Remote Distance Measurement from a Single Image by Automatic Detection and Perspective Correction

Md Abu Layek[1], TaeChoong Chung[1] and Eui-Nam Huh[1*]
[1] Department of Computer Science and Engineering,
Kyung Hee University Global Campus
Gyeonggi-do, Yongin-Si, 17104, Korea
[e-mail: {layek, tcchung, johnhuh}@khu.ac.kr]
*Corresponding author: Eui-Nam Huh

Received October 9, 2018; revised November 26, 2018; revised January 9, 2019; accepted February 9, 2019; published August 31, 2019

Abstract

This paper proposes a novel method for locating objects in real space from a single remote image and measuring actual distances between them by automatic detection and perspective transformation. The dimensions of the real space are known in advance. First, the corner points of the interested region are detected from an image using deep learning. Then, based on the corner points, the region of interest (ROI) is extracted and made proportional to real space by applying warp-perspective transformation. Finally, the objects are detected and mapped to the real-world location. Removing distortion from the image using camera calibration improves the accuracy in most of the cases. The deep learning framework Darknet is used for detection, and necessary modifications are made to integrate perspective transformation, camera calibration, un-distortion, etc. Experiments are performed with two types of cameras, one with barrel and the other with pincushion distortions. The results show that the difference between calculated distances and measured on real space with measurement tapes are very small; approximately 1 cm on an average. Furthermore, automatic corner detection allows the system to be used with any type of camera that has a fixed pose or in motion; using more points significantly enhances the accuracy of real-world mapping even without camera calibration. Perspective transformation also increases the object detection efficiency by making unified sizes of all objects.

Keywords: Object detection, Object locating, Deep learning; Real-world mapping, Remote measurement

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2013-1-00717) supervised by the IITP (Institute for Information \& communications Technology Promotion).

1. Introduction

Since the last few years we have been witnessed the huge development in the computer vision field and currently powered by deep learning it has been accelerating even more. The high-level target of Computer vision is to gain human-level understanding from digital images or videos. Because it involves understanding, computer vision makes use of machine-learning technologies hugely along with the primitive image and video processing tasks. Especially, these days deep learning becomes the most dominant technology for computer vision [1]. Deep Neural Networks are also a good candidate for image processing jobs [2]. Darknet [3] provides a faster deep learning framework implemented in C language. Natively, the popular real-time object detection system YOLO [4] uses the Darknet. YOLO is very fast as compared to the previous object detection techniques because it takes the detection task as a single regression problem and uses a single network for the whole detection [5], [6]. The active development of YOLO results in continuous improvement [5], [7] and it is extending to 3D [8]. Moreover, Several other popular Convolution Neural Network (CNN) architectures such as DenseNet [9] and ResNet [10] can also be used and combined with YOLO[6], [11], [12]. This article uses YOLOv2 [7].

In many situations, only detection of an object within an image or video is not enough rather location of the object in real space is needed, for example, it enables us to measure the distance between objects which can further be utilized in alarm, robot vision and many other services. However, pixel position representing an object on an image greatly varies depending on the camera position, angle, as well as quality of the camera thus we need to consider these things to get real locations of objects from an image. Camera calibration [13] can find out the intrinsic and extrinsic camera parameters along with the distortion vectors such as Radial and Tangential. Using this information we can remove some of the distortions from the images taken by that camera. Calibration also helps us locating an object in real world but in such case either we need to keep both camera and the interested region fixed, or placing a check-board like object in the Field of View (FoV) [14] which is not always easy. Additionally, if our interested region surrounds many unwanted things within the FoV then the chance for misdetection increases. To overcome the problem, before detection we need to find our region of interest (ROI).

