https://doi.org/10.2352/ISSN.2470-1173.2017.6.MOBMU-299

Android Door/Window Image-based Measurements Application

Khader Mohammad, Ahmad Alsadeh, Amer Qarabsa, Shatha Khalil, Mona Dirieh; Birzeit University, Palestine

Abstract

Due to the huge growth in the digital image processing scope and its wide usage in many applications, find objecting dimensions and area remotely will have a great advantage in engineering, industry, and personal business usage. Therefore, this paper proposes an approach to find the object dimensions by using digital images. The system is built for automated image-based measurement using an android mobile application where a user takes an image of an object that could be door/window or any desired object to be measured. The system detects the image of objects to be processed, then it automatically finds the image scale using camera zoom property to measure the dimensions of the desired object. The proposed method applied and tested using images of different objects taken from different places with different features such as size, brightness and angle, The accuracy reached 100% for determining the object corner and dimensions of objects.The system reached 81% of detecting the objects due to noise and clarity issues. Also, the system shows an error of $\pm 0.9\%$ in dimensions measurements that differs according to the focal length of the camera and zoom.

Introduction

Intelligent devices and Systems have great potential to save time, lives, and improve our environment. It has considerable potential to be a future commercial success. These systems are also closely linked to other major emerging technologies; the internet, mobile data services, smart sensors, artificial intelligence, position technologies, geographical information systems (GIS). Obtaining measurements is one of the important fields for home and industry. This is due to the importance of the getting the needed data or measurements for different purposes in daily life. Visual language can be interpreted by the carpenters, modeling civil engineers, or home internal designers. Also, it is helpful for finding object dimensions, such as trees , car heights, nominal size and so on.

With the rapid spread of Smart phones and tablets among people from different ages, the dependency on these devices increases day after day. Thus, huge part of the IT industry focusing on creating applications to make human life easier. We propose an approach for finding the dimensions of objects with mobile application.

This research help for easing some of the difficulties people face in measuring objects' dimensions especially doors/windows or any other object in making decisions to replace, enhance or fix. Obviously having the measurement tools are costly. Even if these tools are available, it is hard to carry out an accurate measurements of these objects manually, especially in high or crowded places. Therefore, our application overcomes this problem by providing an easy, fast measurement tool to help people by making their work much easier, less time consuming, more efficient and more enjoyable by shorting a long exhausting task in simple process using smart devices which are almost available with everyone these days.

The proposed system includes three main components: object detection, color recognition and calibration. Object detection starts by detecting the objects in the image and since it is a door/window detection, it will use corner/edges algorithms to detect these objects. Color recognition defines the color regions in the input image. Color recognition can help in future to obtain 3D model. Calibration takes the output of the object detection as a corner image and processes it to retrieve some features which are used to measure dimensions of objects. Each component will work as a black box with the least coupling with the other component. The system requires from the user to take at least three images of the desired object with the surrounding area from different angles to start processing. Figure 1 shows top level system block diagram.

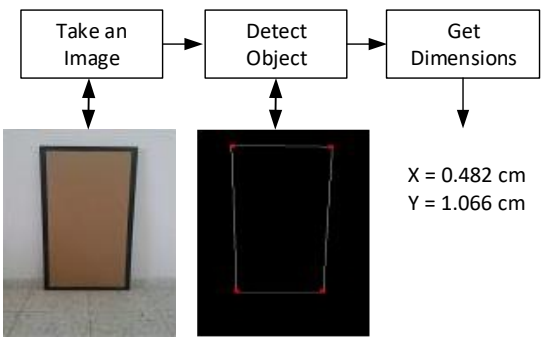

Figure 1: System Top level block diagram

Related Work

Some related work, which serves same goal are shown in this section. Some are overall systems which succeeded completely or partially in measuring the dimension of objects. Other methods and algorithms might be enhanced to serve our purpose.

An image-based application to measure door and/or window using an 8MP camera is presented in [4]. The system creates a 2D perspective plane to perform next a calibration process to measure the scale between the rectified images and the actual scale.

Another system used light indication spots to measure object dimensions in digital images [8]. They aim to find objects dimensions using digital images which are a binary matrix that consists of pixels. Resolution was measured according to the number of pixels per inch. Two green laser light sources are used for measuring object dimensions. The camera was set in a frame in the middle of the horizontal light line. They used canny edge detection and point search algorithm. First Canny has the property to show weak edges, which are related to strong edges. They also used

threshold limit of 0.6 which limits the kind of image to be used or tested in terms of brightens and color. Second, point search algorithm is used to find the center of two light spots on the object where a 2D image used as input to this algorithm. Using light indication spots to measure object dimension in digital images is depicted in Algorithm 1.

