A Sensor Fusion Approach to Extending Range of Environment Mapping Devices

Andrew Ciambrone, Denis Gračanin

Virginia Tech, Blacksburg, VA, USA

Krešimir Matković

VRVis Research Center, Vienna, Austria

Abstract

Currently available commercial spatial mapping devices mostly use infrared camera to obtain a depth map which is effective only for short to medium distances (3-4 meters). However, that range can be extended by using existing environment mapping devices and techniques and a combination of a camera, Inertial Measurement Unit, and Light Detection and Ranging devices supported by sensor fusion and computer vision techniques. The proposed approach consists of three steps. The first step is data collection and data fusion using embedded hardware; the second step is data processing (segmentation) and the third step is creating a geometry mesh of the environment. The developed system was evaluated by measuring the room dimension and objects within the room. This low cost system can expand the mapping range of the existing mixed reality devices such as Microsoft HoloLens device.

Keywords: mixed reality, sensor fusion, environment mapping, image processing, computer vision

1. Introduction

Spatial mapping or environment mapping is the process of exploring a real world environment and creating a digital representation of it. This process is used in various application domains, including mixed reality (MR). Current MR commercial devices, such as Microsoft HoloLens device, include support for environment mapping. However, the mapping is effective for short to medium ranges (3-4 meters). Most built environments contain large spaces that are not suitable for these devices thus limiting the applicability. The described approach focuses on augmenting the environment mapping capabilities of the current MR devices.

Real time environment mapping is an important component of many systems and is a well-researched concept. The real time approaches map the environment on the fly, usually using cloud point data. They are limited by the amount of time the system has to process the data (the amount of time it takes for the next frame of information to come in). Due to the limitations of a MR system, processing high density point cloud is difficult due to the limitations of portable Light Detection and Ranging (LIDAR) technologies and computer vision techniques. Creating a dense point cloud is often achieved by using expensive image processing techniques or LIDAR systems.

The proposed approach extends the existing environment mapping devices and techniques to map larger architectural environments using a combination of a camera, Inertial Measurement Unit (IMU), and LIDAR devices supported by sensor fusion and computer vision. The developed system was evaluated on how accurately it can estimate the dimensions of the space it is located in. The system was also compared to an existing high density LIDAR system.

2. Related Work

Mapping an environment can be done in three different ways, geometrically, radiometrically and semantically [1]. The geometric mapping spatially maps the real world, the radiometric mapping maps the color of the world, and the semantic mapping creates an understanding of the environment. Geometric mapping usually uses sensors such as LIDAR or range finder cameras while radiometric mapping typically uses cameras but could also use thermal imaging. Semantic mapping tries to create understanding in the scene such as object detection or understanding where objects are in reference to each other. Recently, the use of deep learning [1] has gained attention.

For cloud point collection there are two main types of methods, passive and active methods. Passive methods rely on reasonable lighting and typically use imagery for their methods of point collection. Active methods manipulate the scene to gather more information. Microsoft HoloLens and Kinect are good examples of mapping devices that use active methods to determine distances by using an infrared laser. An example of a passive method would be the "structure from motion" technique [1].

Feature extraction is the idea of locating point of interest from an image. Examples of the features in an 2D image are: pixel properties, textures, and shapes within an image. Texture features are the spatial placement of the intensity values in an image. For example, a checkerboard would give a different texture compared to a spherical contour. Looking at the texture features of an image is used to determine regions of an image [1]. Shape features can be detected by using contour filters such as the Canny, Sobel, Roberts, and Prewitt operators [1].

The simultaneous localization and mapping (SLAM) problem is one of incrementally building a consistent map of the environment and simultaneously determining its location within the map [2]. Work done in this area is closely related to the work done for the MR environment mapping.