Optical distance measurement techniques primarily are of two kinds, contact or non-contact. Most of the early classical methods are based on laser or ultrasonic sensors even with a physical ruler attached in special arrangements [13]–[15]. On the contrary, vision-based methods rely only on images captured by cameras which is very convenient and flexible. Several works described about measuring real-world distance from images and most of them use stereo camera or dual camera placed on a known distance [18], [19] as well as with single motionless camera [13], [20]. Also, several methods are based on special types of cameras such as CCD or depth [15], [21]. Other approaches [18], [20]–[22] to measure distance between camera and an object use one or more reference objects in the FoV, Chan et al.'s [25] car to car distance measuring system is actually a variation of this category. In contrast, our method deals with locating and measuring distances between objects within ROI. Raza et al. [26] proposed an image based framework for estimating distance and dimensions of pedestrians which exhibits very good results. Their method used a fixed arrangement with a motionless camera where the boundary marks are predefined. Mobile Robot Localization Systems discussed in [27], [28] use simple background subtraction methods for object detection and parallel distance measurement and morphological processing respectively for measuring distances. However, It is possible to get better results using YOLO object detection along with perspective transformation.

In this work, we use a single image frame from a single camera and through automatic corner detection, this approach can work well even if the camera position changes. Although in this work change of ROI orientation is not considered, it can be solved by using varying color corner points. After the corner detection phase, the orientation can be adjusted as desired. This method also does not depend on the type of camera used to capture the image. The proposed system can determine an object's location in real space from which distance and motion measurement, tracking etc. can be done. The region of interest (ROI) is determined with some predefined marks and any existing objects within the FoV can be trained as marks using deep learning. The relative positions of the marks are known prior to their detection. After corner detection, the image is cropped to the ROI and applied warp perspective transformation to make a new image with the same aspect ratio of the real space. Then, object detection is run on this normalized image and we can easily determine the real locations of the objects. However, the performance greatly depends on the number of marks for defining the boundary (also referred to as corner points in this paper) as well as the accuracy of their detection. Removing the distortion before detection decreases the error and if we take more boundary marks for defining the ROI, the accuracy improves even without the camera calibration. From the experiments, we see that perspective transformation also improves the detection performance apart from the real-world mapping.

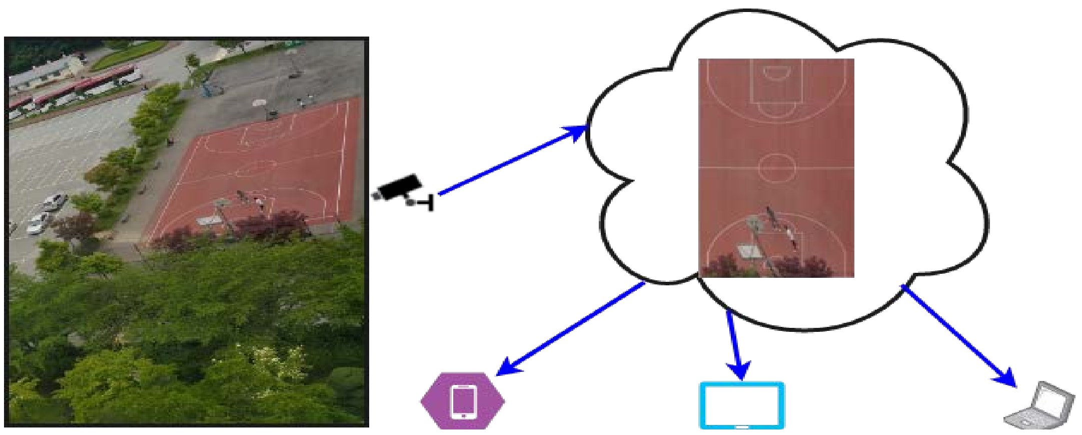


Fig. 1. Example application scenario using the proposed approach

Fig. 1 shows a simple application scenario employing the proposed approach. After capturing the scene, the camera sends an image to the server; the server performs transformation, detection and other processing then shares measurement information with all of the connected devices. If the camera unit is equipped with enough capacity, then the processing can be done immediately and send measurement information to the server.

We can summarize the contributions and importance of this work as below:

i.) The proposed approach detects objects using ‘corner detection - perspective transformation - object detection’ pipeline which normalizes the size of the objects inside ROI and improves the performance of detection. In our experiments, we found that the objects which remain undetected in their original size can be detected after the transformation.

ii.) We propose a novel integrated distance measurement method from a single camera image combining perspective transformation and object detection. Using YOLO architecture makes the detection faster and suitable for real-time application.

iii.) We modified the Darknet framework by integrating camera calibration, perspective transformation, un-distortion; and other necessary adjustments are also made.

iv.) We demonstrate the effect of the number of boundary points on distance measurement accuracy. Experiments show that using 9 corner points with the original image gives a similar result as we get using camera calibration and undistorted image. Also, for any fixed camera arrangement, the corner points need to be detected only once, if camera position changes then another detection will be needed and so on.