Algorithm 1 :Light indication spots algorithm

Input: Image

Output: Dimensions (Image)

1: Image=Gradient-green-Image (Image)
2: Threshold=0.6
3: Image=Image(Threshold)
4: 2D-Image=Canny (Image)
5: Thin(Image)
6: Point-Search (Image)
7: Image-light-distance(light-spot-center1, light-spot-center2)
8: Get-scale(Image)
9: Actual-scale+Image-light-distance/Scale

A smart Android application for measuring the user distance to a target object and its height is presented in [3]. This system is based on trigonometric approach, which calculates the distance of the object as shown in Figure 2 Step1 using equation 1. The object's height is measured using this equation 2. Initially, the angle θ in Step2 is measured automatically by the device's built-in sensors, which can be modified in the calibration stage. If the user stands on a higher land level (building), the building height must be known to be able to calculate distance and object height as shown in Figure 2-Step3.

$$AB = h \times \tan\theta \tag{1}$$

$$height = h + (AB \times \tan\theta) \tag{2}$$

This application is based on a simple concept and free to use. The accuracy was the main problem in this approach, since the user needs to modify the height manually. Also it has a maximum height that depends on the entered reference height which prevents the user from measuring some targets.

Harris corner detector [5] is a mathematical detector that looks for image features which makes it fast and simple to be used. It handles rotation, scale and for that it is known as the corners algorithms. This algorithm aims to find small windows that generate a large difference values when moved using equation 3. The goal is to find a window that produce a large E.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \tag{3}$$

where:
E: Difference between original and moved window.
u,v: Window's displacement.
$w(x,y)$: Window at (x,y).
I: Intensity of the image at (x,y).
$I(x+u,y+v)$: Moved window intensity.
$I(x,y)$: original intensity.

Another detector that we evaluate is Canny edge detector [2, 12]. Canny finds edges by looking up for gradient's local maxima in the image. The gradients are calculated using Gaussian filter derivatives. It uses two thresholds to detect strong and weak edges, where the weak edges appears in output only if they are connected to strong edges [9].

Table 1 presents a comparison between Sobel, Canny, and Laplacian edge detectors and presents advantages and disadvantages for each one. Canny edge detection algorithm shows the best performance, accuracy, and execution run time comparing with Sobel and Laplacian [10].

Table 1: Comparison current of Edge detectors

Edge detectors	Advantages	Disadvantages
Sobel	Simple	Sensitive to noise. Could be Inaccurate.
Canny	Improves signals using Non-maximum suppression method. Uses probability to find errors	Complex with Computations. Uses false zero crossing
Laplacian	Detects edges accurately. Do Tests for larger area around the tested pixel to easily find Edges.	Using Laplacian filter reduces the accuracy in finding edges Orientation.

Color constancy represents the ability of a visual system to recognize an objects true color across a range of variations of factors extrinsic to the object, such as light conditions [11]. This definition means that the purpose of color constancy algorithms is to generate illumination-independent descriptors of the scene colors measured in terms of the cameras RGB coordinates [13]. Determining the color will help in finding edges and for future use in creating 3D model for the object.

Two of the most popular algorithms, Zhang algorithm [13] and Tsai's camera calibration method [7] have been evaluated. Both mathematical approaches which extract the intrinsic and extrinsic parameters through mathematical operations. Tsai shows higher accuracy when having low measurement error, but this needs accurate 3D measurement which is not available in our case. In other hand, Zhang does not need this complexity, the error is decreased by taking more views of the image. Zhang algorithm is planar calibration approach which combines the advantages of both world-references based and auto-calibration approaches. The relationship between a 3D point $\tilde{M}$ and its image projection $\tilde{m}$ is given, where R,t are the extrinsic parameters, R is the rotation matrix, t is the translation vector as shown in equation 4.

$$s\tilde{m} = A[Rt]\tilde{M} \tag{4}$$

Intrinsic parameters represented in $A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, where:

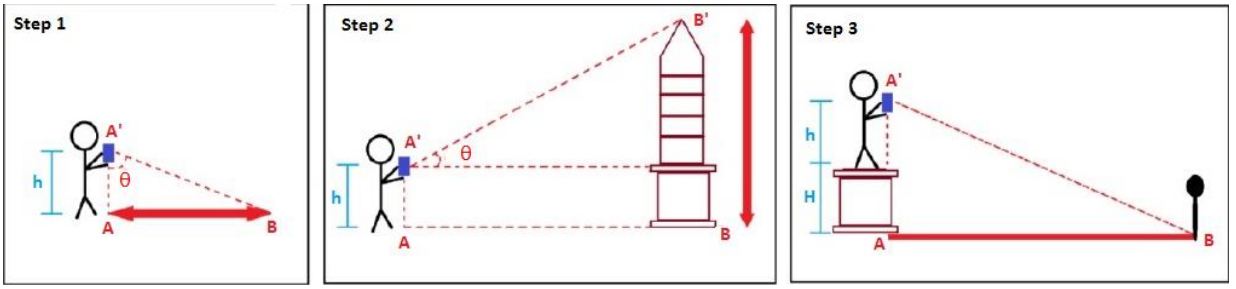

Figure 2: Smart measure application

$\alpha = f.mx$; mx is the horizontal scale factor of pixel to distance.
$\beta = f.my$; my is the vertical scale factor of pixel to distance and f is the focal length.
γ: skew coefficient between x and y, often = 0.
u_0, v_0; the principle point.

Model plane is on $Z = 0$ of the world coordinate system.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A\begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Since Z is always 0,
$\tilde{M} = [X, Y, 1]^T$
$s\tilde{m} = H\tilde{M}$ with $H = A\begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$
The $3X3$ matrix H is defined up to a scale factor.
Homograph is denoted as:
$\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda A\begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$
To summarize , the calibration approach goes through these steps:

1. Print a pattern and attach it to a planar surface;
2. Take a few images of the model plane under different orientations by moving either the plane or the camera;
3. Detect the feature points in the images;
4. Estimate the five intrinsic parameters and all the extrinsic parameters using the closed-form solution;
5. Estimate the coefficients of the radial distortion by solving the linear least-squares;
6. Refine all parameters by minimization.

Proposed Method

The proposed system goal is to achieve an intelligent and efficient tool for door and window measurements, preventing user mistakes, save time and resources. Since only a smart phone camera is needed, image-based measurement is delivered using an Android mobile application. The user takes an image of the desired object that could be door/window to be measured. The system detects the objects in the image to be processed, then it automatically finds the image scale to start measuring the vertical and the horizontal dimensions of object. This Application is directed to users need to measure objects specially doors and/or desired object that are either outdoor or indoor objects which can be used by carpenters.

The system top level diagram is shown in Figure 3, which consists of three main components, object detection, color recognition and calibration.

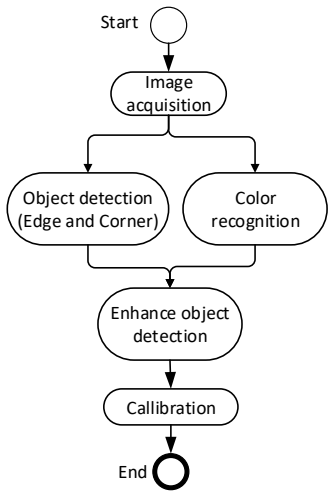

Figure 3: Top level System process

First, an image is captured for the desired object. The image should be clear (no image enhancement is done in this stage) and taken from the right angle. For calibration, at least three images should be taken to the same object from different angles.

The object is detected for the three images using Canny edge detector. This result will be passed to the Harris corner detector to produce corner images that will be enhanced by the color recognition output which, defines the color regions in the input image [1,7]. Then, the two enhanced output will be passed to the enhance object detection stage.

Now these corner images will be processed in calibration stage to acquire specific properties, intrinsic and extrinsic parameters, which will be used to measure the objects dimensions. Intrinsic parameters [6] are represented in matrix that contains 5 parameters, focal length x, y scales for image sensor format and x, y dimensions of a principle point. For extrinsic parameters it consists of a translation vector, which is the position of the origin of the world coordinates system expressed in the coordinates of the camera-centered coordinate system and the rotation matrix which represents the rotation of the objection in the x, y and z plane. By extracting these parameters, the system will be able to define the

scale between the real dimensions and the pixels in the image and calculate dimensions. Each component will work as a black box with least coupling with the other component.

