
        if
            mean(table2array(temp_object(m).X_positive))>mean(abs(table2array(temp_
            object(m).X_negative)))
            x=mean(table2array(temp_object(m).X_positive));
            temp_object(m).X_direction=('X_positive');

        else
            x=mean(abs(table2array(temp_object(m).X_negative)));
            temp_object(m).X_direction=('X_negative');
        end
    end

    if isempty(temp_object(m).Y_positive)==0 &&
        isempty(temp_object(m).Y_negative)==0
        if
            mean(table2array(temp_object(m).Y_positive))>mean(abs(table2array(temp_
            object(m).Y_negative)))
            y=mean(table2array(temp_object(m).Y_positive));
            temp_object(m).Y_direction=('Y_positive');

        else
            y=mean(abs(table2array(temp_object(m).Y_negative)));
            temp_object(m).Y_direction=('Y_negative');
        end
    end

    if isempty(temp_object(m).Z_positive)==0 &&
        isempty(temp_object(m).Z_negative)==0
        if
            mean(table2array(temp_object(m).Z_positive))>mean(abs(table2array(temp_
            object(m).Z_negative)))
            z=mean(table2array(temp_object(m).Z_positive));
            temp_object(m).Z_direction=('Z_positive');

        else
            z=mean(abs(table2array(temp_object(m).Z_negative)));
            temp_object(m).Z_direction=('Z_negative');
        end
    end

    if x>y
        if x>z
            temp_object(m).direction=temp_object(m).X_direction;
            disp(temp_object(m).X_direction);
        else
            disp(temp_object(m).Z_direction);
            temp_object(m).direction=temp_object(m).Z_direction;
        end
    else
        if y>z

```

```

        disp(temp_object(m).Y_direction);
        temp_object(m).direction=temp_object(m).Y_direction;
    else
        disp(temp_object(m).Z_direction);
        temp_object(m).direction=temp_object(m).Z_direction;
    end
end

end

wall=struct([]);
p=0;q=0;r=0;s=0;
for n=1:size(temp_object,2)
    if isempty(temp_object(n).direction)==0
        if temp_object(n).direction=='X_positive'
            p=p+1;
            wall(1).X_Positive_wall(p).Vertex=temp_object(n).Vertex;
        elseif temp_object(n).direction=='X_negative'
            q=q+1;
            wall(2).X_Negative_wall(q).Vertex=temp_object(n).Vertex;

        elseif temp_object(n).direction=='Z_positive'
            r=r+1;
            wall(3).Z_Positive_wall(r).Vertex=temp_object(n).Vertex;
        elseif temp_object(n).direction=='Z_negative'
            s=s+1;
            wall(4).Z_Negative_wall(s).Vertex=temp_object(n).Vertex;
        end
    end
end

end

ceiling_floor=struct([]);
for n=1:size(temp_object,2)
    if isempty(temp_object(n).direction)==0
        if temp_object(n).direction=='Y_positive'
            p=p+1;
            ceiling_floor(1).Y_Positive_wall(p).Vertex=temp_object(n).Vertex;
        elseif temp_object(n).direction=='Y_negative'
            q=q+1;
            ceiling_floor(2).Y_Negative_wall(q).Vertex=temp_object(n).Vertex;
        end
    end
end

end

wall(1).wall1=[];
for t=1:size(wall(1).X_Positive_wall,2)
    % find the (Xmax,Zmax), (Xmin,Zmin) & (Xmax,Zmin)|(Xmin,Zmin) TO FIND
    % THE ORIENTATION OF ROOM w.r.t X,Y,Z COORDINATE SYSTEM.
    % [u,o]=max(table2array(wall(1).X_Positive_wall(t).Vertex));

    wall(1).wall1=vertcat(wall(1).wall1,wall(1).X_Positive_wall(t).Vertex);
    %
    wall(1).X_Positive_wall(t).dimensions=array2table(max(table2array(wall(

```