The rest of the paper is organized as follows. Section 2 describes some underlying theories and related techniques. Sections 3 and 4 discuss the experimental arrangements and scenarios whereas the results are shown in Section 5. Finally, we conclude our paper in section 6.

2. Background

In this section, we briefly review all underlying theories that the content of this paper relies on: Camera Calibration, Perspective Transformation and Object detection with Darknet YOLO.

2.1 Camera Calibration

Camera calibration is the process of estimating the parameters of a lens and image sensor of an image or video camera. The **Intrinsic** parameters are inherent to specific camera hardware and include **focal length**, the **optical center,** and the **skew coefficients**. It is usually expressed as a 3×3 camera matrix (1). The parameters of equation (1) are described in **Table 1**. The **extrinsic** parameters correspond to rotation and translation vectors $\begin{bmatrix} R \\ t \end{bmatrix}$.

There are two additional distortion parameters: **Radial distortion** causes straight lines to be appeared curved. It occurs when light rays bend more near the edges than at its optical center (**Fig. 2**). The smaller the lens, the greater the distortion is. The distortion can be positive (barrel) or negative (pincushion), with 2 coefficients it is represented as in equation (2). **Tangential distortion** occurs if the image taking lens is not parallel to the image plane and represented as in equation (3). The distortion coefficients of (2) and (3) are represented as a single vector (4).

$$CM = \begin{bmatrix} f_x & 0 & c_x \\ S & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Table 1. Intrinsic parameters of a camera

Parameter	**Description**
(cx, cy)	Optical center in pixels
F	Focal length in world units
$[px, py]$	Size of the pixel in world units.
(fx, fy)	Focal length in pixels, $fx = F/px, fy = F/py$
S	Skew coefficient, $S = fy \tan\alpha$

$$\begin{aligned} x_{distorted} &= x(1 + k_1 r^2 + k_2 r^4) \\ y_{distorted} &= y(1 + k_1 r^2 + k_2 r^4) \end{aligned} \tag{2}$$

$$\begin{aligned} x_{distorted} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{distorted} &= y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{aligned} \tag{3}$$

where, $r^2 = x^2 + y^2$; x and y are the undistortrd pixel locations in normalized image coordinates.

$$DV = [k_1, k_2, p_1, p_2] \tag{4}$$

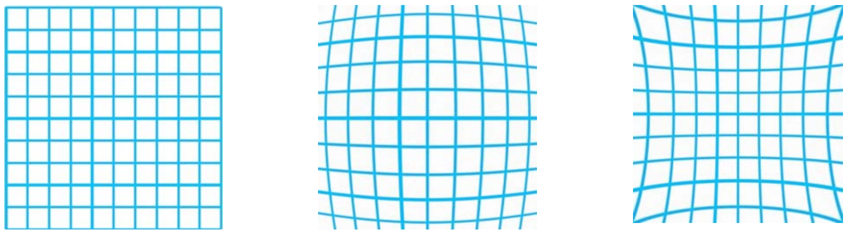


Fig. 2. Barrel and Pincushion distortions

2.2 Perspective Transformation

For perspective transformation of the ROI, we use the $WarpPerspective$ function built in the OpenCV C library and declared as in equation (5). The function transforms the source image using a specified 3×3 matrix M (6), the parameters of the function is described in **Table 2**. The 3×3 transformation matrix M can be estimated using another built-in function $getPerspectiveTransform$ declared as in equation (7), where src and dst are four corresponding points in the source and destination images respectively. We modified Darknet to integrate $WarpPerspective$ along with the $getPerspectiveTransform$ function.

$$warpPerspective(src, dst, M, flags, fillval) \tag{5}$$

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{6}$$

$$getPerspectiveTransform(src, dst, M) \tag{7}$$

Table 2. Parameters of a warpPerspective function

src -	Input Image
dst -	Output Image
M -	3×3 transformation matrix
dsize -	Size of the output image
flags -	combination of interpolation methods
fillval -	pixel extraoplation method