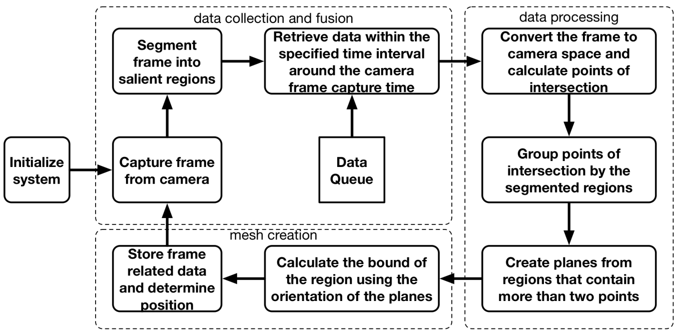
Sensor fusion [3, 4] uses intelligent integration of data derived from a collection of disparate sensor so that the resulting information provide more accurate information or information that cannot be derived from individual sensors.

Image segmentation is the process of dividing a image into discrete parts [5]. The most common alfgorithms include clustering, edge detection, and machine learning techniques such as neural networks. Superpixel segmentation [6] is a method of image segmentation. Seeded Region Growing [7] is a image segmentation method that is quite often used in computer vision. The algorithm starts by picking an set of *N* seeds. Then for each iteration each of the seeds gain one additional pixel to its region. This is done by looking at the pixels that border the regions. The pixel that is added to the region is determined by the least difference from the region. If a pixel is approached by two regions then the algorithm considers that to be a boundary pixel [8]. The success of the algorithm relies on the selection of initial pixel seeds, e.g., the converging squares algorithm [9].

3. Problem Description

MR devices tend to be lightweight mobile devices. Their processing power, memory and other resources are constrained. Limited memory and limited CPU power mean that decisions must be made on how much data can be collected. Collecting too much data could introduce delay causing a latency between what the user sees and what the systems outputs. One solution to this problem would be to collect the sensor data on the device and process it on a more powerful machine by sending it over the network. However, this would add a bottleneck to the system due to network constraints.

Mapping large architectural (indoor) spaces is challenging due to a limited range of currently available MR devices. It can take some time for the user to navigate the whole area and complete space mapping. While user interactions do take place in the immediate surrounding of the user, having some information of the more distant parts of the environment (without the need for the user to go there) could provide better context for user interactions. For example, mapping objects further away



data collection and fusion
Segment frame into salient regions
Retrieve data within the specified time interval around the camera frame capture time
Initialize system
Capture frame from camera
Data Queue
data processing
Convert the frame to camera space and calculate points of intersection
Group points of intersection by the segmented regions
Create planes from regions that contain more than two points
mesh creation
Store frame related data and determine position
Calculate the bound of the region using the orientation of the planes

Figure 1: An outline of the system workflow.

(such as walls and high ceilings) enables construction of the model of the surrounding architectural space.

That can be achieved by augmenting the current MR devices with additional longer-range sensor technologies. However, these additions must be light and mobile and safe for the user to use. Due to these constraints main difficult design decisions must be made to produce fast but not necessarily highly accurate results. However, since this those results describe areas of space more distant form the user, even less accurate results are sufficient. When the user moves closer, the near-range mapping hardware is used to improve the accuracy.

4. Proposed Approach

Figure 1 shows the overall approach. The main processing loop is based on the camera input. Each camera frame is annotated by LIDAR and IMU data within a specified time distance from the frame capture time. The fused data is used to determine region in space and construct the surrounding geometry.

Overall, there are three main components. The first part focuses on data collection and data fusion. The second part processes the collected data and finally, the third component formulates and outputs a 3D mesh.

There are several types of data to be collected and processed by this system: images, range measurements, and the devices specific acceleration, gyro and magnetic data. This is achieved by using a combination of a camera, a LIDAR device, and an IMU device. The quality of the data and the rate of collection depends on the quality of the device. While having more data samples would create a better result, the system must fit within the constraints of a wearable device. Most LIDAR devices are big and bulky and are not portable. The most promising type of LIDAR would be a singular range finder device such as the Garmin LIDAR Lite v3.

Once the data is collected and bundled together based on the time it was captured, the system can then start processing the data. The first step is to segment the image into meaningful chunks. The purpose of the segmentation is to group pixels of the image into sections so that when the LIDAR does hit that portion of the image we can interpolate entire regions instead of just points of the image. There are various strategies for segmenting an image, such as edge detection, convolutional neural networks, or region-growing.