Object Detection

Object detection is a process for identifying objects in image processing. It relies on matching, and learning algorithms, the object can be detected by edge detection or using corner detection. Corners are feature that are used to find the correspondence between images. There are five steps to show and define how object detection process is handled. A final result of this process is shown in Figure 4. Algorithm 2 shows the corresponding object detection process. The following summarizes what is performed in each step:

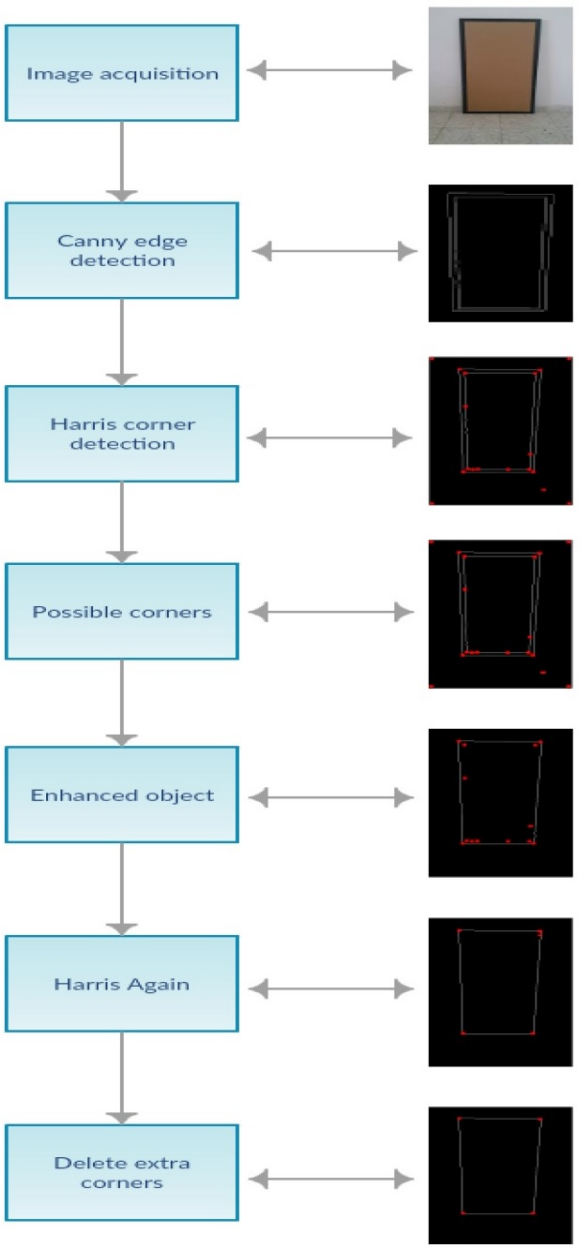

Figure 4: Object Detection process

Algorithm 2 : Object detection process algorithm.

Input: Original image containing the desired object

Output: Image contains only the edges of the object

```
ResizedImage=Resize(inputImage);
EdgeImage=Canny(resizedImage);
CornerList=Harris(EdgeImage);
for (corner:cornerList) do
    if (corner is on possible edge) then
        possibleCorners.add(corner);
    end if
end for
for (corner:possibleCorners) do
    removeCornerFromPossibleCorners()
    create Track()
    setCornerTotrack()
    while (true) do
        if (edge is continuous) then
            if (atEntryPoint) then
                break;
                addPPixelsToTrack()
            else
                if (track.isEmpty()) then
                    removeTrack()
                else
                    clearTrack()
                    goBackToPrviousTrack;
                end if
            end if
        end if
    end while
end for
if (!trackList.isEmpty()) then
    objectImageList.add(getImageFromTrackList(trackList))
end if
```

Step 1: Detect edges. In our proposed approach, edge detection is used to define boarders and edges. Canny edge detection algorithm used because of its precision in detecting edges. Started by building a Gaussian filter to smooth the image and calculate the gradients on each single point of the image (Looking for high gradient values). After that we used non-maximum suppression to compare the gradient magnitudes to the direction of the gradient to check if the value is a local maximum then it is an edge. The threshold is set to $2.5f$ for low-threshold and $7.5f$ for high-threshold based on experimental results for images we tested.

Step 2: Find corners. Starting by computing the derivatives and the gradients using Sobel gradient $3x3$, then we measured each pixel value and checked the threshold to be bigger than $1E-3$. Next, we considered only local maximum corners by adding them to a list we named it as a potential corner list. We check the list again, remove all close corner values and display the final corners.