```

1).X_Positive_wall(t).Vertex))+abs(min(table2array(wall(1).X_Positive_w
all(t).Vertex))));
% wall(1).dimensions(t,:)=wall(1).X_Positive_wall(t).dimensions;
end
u=0;o=0;
% find maximum of x,y,z, and apply distance formula between 2 corner
points.
%wall(1).X_Positive_wall(t).dimensions=array2table(sqrt(max(table2array
(wall(1).X_Positive_wall(t).Vertex).^2+min(table2array(wall(1).X_Posit
ive_wall(t).Vertex).^2));
[u,i]=max((wall(1).wall1.Object1(:,1)));
u=(wall(1).wall1(i,:));
[o,i]=min((wall(1).wall1.Object1(:,1)));
o=(wall(1).wall1(i,:));
wall(1).Length=abs(u.Object1)+abs(o.Object1);
disp("length1");
disp(wall(1).Length);
figure
plot3(wall(1).wall1(:,1),wall(1).wall1(:,2),wall(1).wall1(:,3));

wall(2).wall2=[];
if isfield(wall(2),'X_Negative_wall')==0
l2=0;
wall(2).Length=0;
else
for t=1:size(wall(2).X_Negative_wall,2)
%
wall(2).X_Negative_wall(t).dimensions=array2table(sqrt(max(table2array(
wall(2).X_Negative_wall(t).Vertex).^2+min(table2array(wall(2).X_Negati
ve_wall(t).Vertex).^2));
% %
wall(2).X_Negative_wall(t).dimensions=array2table(max(table2array(wall(
2).X_Negative_wall(t).Vertex))+abs(min(table2array(wall(2).X_Negative_w
all(t).Vertex))));
% wall(2).dimensions(t,:)=wall(2).X_Negative_wall(t).dimensions;

wall(2).wall2=vertcat(wall(2).wall2,wall(2).X_Negative_wall(t).Vertex);
end
% l2=max(table2array(wall(2).dimensions(:,1)));
[u,i]=max((wall(2).wall2.Object1(:,1)));
u=(wall(2).wall2(i,:));
[o,i]=min((wall(2).wall2.Object1(:,1)));
o=(wall(2).wall2(i,:));
wall(2).Length=abs(u.Object1)+abs(o.Object1);
disp("length2");
disp(wall(2).Length);
figure
plot3(wall(2).wall2(:,1),wall(2).wall2(:,2),wall(2).wall2(:,3));

end

wall(3).wall3=[];
for t=1:size(wall(3).Z_Positive_wall,2)
%wall(3).Z_Positive_wall(t).dimensions=array2table(sqrt(max(table2array
(wall(3).Z_Positive_wall(t).Vertex).^2+min(table2array(wall(3).Z_Posit
ive_wall(t).Vertex).^2));
%
wall(3).Z_Positive_wall(t).dimensions=array2table(max(table2array(wall(

```

```

3).Z_Positive_wall(t).Vertex))+abs(min(table2array(wall(3).Z_Positive_w
all(t).Vertex))));
% wall(3).dimensions(t,:)=wall(3).Z_Positive_wall(t).dimensions;

wall(3).wall3=vertcat(wall(3).wall3,wall(3).Z_Positive_wall(t).Vertex);
end
[u,i]=max((wall(3).wall3.VarName4(:,1)));
u=(wall(3).wall3(i,:));
[o,i]=min((wall(3).wall3.VarName4(:,1))));
o=(wall(3).wall3(i,:));
wall(3).Length=abs(u.VarName4)+abs(o.VarName4);
disp("length3");
disp(wall(3).Length);
figure
plot3(wall(3).wall3{:,1},wall(3).wall3{:,2},wall(3).wall3{:,3});

wall(4).wall4=[];
if isfield(wall(4),'Z_Negative_wall')==0
    b2=0;
else
    for t=1:size(wall(4).Z_Negative_wall,2)
        %
        wall(4).Z_Negative_wall(t).dimensions=array2table(sqrt(max(table2array(
wall(4).Z_Negative_wall(t).Vertex).^2+min(table2array(wall(4).Z_Negati
ve_wall(t).Vertex).^2)));
        %
        wall(4).Z_Negative_wall(t).dimensions=array2table(max(table2array(wall(
4).Z_Negative_wall(t).Vertex))+abs(min(table2array(wall(4).Z_Negative_w
all(t).Vertex))));
        % wall(4).dimensions(t,:)=wall(4).Z_Negative_wall(t).dimensions;

wall(4).wall4=vertcat(wall(4).wall4,wall(4).Z_Negative_wall(t).Vertex);
end
% b2=max(table2array(wall(4).dimensions(:,3)));
[u,i]=max((wall(4).wall4.VarName4(:,1)));
u=(wall(4).wall4(i,:));
[o,i]=min((wall(4).wall4.VarName4(:,1))));
o=(wall(4).wall4(i,:));
wall(4).Length=abs(u.VarName4)+abs(o.VarName4);
disp("length4");
disp(wall(4).Length);
%figure
plot3(wall(4).wall4{:,1},wall(4).wall4{:,2},wall(4).wall4{:,3});
end