Given the segmented image the next step in the processing phase is to create the actual mesh from the images. Since the image is segmented into parts, the system assumes that the segments lie the same plane of orientation. To determine the region's orientation at least three LIDAR points must be captured on that particular region.

With a region's orientation it is possible to determine the size and the bounds of the region in the real world. Calculating the bounds of the region in the world space can be done by finding the intersection between a ray that coming from the camera and the calculated plane in camera space.

4.1. Data Collection and Processing

The following data types of data are being collected and used: the specific force or acceleration of the system, the angular rate, the magnetic field, image data, and distance data. To reduce the natural error of the IMU, sensor fusion is performed on the data. Since time performance is essential, a simplified version of Kalman filter, called RTQF, from the RTIMULib library is used [10]. The sampling rate for these components are 80 samples / second for the IMU sensor and 100 samples / second for the LIDAR sensor. The camera is capable of capturing up to 30 frames per second. However, due to the time it takes to process each frame the effective camera frame rate is reduced to approximately 5-6 frames per second. However, that is sufficient to map the environment given the user's speed.

We first process the data collected from the sensors before converting it into geometry meshes. This is done in three main steps, image segmentation, image annotation, and plane creation. The success of this system relies heavily on the segmentation image. A segmentation algorithm takes in image and partitions it into discrete parts. The main goal of any image segmentation is to simplify the image by grouping pixels together by corresponding surfaces or objects.

In this system the goals of the segmentation is to separate the image into surfaces and to get the bounds of the region. The system needs a segmentation algorithm that maintains consistency of the generated regions, be as close to real time as possible, and add a bias to creating larger regions. In indoor spaces there are a lot of variations in the image due to surface textures and lighting differences. Overhead lights and windows can create areas of intense light differences. Consequently, creating a good segmentation is difficult. To overcome these issues, images must be preprocessed before segmentation.

One major flaw with many image segmentation approaches is that they are often prone to noise in the image so steps must be taken to reduce the noise before the segmentation can be ran on the image. This can be done by blurring or smoothing the image. Blurring an image reduces the amount of noise and details in an image. Figure 2 left shows a segmentation without filtering. The black regions are segments below the minimum filter size. Figure 2 right shows a segmentation with filtering.

A filtered image is preferred compared to the unfiltered image because the noise and tiny details in a image prevent segmenting surfaces into larger segments. For example, when segmenting a brick wall the system would perform better if it created only one segment for the wall instead of having a segment for each brick. However, image blurring does have a cost. Blurring processes each pixel in an image which does add a computation required for each frame. It is important to realize the amount of blurring that takes place. Too much blurring and the image will lose critical details such as the edges between surfaces. Popular methods of image smoothing include Gaussian, median, and bilateral.

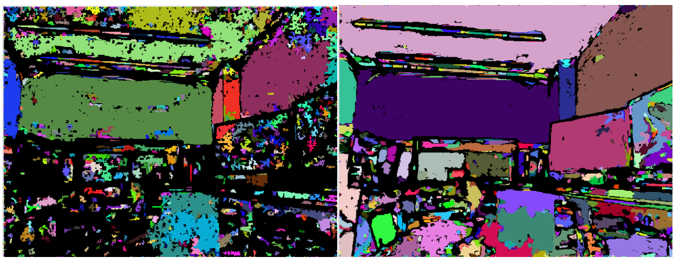


Figure 2: Segmentation without (left) and with filtering (right).

The median filter is used because it preserves the edges better then the Gaussian filter while at the same time removing noise in it. Although for this application it was found that while the bilateral filter does perform better than the median filter in retaining edges, it also takes much longer to process compared to the median filter. *RGB* is the typical format that frames from a camera come in however a color translation to a different color space such as the $L*A*B$ is rather simple. In this case the color space transformation is done using the OpenCV library.

4.2. Segmentation Algorithm

While the filtering and changing the color space does help with the differential lighting, it still holds a major influence in the image segmentation choice. Many of the image segmentation processes use a process called global thresholding. That works well for images that are consistent in lighting such as microscopic images but does not work well in an indoor space with overhead lighting. Therefore, the segmentation algorithm has to use adaptive threshold or some other method to compensate for the light differences.