Step 3: Possible corners. At each corner, we move with the connected edges and calculate the displacement in each direction (right, left, up, down). If one of these displacements is greater than the width of the picture/9 (the number 9 is selected based on experimental), then this is a possible corner. Otherwise, the

intended corner is not a possible corner and all the connected edge are removed so we do not have to check its siblings corners.

Step 4: Enhance object detection. After having all the possible corners, we loop over them moving with connected edges in four directions starting with default from down until reaching the next possible corner. A group of pixels is defined to be a track which starts at a possible corner and ends at another possible corner. For each pixel we pass passed, we delete it from the original image and then we add it to another new image. This process goes until we reach another possible corner. This track saved in a list and then we build a new track to start the same procedure again. This tracking continues until it comes close to the entry corner point with a distance of (width/40). Now if there is no pixels and a close point, we still have to check if we are coming closer to the entry point or if we are going farther (we stop if it is a wrong track). In case a dead-end (space) is reached, we check the area around. If there is a white point to continue from, then find a corner, and make a track from the space. We stopped and draw a line of pixels for the corner. If we does not find a white point (edge pixel), then we check this track if it is clear or not. If not, we empty it. If it is empty, we go back to its corner point; we delete and clear the track before it. If we are close to the entry point, we search around. If we do not find anything then we are done. However, still we are not sure if we detected the right/ valid object.

Step 5: Validate object. Check the corners in the list by taking the $max(x)$, $min(x)$, $max(y)$, and $min(y)$, where x and y represents the with and the height respectively. We check the difference between $max(x)$ and $min(x)$, $max(y)$ and $min(y)$ of the object, if they are less than $(width/7)$, then this is not a valid object.

Calibration

In order to get accurate dimensions, we need at least three images of the same object captured from different angles to present them on a pattern image. One image is the pattern image with its dimensions will be used to calculate the actual dimensions. And other images representing the desired object from different angles which we will call angle images. After getting the object detected, the calibration component starts applying Harris algorithm on the images of the object which returns several points. In fact, not all of these points are the actual corners so they need to be enhanced.

Harris algorithm returns different corners for each image because of the object position in each image. We get the point's coordinates of each corner in all images to build a list of pairs for each image. Each pair contains one corner from the angle image. In order to enhance the corners we used two cases as shown in algorithm 3:

Each angle image is compared with the pattern image, if the angle image has the same number of corners then it will be used for the calibration, if not it will be discarded.

Initially the corners will be ordered depends on the X-coordinate in descending order. If the horizontal distance between set of corners is less than a specified threshold, corners are considered to be vertically aligned and there is a need to check their order which could be corrupted due to difference of acquisition angels, otherwise the corners have the right order.

In order to check the order for two corners, we compare the change in Y-coordinates in the current angle image with the change in the pattern image, if it has the same sign then the order is correct and we proceed to the next corner.

Algorithm 3 : Corner determination

```
if (the point is a real corner) then
    check the previous and next points
    if (the same slope) then
        then it is a line, it will be discarded.
    end if
    if (different slope) then
        this is a corner will be considered.
    end if
end if
if (points in the same corner) then
    take the average.
end if
```

In case we have three points above each other, we check the first and second points, if the order is correct, we move to check the second with the third, if the order is violated we swap the second and third corner and go back to check order from the beginning to make sure our change did not corrupt the order of the previous corners.

The X and Y focal length values obtained from calibration are used to calculate the scale of X and Y coordinates. So we calculate the real length of each pixel of the object by dividing the focal length of the camera over the focal length of the X coordinate to get the scale-X, and by dividing the focal length of the camera over the focal length of the Y coordinate to get the scale-Y. Multiplying scale-X and scale-Y with the object's width and height provides the real dimensions.

Our system has the ability to re-size images in order to reduce time and achieve good performance with big images. It works by comparing which is larger the width or the height, then divide the bigger one by 600 which will left us with another scale of the input image. In case of changing the image width or height of course the calibration resulting values will change and this issue is solved by divide each of calculated scales (scale-X and scale-Y) by the same number (600) to get the dimension in *cm*. Note that if the image was scaled the dimensions calculated will be divided by the scale rate. The system calibration process is shown in Figure 5.

Test and Evaluation

In order to test and evaluate our application we collected a dataset which is a collection of doors and/or windows and other objects from different positions and angles. The data-set contains images with different features such as high/low brightness, distance from the desired object and high/width for each object. The dataset contains 180 images from different places.