for t=1:size(ceiling_floor(2).Y_Negative_wall,2)
    %
    idx=all(cellfun(@isempty,ceiling_floor(2).Y_Negative_wall(t).Vertex{:,
}),2);
    % ceiling_floor(2).Y_Negative_wall(t).Vertex(idx,:)=[];
    if isempty(ceiling_floor(2).Y_Negative_wall(t).Vertex)==0

ceiling_floor(2).Y_Negative_wall(t).dimensions=array2table(max(table2ar

```



```

ray(ceiling_floor(2).Y_Negative_wall(t).Vertex))+abs(min(table2array(ceiling_floor(2).Y_Negative_wall(t).Vertex))));
%
wall(4).Z_Negative_wall(t).dimensions=array2table(max(table2array(wall(4).Z_Negative_wall(t).Vertex))+abs(min(table2array(wall(4).Z_Negative_wall(t).Vertex))));

ceiling_floor(2).dimensions(t,:)=ceiling_floor(2).Y_Negative_wall(t).dimensions;
end
end

for t=1:size(ceiling_floor(1).Y_Positive_wall,2)
    if isempty(ceiling_floor(1).Y_Positive_wall(t).Vertex)==0

ceiling_floor(1).Y_Positive_wall(t).dimensions=array2table(max(table2array(ceiling_floor(1).Y_Positive_wall(t).Vertex))+abs(min(table2array(ceiling_floor(1).Y_Positive_wall(t).Vertex))));
%
wall(4).Z_Negative_wall(t).dimensions=array2table(max(table2array(wall(4).Z_Negative_wall(t).Vertex))+abs(min(table2array(wall(4).Z_Negative_wall(t).Vertex))));

ceiling_floor(1).dimensions(t,:)=ceiling_floor(1).Y_Positive_wall(t).dimensions;
end
end

h1=max(table2array(ceiling_floor(1).dimensions(:,2)));
h2=max(table2array(ceiling_floor(2).dimensions(:,2)));

height=2.7;
length=mean(length1,length3);
breadth=mean(length2,length4);

%for creating data table
l1=wall(1).Length;
l2=wall(2).Length;
l3=wall(3).Length;
l4=wall(4).Length;

Room(1).Wall(1).Coordinates(1:4,1:3)=[0 0 0;0 0 height ;0 breadth height ; 0 breadth 0 ];
Room(1).Wall(2).Coordinates(1:4,1:3)=[ 0 breadth 0;0 breadth height; length breadth height ;length breadth 0];
Room(1).Wall(3).Coordinates(1:4,1:3)=[length breadth 0; length breadth height ; length 0 height ;length 0 0 ];
Room(1).Wall(4).Coordinates(1:4,1:3)=[length 0 0; length 0 height ;0 0 height ;0 0 0];

figure;
hold on;
for i=1:numel(Room)
    for j=1:4
        w=Room(i).Wall(j).Coordinates;

```

```

patch(w(:,1),w(:,2),w(:,3),'b');
axis equal;
axis tight;

alpha(0.5);

end
end

figure;
plot(wall(1).wall1(:,1),wall(1).wall1(:,3));

newt=[];
size=size(Object);
for i=1:size(1,2)
    newt=vertcat(newt,Object(i).Vertex);
end
figure
plot(newt(:,1),newt(:,3));
plot3(newt(:,1),newt(:,2),newt(:,3));