Finally, the amount of time needed for the segmentation was a major decision point in choosing a optimal segmentation algorithm. Currently, a great deal of research is ongoing in using convolutional neural network (CNN) to segment an image. The selected image segmentation algorithm (Algorithm 1) is a simplified unseeded region growing algorithm.

Algorithm 1 Image Segmentation

```
procedure REGION GROWING($Img, MinRegionSize$)
    $Regions \leftarrow \emptyset$
    $RegionMap \leftarrow$ [][] (all zeros)
    $RegionCounter = 1$
    $Seeds \leftarrow \emptyset$
    $CurrentRegion \leftarrow$ NULL
    while Pixels left to process do
        if Seeds.length = 0 then
            if CurrentRegion != NULL then
                $Regions \leftarrow Regions \cup \{CurrentRegion\}$
            $newSeed \in RegionMap \neq 0$
            $CurrentSeeds \leftarrow newSeed$
            $CurrentRegion \leftarrow New Region$ based on $newSeed$
        repeat
            $CurrentSeed \leftarrow seed \in Seeds$
            for all $Neighbor \in$ 8 or 4 neighbor near $CurrentSeed$ do
                if $Distance(CurrentSeed.Color, Neighbor.Color) < Threshold$ then
                    $CurrentSeeds \leftarrow CurrentSeeds \cup \{Neighbor\}$
                    $CurrentRegion \leftarrow CurrentRegion \cup \{Neighbor\}$
        until $Seeds.length = 0$
        Return Region Merging($Regions, MinRegionSize$)
```

The region growing was selected because it creates consistent segments and follows the guidelines of the parameters pass such as minimum segment size. Another reason why region growing was selected over other methods was because unlike many of image segmentation algorithms region growing is a local method. It is only concerned with certain portions of the

Algorithm 2 Merge Regions

```
procedure Region Merging(Regions, MinRegionSize)
    for all Region $\in$ Regions do
        if Region.size < MinRegionSize then
            if All Neighboring Regions are the same then
                NRegion = NRegion $\cup$ Region
            for all NeighRegion do
                if Distance(Region.AvgColor, NRegion.AvgColor) < Threshold then
                    Region = Region $\cup$ NRegion
        Return Regions
```

image during the segmentation which works well with the differences in lighting typically found in architectural spaces.

The region growing algorithm includes seed selection, region growing and mesh creation. Seed selection can be crucial to seeded region growing algorithms because those algorithms typically only have a number of regions they can grow. The algorithm above is considered unseeded because no initial seeds are set through the parameters. There is also no real hard limit to the number of regions except for the number of pixels the image. Since there is no limit to the number of seeds, the selection of initial seeds for a region has little impact and can be negated by region merging. However, a heuristic that uses a peaks method in [7] is a good method of seed selection. This method looks for the points of highest variation in the image.

After an initial seed is selected the algorithm the region growing portion begins. Popular methods for defining what a neighbor are the 8-connected or the 4-connected. In this system it was found that the 4-connected method has a slight advantage in creating an accurate segmentation. However, the advantage is not enough to justify the additional iterations the 4-connected neighborhood does create. If the color difference or distance between current region seed and the pixel neighbor isn't too large, the pixel neighbor will be added to the current region being built and added to the list of pixels to process. This process continues until there are no more seeds left. If a region runs out of seeds to process and there are still pixels left in the image left unlabeled, then another region is created on an unlabeled pixel.

Once every pixel in the image has been labeled with a region, the region merging portion of the algorithm begins. Region merging is used because the image contains artifacts from the camera sensor that were not filtered out by the median filter. For every region that is smaller then the minimum desired size, the region looks at its neighboring regions and determines if it can merge with one of those. There are two conditions to decide if a region will be merged with another. The first condition is if the difference between the average color of the region and its neighboring region is within the threshold. This condition prevents a bias from occurring with the order of the pixel processing. The second condition is when another region encompasses a region they are merged. This means that small regions that are surrounded by the same larger region will be merged with that region.

The properties of the created segments are.

1. All the pixels in the frame must belong to a region.
2. The points in the pixel must be connected at some point meaning no two isolated groups of pixels can be considered parts of the same region.
3. All regions are disjoint. There are no overlaps.