To evaluate the performance of our system, 123 images are tested. Images were taken from different places using different cameras. During the evaluation, we did test two parts, one for testing the detection of the objects and another part testing the calibration and dimension results.

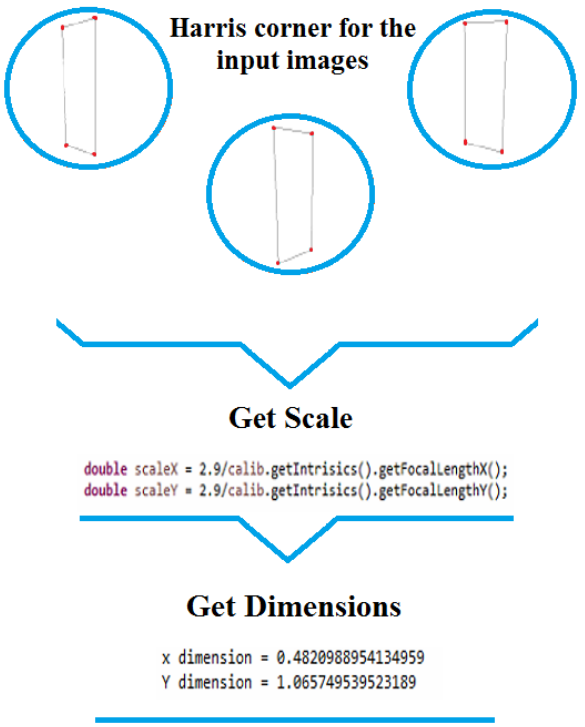

Figure 5: Calibration process.

Object detection

Example images of object detection are shown in two Figures 6 and 7. Figure 6 for 100% correct results of detecting the object. Figure 7 with corner error due to false in edge detection of object.

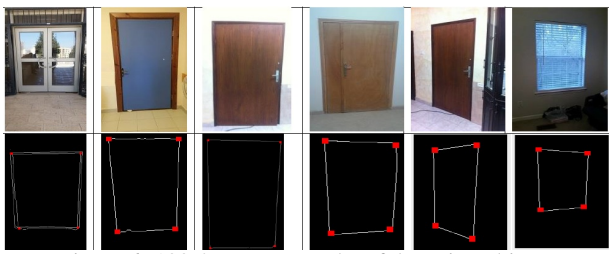

Figure 6: 100% correct Results of detecting objects

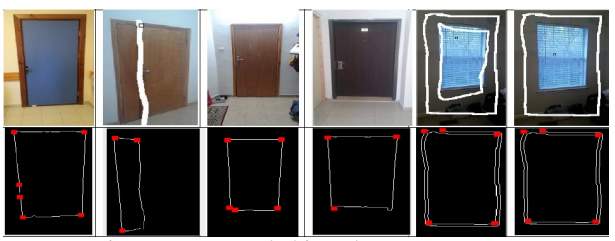

Figure 7: Detected objects have corner error

The final Object detection evaluation results of 123 images is summarized in Table 2. The running time for object detection process depends on the object size. Running time for all tested images is ranged between 1.12–2.75 seconds.

Table 2: Final object detection results for 123 tested images

Tested Images	Samples	Rate
Correct 100%	65	52.85
Corner Error	16	13.01
Error due wrong angle capture	18	14.63
Error due false in edge detection	24	19.51

Calibration (Dimension)

Examples for calibration and getting objects dimensions are shown in Figure 8. Evaluation for final result of the calibration process is shown in Table 3. The running time depends on the object size and number of entered images for the same object. The running time for all tested images is between 15.80–23.85 seconds.

Table 3: Examples from the calibration process evaluation

Tested Images in Figure 8	Focal length	Real dimension	Calculated dimension	Time (sec)
Object 1	3.5	$x = 1.27$ $y = 0.56$	$x = 1.12$ $y = 0.44$	23.85
Object 2	3.5	$x = 0.51$ $y = 0.69$	$x = 0.67$ $y = 0.55$	17.37
Object 3	2.9	$x = 0.45$ $y = 1.04$	$x = 0.47$ $y = 1.04$	15.84