2.3 Object detection with Darknet YOLO

As mentioned in the introduction, YOLO is a powerful tool for real-time object detection. In the training process of YOLO, the objects are marked as bounding boxes and stored in an XML file for each image where every objects are described with four spacial parameters {X$_{min}$,

Y_{min}, X_{max}, Y_{max}}specifying object's location inside the image. There are various marking tools and in this work we used LabelImg [29]. However, Darknet takes a .txt file for each image with a line for each ground truth object in the image as {<*object-class*> <*x*> <*y*> <*width*> <*height*>}. We used a python script to generate those text files from the xml files.

Detections are also performed as bounding boxes. Every object's position in the image is specified by a four element vector $[Left, Right, Top, Bottom]$ where $Left$, $Right$ are X-values and Top, $Bottom$ are Y-values. We modified Darknet to store these position vectors in a file and we determine the middle position of an object by the tuple $\{\frac{Left+Right}{2}, \frac{Top+Bottom}{2}\}$ which is used as the one point determiner for the object position, however, for corners, we calculate differently.

Table 3. X, Y values for boundary marks

Corners	(X,Y)	Point on Real-Space Size = 2070 × 1710 (mm)	Point onTrans. Image Size = 1024 × 846 (px)
Top-Left	(L, T)	(0,0)	(0,0)
Top-Middle	$(\frac{L+R}{2}, T)$	(1035,0)	(512,0)
Top-Right	(R, T)	(2070,0)	(1024,0)
Middle-Left	$(L, \frac{T+B}{2})$	(0,855)	(0,423)
Middle	$(\frac{L+R}{2}, \frac{T+B}{2})$	(1035,855)	(512,423)
Middle-Right	$(R, \frac{T+B}{2})$	(2070,855)	(1024,423)
Bottom-Left	(Left, B)	(0,1710)	(0,846)
Bottom-Middle	$(\frac{L+R}{2}, B)$	(1035,1710)	(512,846)
Bottom-Right	(R, B)	(2070,1710)	(1024,846)
L=Left	**R=Right**	**T=Top**	**B=Bottom**

Table 3 describes corner points, the corresponding (X, Y) values for 9 boundary marks, the real-space points and the transformed image points used in our experiments. The calculated (X, Y) values are used as the source points for perspective transformation. Details of YOLO can be found on the Darknet website [3].

In darknet, training process takes a configuration file describing the neural network architecture and we used a slightly modified version of the default YOLOv2 configuration as depicted in **Fig. 3**. There are 23 convolutional layers, the initial image is resized to 512 × 512 and the final feature image size before classification is made to 16 × 16 with a combination of 3 × 3 , 1 × 1 filters, maxpool, reorg and route layers. Some related topics on the training and CNN artitecture are discussed below:

Grid cell: YOLO divides an image into $S \times S$ grid cells and each cell predicts a fixed number of bounding boxes. In our case, we use 16 × 16 grid cells whereas every grid predicts bounding boxes with different aspect ratios based on the number of anchors, in our network we use 10 anchors. We keep 20 object classes as in PASCAL-VOC, although only three labels (Cart1, Cart2, and Robot) are provided during object detection trainng and one label (Corner) during corner detection training. The bounding box prediction has 5 components: (*x*, *y*, *w*, *h*, *confidence*). The (x, y) represents the center of the box, relative to the grid cell location. These coordinates are normalized to fall between 0 and 1. The (*w*, *h*) box dimensions are also normalized to [0, 1], relative to the image size. A bounding box is removed if it has no object or confidence score is less than a threshold. On the other hand, if a bounding box contains an object with sufficient confidence then redundancy of identifying the same object is removed using Non Max Suppression and Intersection over Union (IOU).

Batch Normalization: Batch normalization (BN) normalizes the value distribution before going into the next layer. From YOLOv2 batch normalization (BN) han been introduced which improves the convergence without other regularizations such as dropout.