While it would be best if all of the data points are used, only those data points around the time of frame capture can be used to create independent planes. This constraint verifies that the initial position of the LIDAR data points were collected are around the same position that the frame was collected. The time range value is determined based on the assumption that the user does not move faster than one meter / second. With the frame segmented and the relevant data selected, the next step is to find where on the frame the LIDAR points lie. This is achieved by placing the frame into camera space.

Since there is little depth data, the system needs to extrapolate the depth's of other parts of the environment. This is done by assuming that the regions captured during the segmentation process lie on the same orientation plane. With the orientation of a region its bounds and size can be determined. Which allows the system to extrapolate additional depth points of the region. To achieve this the LIDAR points need to be grouped by the region which they fall into.

For each region with three or more points, a check is done to verify if the points are well suited to create the plane. If the three points are overlapping or are co-linear, creating a plane is not possible. In cases where there are more then three points a search must be performed that chooses the three points that form the largest possible triangle. By creating the largest possible triangle the amount of error in the plane creation and boundary extrapolation is reduced.

Once the system has determined that the region does contain three points that are optimal for creating a plane, the system can then proceed to calculating the position of the bounds of the region in camera space. The current implementation only calculates for eight boundary points. However, this could be easily extended to more points. In order to correctly determine the boundary point locations we need to also calculate a plane for the region. All the boundary points and LIDAR points are saved and used are used to represent the mesh of that region.

4.3. Implementation

Because the target MR devices are lightweight, the developed system also has to be lightweight. We use Raspberry PI 3, Pi camera, the Garmin LIDAR Lite 3 and the IMU based on MPU9250. The total cost of the system is approximately $200. Figure 3 shows the develop system (bottom right) deployed during the experiment.

5. Evaluation

The described system was evaluated by its ability to predict environment dimensions from the points in the created mesh and by its ability to estimate the dimensions of an object in the environment.

When evaluating the system for its ability to detect the dimensions of the room, the system's data was compared against the data from the LIDAR puck and against a ground truth which was determined from measurements taken from both a laser

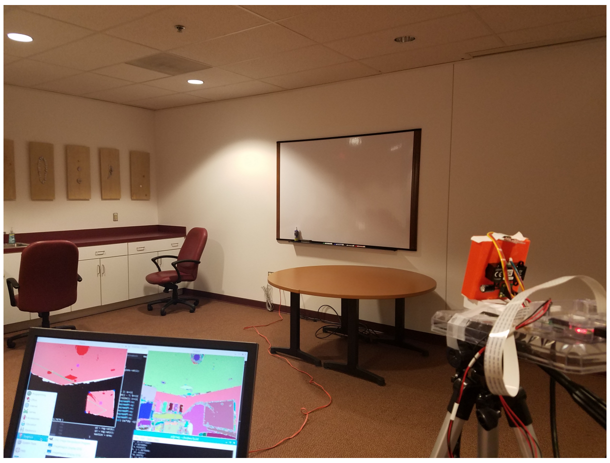


Figure 3: Object dimension evaluation setup.

tape measure and a measuring tape. The evaluation was conducted in a room where Microsoft HoloLens device would be unable to detect the walls if placed in the center of the room due to the size of the room. To provide a comparison between modern state of the art versus our implementation the system was compared against the Velodyne LIDAR Puck (VPL-16). Both the implemented system and the LIDAR Puck were placed in the center of the room at approximately the same height. The implemented system was placed on top of a tripod where it was rotated around the room completing a full rotation capturing different parts of the room.

The results from the Room Dimension evaluation can be found in Table 1. The dimensions of the room were calculated by finding the maximum and minimum values points in each of the spatial dimensions. However, due to the error in the gyroscope yaw value some of the extrapolated region boundary points may had caused some of the data points to be erroneous. To account for these erroneous data points a median value was calculated at each of the directions where meshes were created. Table 1 contains the calculated dimension of both data sets.