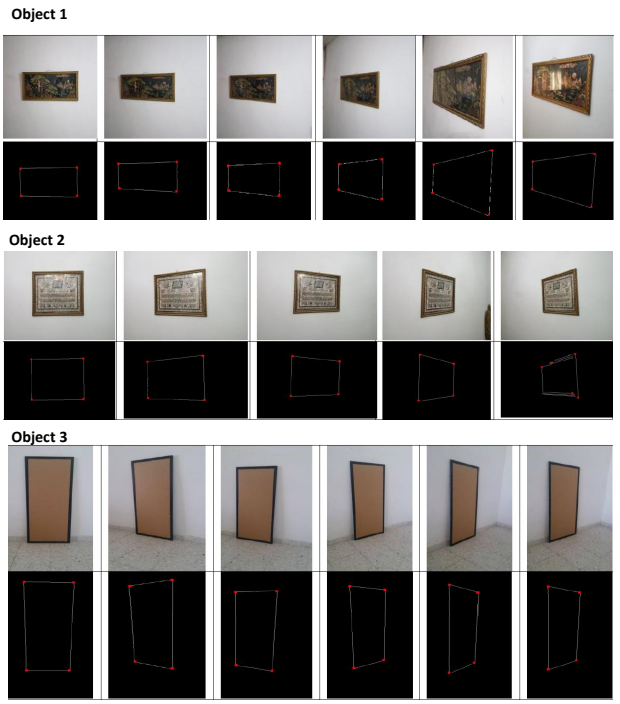

Figure 8: Tested images for object 1,2, and 3.

Conclusion

A final Android application is developed that has the ability to detect objects and find their dimensions from either captured images or selected from gallery. Since no complex setup is required, the proposed system is considered a practical solution with good performance to achieve user's goal. The system is integrated as an Android application which is an open source, along with open source libraries, it will be available for all android users. This application will be also modified to have the ability to measure dimensions of any desired object not only door/window objects.

Images can be of different brightness and color densities which makes it difficult to determine the required objects. The application uses the phone camera to take an image or the user can directly choose from gallery any desired image to detect the object and find its dimensions. The programming language used is JAVA as it used for Android application. As future version of this work will be built on IOS platform to serve IOS users together with Android users. A study of 3D modeling to model different doors/windows models in the system as a replacement option of the selected object and an enhancement modification will be done to object detection results using color recognition process. Also a zoom feature of the image along with camera properties will be added to increase the accuracy of the system.

References

[1] N. Bhatia and M. Chhabra. Accurate corner detection methods using two step approach. *Global Journal of Computer Science and Technology*, 11(6):25–30, April 2011.

[2] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–69, June 1986.

[3] Google. Smart measure. https://play.google.com/store/apps/details?id=kr.sira.measure&hl=en, September 2016.

[4] M. Guangyao, J. Manishankar, and A. Gady. Door and window image-based measurement using a mobile device. In *SPIE/IS&T Electronic Imaging*, volume 9411, pages 94110A–94110A. International Society for Optics and Photonics, 2015.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, 1988.

[6] A. Henrik, J. Klas, A. Francois, B. J. Andreas, and K. Fredrik. Camera resectioning from a box. In A.-B. Salberg, J. Y. Hardeberg, and R. Jenssen, editors, *Lecture Notes in Computer Science*, volume 5575 of *Lecture Notes in Computer Science*, pages 259–268. Springer Berlin Heidelberg, 2009.

[7] B. K. Horn. Tsai's camera calibration method revisited. *Online: http://people. csail. mit. edu/bkph/articles/Tsai_Revisited. pdf*, pages 1–13, 2000.

[8] S. K. Ibrahim. The use of lighted point in finding the body dimensions in digital images. *Rafidain Journal of Computer Sciences and Mathematics*, 7(3):31–46, Nov. 2010.

[9] M. Juneja and P. S. Sandhu. Performance evaluation of edge detection techniques for images in spatial domain. *international journal of computer theory and Engineering*, 1(5):614, Dec 2009.

[10] G. Shrivakshan, C. Chandrasekar, et al. A comparison of various edge detection techniques used in image processing. *IJCSI International Journal of Computer Science Issues*, 9(5):272–276, 2012.

[11] M. Sridharan and P. Stone. Towards on-board color constancy on mobile robots. In *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference on*, pages 130–137. IEEE, 2004.

[12] O. Vincent and O. Folorunso. A descriptive algorithm for sobel image edge detection. In *Proceedings of Informing Science & IT Education Conference (InSITE)*, volume 40, pages 97–107, June 2009.

[13] Z. Zhengyou. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.