Convolution with anchor boxes: Original YOLO predicts the bounding box coordinates with the fully connected layers, however, inspired with the Faster R-CNN [30] YOLOv2 introduced anchor boxes. It simplifies the learning process because now only offset predictions are enough instead of predicting coordinates. YOLOv2 shrink the input image from 448 images to 416×416 to make the output feature size 13×13 so that there be a single center cell. They argued that larger objects tend to reside in the center and single-center feature grid makes the detection faster. However, in our case Robots, Carts and Corners are not too large with respect to the ROI and we found that 512×512 input images with output feature size 16×16 give a better prediction. Moreover, YOLOv2 has Multi-Scale Training which enables it to work on a range of dimensions from 320 to 608 in a multiple of 32. As a result, our initial choice of 512×512 does not create any inconsistency.

Fine-Grained Features: Objects in our experiments are relatively smaller and the high-resolution feature size 16×16 helps in this regard. The feature map is further improved by adding a passthrough layer (route layer) to bring features from an earlier layer at 32×32 resolution.

Loss Function: The loss function is a multipart function and for a grid, cell pair (*i*,*j*), it can be defined as in equations (8), (9). The first part indicates bounding box parameter loss, the second is the class prediction loss whereas the third indicates the confidence score loss.

$$\begin{aligned}
Loss_{ij} &= Loss_{ij}^{xywh} + Loss_{ij}^{p} + Loss_{ij}^{c} \\
Loss_{ij}^{xywh} &= \frac{\lambda_{coord}}{N_{L_{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \Big[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2 + \\
&\qquad \left(\sqrt{w}_{ij} - \sqrt{\hat{w}}_{ij}\right)^2 + \left(\sqrt{h}_{ij} - \sqrt{\hat{h}}_{ij}\right)^2 \Big] \\
Loss_{ij}^{p} &= -\frac{\lambda_{\text{class}}}{N_{L_{obj}}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \sum_{c \in \text{class}} p_{ij}^{c} \log(\hat{p}_{ij}^{c}) \\
Loss_{ij}^{c} &= \frac{\lambda_{\text{obj}}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left(IOU_{\text{pred}_{ij}}^{\text{truth}_{ij}} - \widehat{C}_{ij} \right)^2 \\
&\quad + \frac{\lambda_{noobj}}{N^{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{noobj}} \left(0 - \widehat{C}_{ij} \right) \\
&\text{Where,} \\
N_{L^{obj}} &= \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \\
N^{conf} &= \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} + L_{ij}^{\text{noobj}} (1 - L_{ij}^{\text{obj}})
\end{aligned} \tag{8}$$

The predicted values $pred_{ij} = (\hat{x}_{ij}, \hat{y}_{ij}, \widehat{w}_{ij}, \hat{h}_{ij})$ and the ground truth values $truth_{ij} = (x_{ij}, y_{ij}, w_{ij}, h_{ij})$; λ_{coord}, λ_{class}, λ_{obj}, and λ_{noobj} are scalar weights. L_{ij}^{noobj} and L_{ij}^{obj} are 0/1 indicator such that:

$$
\begin{aligned}
L_{ij}^{\text{obj}} &= \begin{cases} 1 & \text{if } C_{ij} = 1 \\ 0 & \text{else} \end{cases} \\
L_{ij}^{\text{noobj}} &= \begin{cases} 1 & \text{if } \max_{i'j'} \ IOU_{\text{pred}_{ij}}^{\text{truth}_{i'j'}} < 0.6 \text{ and } C_{ij} = 0 \\ 0 & \text{else} \end{cases}
\end{aligned}
\tag{9}
$$

Dataset and training: Our training dataset have 1000 images captured from different directions while running the carts and robots, then the corners and objects are annotated using LabelImg. We converted the *Darknet*19 448 × 448 pre-trained weight file to use as initial convolutional weights. Although not fully converged, after 70,000 iterations there was no noticeable improvement, as a result, the weight file after 80,000 iterations was used. Training for boundary marks (Corners) and the objects (Robots and Carts) are performed separately and we get two separate weight files (**Fig. 5** and **6**).