The percent error on the x-axis of the developed system for all data points is 42.8% however when accounting for the erroneous points the percent error on the x-axis is 1.5%. For the compared system the percent error on the x-axis is 26.5%. The error in the compared system was high due to the windows in the room. The percent error found on the y-axis of the developed system for all data points 30.1%. However when accounting for the erroneous points the percent error on the y-axis is 10.3%. For the compared system the percent error on the y-axis is 0.2%. The percent error on the z-axis of the developed system for all data points is 176.02% however when accounting for the erroneous points the percent error on the z-axis is 35.5%. For the compared system the percent error on the x-axis is 6.2%.

6. Discussion

Due to the error in the gyroscope's yaw an absolute orientation of the room was not going to accurately represent the direction that the developed system was pointed at. An estimate

Table 1: Dimensions of the room.

Metric	Length x (cm)	Width y (cm)	Height z (cm)
Ground Truth	1097.28	871.22	273.05
LIDAR Puck	1388.11	873.75	289.98
System (All Points)	1567.01	1133.04	482.54
System (Median)	1080.80	960.72	176.02

of the system's orientation was later applied to the data points collected. Frames were captured at eight different orientations evenly distributed on the z-axis. This estimate may have introduced errors into the estimate of the room dimensions. Another factor that introduced error on the z-axis was that no data was collected of the floor and ceiling. That is evident in the high error in approximation of the rooms height.

Another important observation to note is of the error on the x-axis for the LIDAR puck. The LIDAR puck was calibrated and measurements were taken multiple times following the direction of the manufacturer (Figure 4). However, the error on the x-axis remained. Figure 5 shows the corresponding data captured by the developed system in each corner of the room and the center areas of the walls.

An important part of environment mapping is accurately representing the dimensions of objects in a room. In this part of the evaluation the system ability to perform this task was evaluated. The dimensions of the object that the system was trying to measure a white board hanging on a wall (Figure 3).

The distance and angle between the object and the system affected the accuracy of the system's estimate of the object's dimensions. In order to test the system's ability to accurately estimate the dimension of an object the system was placed facing that object at various distances and angles about the room.

Table 2 shows the changes in accuracy from the different distances and angles taken around the room. Figure 6 shows

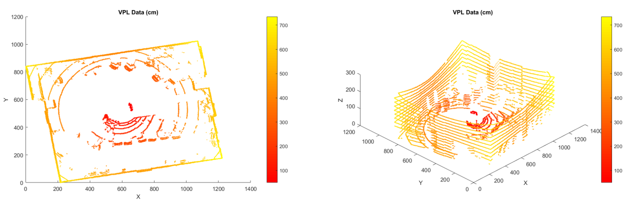


Figure 4: LIDAR data (top view)   LIDAR data (perspective view).

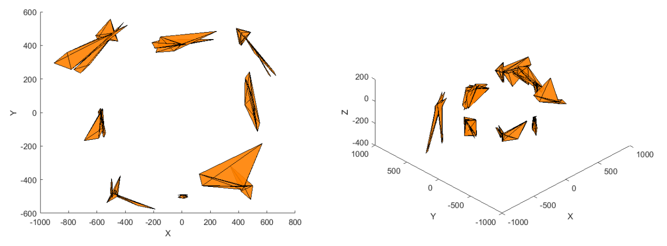


Figure 5: System data (top view)   System data (perspective view).

Table 2: Dimensions of the white board (in centimeters)

Metric	Width	Height	Angle°	Distance
Actual	169.22	117.48	*N/A*	*N/A*
Reading 1	167.63	109.20	0	182.88
Reading 2	164.14	117.58	0	271.78
Reading 3	157.85	106.10	0	383.54
Reading 4	168.79	127.73	0	510.54
Reading 5	147.53	100.62	0	645.16
Reading 6	141.24	114.12	25	353.06
Reading 7	136.00	124.57	50	347.98
Reading 8	94.10	126.81	60	342.90
Reading 9	102.31	127.18	50	490.22

the percent error of the dimensions based on the distance and angle of the implemented system.

There are a few things that cause the error in calculating the error in an objects dimensions. The first is accuracy of the segmentation. If the segmentation is too large or too small the predicted boundaries of the object are going to be different then what they truly are. The effect of the segmentation error has increases as the distance increases because the further the system is from the object the more spatial impact each pixel has.