for i=1:size(temp_object,2)
    figure;
    plot(temp_object(i).Vertex(:,1),temp_object(i).Vertex(:,3));
end

```

## REFERENCES

- [1] T. Gupta, and H. Li, "Indoor mapping for smart cities — An affordable approach: Using Kinect Sensor, and ZED stereo camera," International Conference on Indoor Positioning, and Indoor Navigation (IPIN), pp. 1-8, 2017.
- [2] N. I. A. M. Nazri, and D. R. A. Rambli, "Current limitations, and opportunities in mobile AR applications," International Conference on Computer, and Information Sciences (ICCOINS), pp. 1-4, 2014.
- [3] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile AR Survey: From Where We Are to Where We Go," IEEE Access, pp. 6917 - 6950, 2017.
- [4] M. E. C. Santos, "AR Learning Experiences: Survey of Prototype Design, and Evaluation," IEEE Transactions on Learning Technologies, vol. 7, pp. 38-56, 2014.
- [5] L. C. Chen, N. V. Thai, and H. I. Lin, "Real-time 3-D feature detection, and correspondence refinement for indoor environment-mapping using RGB-D cameras," IEEE International Symposium on Industrial Electronics, Taipei, Taiwan, pp. 1-6, 2013.
- [6] X. Xu, and H. Fan, "Feature based simultaneous localization, and semi-dense mapping with monocular camera," Image, and Signal Processing, BioMedical Engineering, and Informatics (CISP-BMEI), International Congress on, pp. 17-22, 2016.
- [7] J. P. Collomosse, "Real-time environment mapping for stylised AR," The 3rd European Conference on Visual Media Production (CVMP), pp. 184-184, 2006.
- [8] S. Damodaran, A. P. Sudheer, and T. K. Sunil Kumar, "An evaluation of spatial mapping of indoor environment based on point cloud registration using Kinect sensor," Control Communication & Computing India (ICCC), International Conference on, pp. 545-552, 2015.
- [9] S. Singh, and R. Singh, "Comparison of various edge detection techniques," 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, pp. 393-396, 2015.
- [10] X. Ge, "Multipath Cooperative Communications Networks for Augmented, and Virtual Reality Transmission," IEEE Transactions on Multimedia, vol. 19, no. 10, pp. 2345-2358, 2017.
- [14] X. C. He, and N. H. C. Yung, "Corner detector based on global, and local curvature properties," Optical Engineering 47, no. 5, pp. 1-4, 2017.
- [11] P. Ram, and S. Padmavathi, "Analysis of Harris corner detection for color images," International Conference on Signal Processing, Communication, Power, and Embedded System (SCOPEs), pp. 405-410, 2016.
- [12] Bastanlar, Yalin, and Y. Yardimci, "Corner validation based on extracted corner properties", Computer Vision, and Image Understanding, 112, pp. 243-261, 2008.
- [13] X. Yang, and Y. Tian, "Robust door detection in unfamiliar environments by combining edge, and corner features," IEEE Computer Society Conference on Computer Vision, and Pattern Recognition - Workshops, San Francisco, CA, pp. 57-64, 2010.

- [14] X. C. He, and N. H. C. Yung, "Corner detector based on global, and local curvature properties," *Optical Engineering* 47, no. 5, pp. 1-4, 2017.
- [15] P. Ram, and S. Padmavathi, "Analysis of Harris corner detection for color images," *International Conference on Signal Processing, Communication, Power, and Embedded System (SCOPEs)*, pp. 405-410, 2016.
- [16] Bastanlar, Yalin, and Y. Yardimci, "Corner validation based on extracted corner properties", *Computer Vision, and Image Understanding*, 112, pp. 243-261, 2008.
- [17] X. Yang, and Y. Tian, "Robust door detection in unfamiliar environments by combining edge, and corner features," *IEEE Computer Society Conference on Computer Vision, and Pattern Recognition - Workshops*, San Francisco, CA, pp. 57-64, 2010.
- [18] Spatial Mapping. Retrieved from <https://docs.microsoft.com/enus/windows/mixed-reality/spatial-mapping>
- [19] Microsoft HoloLens hardware details. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details>
- [20] Expanding the spatial mapping capabilities of HoloLens. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/case-study-expanding-the-spatial-mapping-capabilities-of-hololens>.
- [21] V. Siddaraju, and G. Koutitas, "An AR Facet Mapping Technique for Ray Tracing Applications", *ICDT 2018: The Thirteenth International Conference on Digital Telecommunications*, Athens, Greece, pp 6-12, 2018.
- [22] Microsoft mixed reality tool kit. Retrieved from <https://github.com/Microsoft/MixedRealityToolkit-Unity>.
- [23] Coordinate systems. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/coordinate-systems>
- [24] Sean Ong. "Beginning of Windows Mixed Reality Programming", Apress 2017.
- [25] MR spatial 230: spatial mapping. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/holograms-230>
- [26] Spatial mapping design. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping-design>
- [27] Spatial anchors. Retrieved from <https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-anchors>
- [28] What's inside Microsoft's HoloLens, and How it works. Retrieved from <https://www.tomshardware.com/news/microsoft-hololens-components-hpu-28nm,32546.html>
- [29] D. H. Okulu, "3D spatial layout extraction of indoor images using RGB-D Data", *IEEE Computer Society Conference on Computer Vision, and Pattern Recognition - Workshops*, San Francisco, CA, pp. 57-64, 2010.