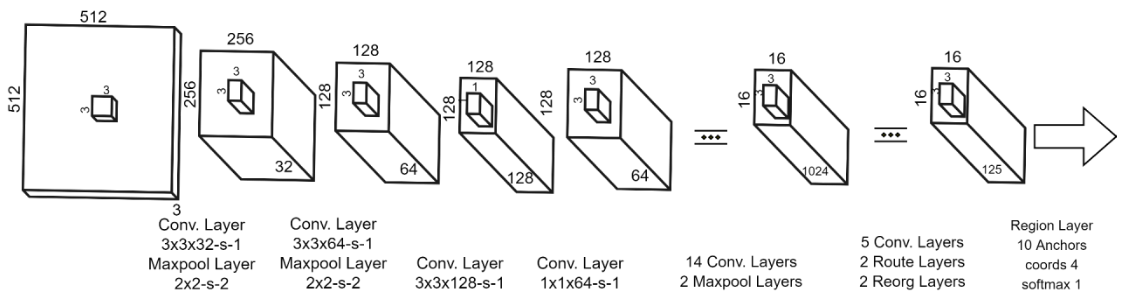
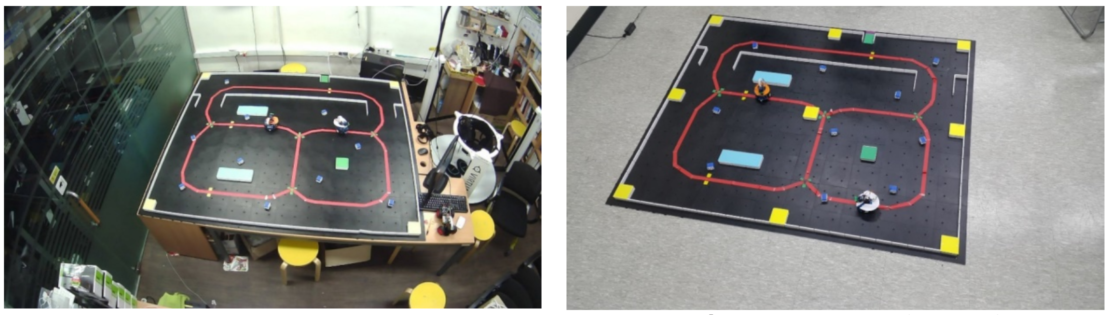


Fig. 3. The deep learning pipeline used in this article

3. Experiment Setup

In our experiment, the ROI is a board and the objects inside are ten Hamster Robots and two AlphaBot2s as shown in **Fig. 4**, the left image (**4a**) is taken by one camera and the right one (**4b**) with another. We covered the Hamsters with blue papers; one of the Alpha Bots with white paper (Cart1) and the other with orange paper (Cart2) as shown in **Fig. 5**.



a. Arrangement1 (Camera1)

b. Arrangement2 (Camera2)