However, the largest factor in the amount of error in the calculation is the error in the gyroscope, particularly the value of the yaw. Looking at Figure 6 bottom it is apparent that the amount of error in the object's width is much higher then the error in the height. Conversely, looking at Figure 6 top when the system is parallel to the white board, the error of the width and height are roughly around the same area.

7. Conclusion

While the system based on the proposed approach has been developed, implemented and tested, many improvements are possible. Because of the limitations of the hardware and how the system is being used, the results depend on the user of the system. If the user does not provide a good amount of rotational range then many of the LIDAR points may overlap or be co-linear meaning planes can't be constructed and meshes can not be generated. The approach is limited to working with sparse point clouds. Providing a visual feedback to the user in terms of space coverage will help improve the cloud point quality and enhance environment mapping.

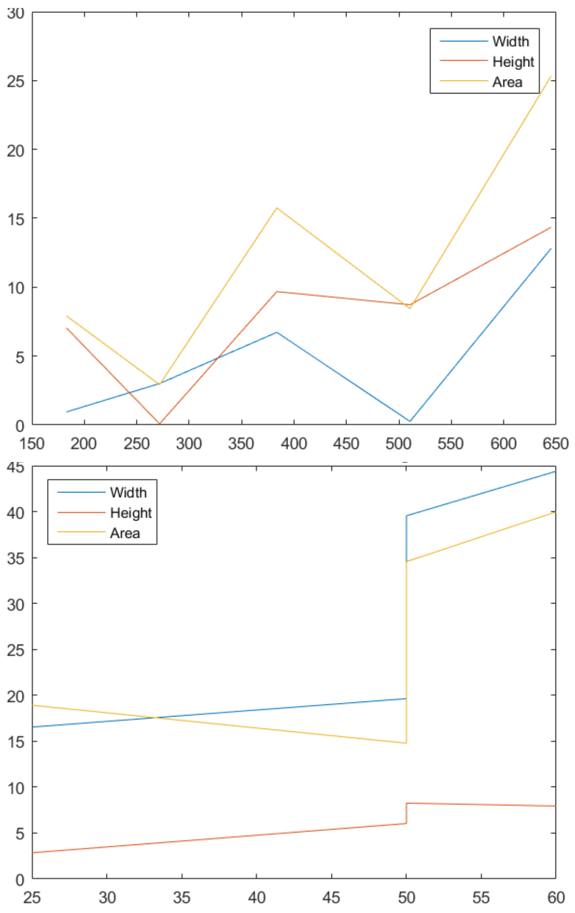


Figure 6: Top: Percentage error of object dimension estimate as a function of distance in centimeters. Bottom: Percentage error of object dimension estimate as a function of angle in degrees.

References

[1] Weinmann, M.. Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes. Springer; 2016.

[2] Durrant-Whyte, H., Bailey, T.. Simultaneous localization and mapping: Part I. IEEE Robotics Automation Magazine 2006;13(2):99–110.

[3] Das, S.. High-Level Data Fusion. Boston: Artech House; 2008.

[4] Kim, J.S., Gračanin, D., Quek, F.. Sensor-fusion walking-in-place interaction technique using mobile devices. In: Proceedings of the 2012 IEEE Virtual Reality Conference (VR 2012). 2012, p. 39–42.

[5] Gonzalez, R.C., Woods, R.E.. Digital Image Processing. Upper Saddle River, New Jersey 07458: Prentice Hall; second ed.; 2002.

[6] Ren, X., Malik, J.. Learning a classification model for segmentation. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. IEEE; 2003, p. 10–17.

[7] Adams, R., Bischof, L.. Seeded region growing. IEEE Transactions on pattern analysis and machine intelligence 1994;16(6):641–647.

[8] Kamdi, S., Krishna, R.. Image segmentation and region growing algorithm. International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2012;2.

[9] O'Gorman, L., Sanderson, A.C.. The converging squares algorithm: An efficient method for locating peaks in multidimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence 1984;PAMI-6(3):280–288.

[10] Lindegaard, J.B.. RTIMULib - a versatile 9-dof IMU library for embedded Linux systems. `https://github.com/mrbichel/RTIMULib`; 2016. [last accessed 27 March 2017].