Fig. 4. Experimental Arrangements with two different cameras

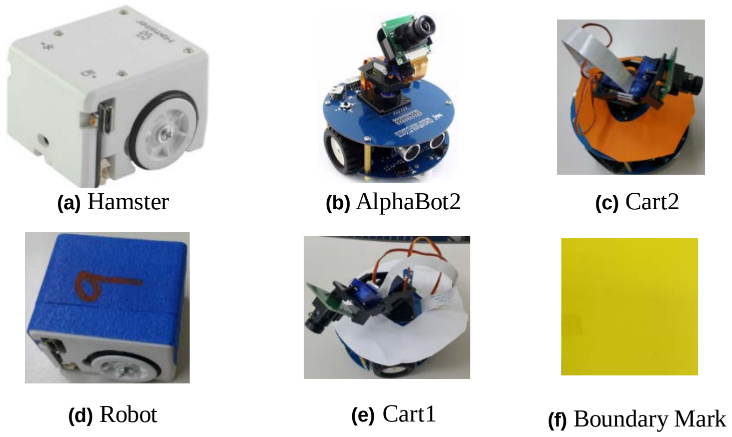


Fig. 5. Objects and marks used in the experiments

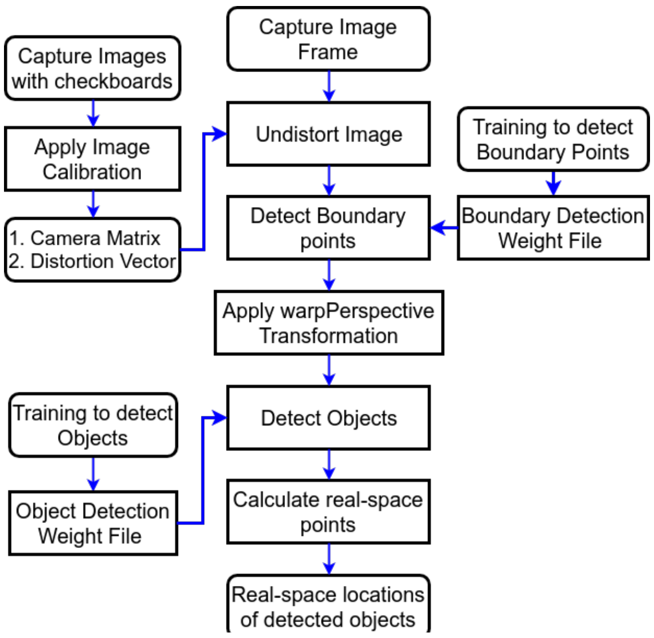

Capture Images with checkboards
Apply Image Calibration
1. Camera Matrix
2. Distortion Vector
Capture Image Frame
Undistort Image
Detect Boundary points
Training to detect Boundary Points
Boundary Detection Weight File
Apply warpPerspective Transformation
Training to detect Objects
Object Detection Weight File
Detect Objects
Calculate real-space points
Real-space locations of detected objects

Fig. 6. Flow diagram for the experimental system

Fig. 6 shows the steps of our proposed system. Images are captured with two kinds of cameras, a 2-megapixel webcam (model- ELP-USBFHD05MT-DL36) and a Galaxy Note Edge mobile camera. As discussed in section 2, as an optional step we can undistort the image taken by a specific camera if we know the camera matrix preferably with the distortion vectors. To get these parameters we take checkboard images captured by the same camera (**Fig. 7**) and apply the calibration process discussed in section 2.1. The calibration process on the checkboard images yields camera matrix along with the distortion coefficients**. Fig.s 4a** and **4b** are taken with two different cameras and as such both have different internal parameters and distortions. The camera matrices of equations (10) and (12) corresponds to equation (1) and the four coefficient distortion vectors of equations (11) and (13) corresponds to (4) and are based on the discussion of section 2.1. From the images, it can be observed that camera1 have barrel and camera2 have small pincushion distortions. After removing the distortion we get the undistorted versions as shown in **Fig. 8**.

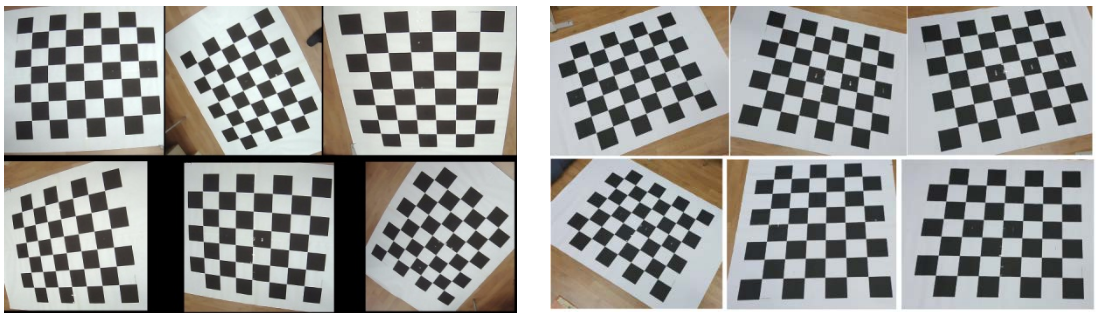


Fig. 7. Checkboard Images taken for calibration

$$CM1 = \begin{bmatrix} 1490.30 & 0.0 & 880.53 \\ -149.76 & 1638.6 & 366.59 \\ 0 & 0 & 1.0 \end{bmatrix} \tag{10}$$

$$DV1 = [-0.68 \quad 0.49 \quad 0.066 \quad 0.039] \tag{11}$$

$$CM2 = \begin{bmatrix} 1100.10 & 0.0 & 1002.10 \\ 3.86 & 1065.81 & 892.23 \\ 0 & 0 & 1.0 \end{bmatrix} \tag{12}$$

$$DV2 = [0.048 \quad 0.016 \quad 0.020 \quad 0.00035] \tag{13}$$

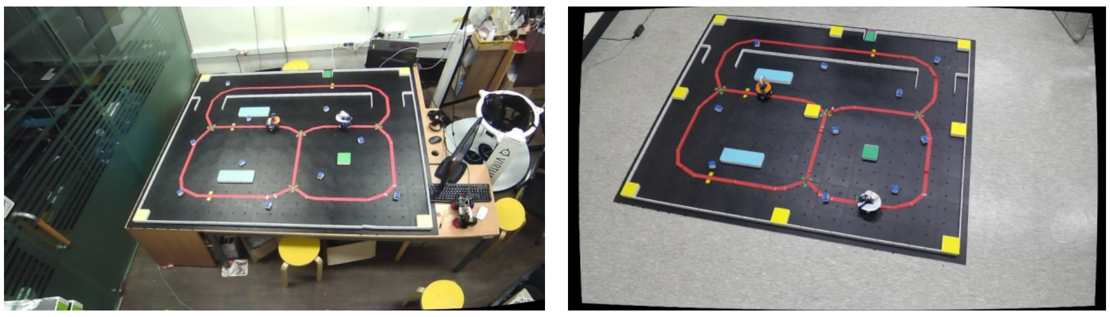


Fig. 8. Undistorted Version of the Arrangements

Corner detection is then applied using the weight file trained with corner labeled images and the corner points are obtained using **Table 3**. Then, as per the techniques discussed in section 2.2, we apply the warpPerspective transformation to crop through the corner points and transform to a new image with same aspect ratio as in the real space. When 9 points are used, the whole image is divided into 4 blocks and transformation is applied on each block separately, details are discussed in section 4.

On this transformed image, another step of detection is applied to detect the objects using the other weight file trained with object labeled images. Finally, we get the object positions which are in the same aspect ratio with real-space and we can estimate the real distances easily.

Along with the exact sequence of steps we considered in **Fig. 6**, some other combinations are taken in the experiments to understand the effects of each process and section 4 describes the cases of these scenarios.

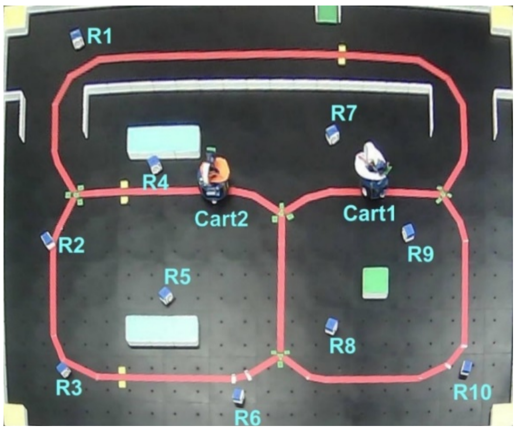
4. Description of Experiment Scenarios

For each arrangement, we consider the following cases for comparison:

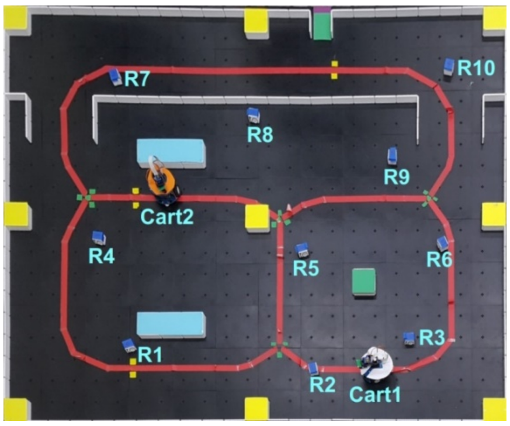
- Manual object detection using manually selected corner points with original and undistorted images.
- Object detection by the system for both original and undistorted images.
    - Using manual corner points
    - Using detected corner points

In arrangement2 (Camera2), we set nine boundary marks where we use both four and nine corners and taking all these we perform object detection on 18 different combinations for both arrangements.

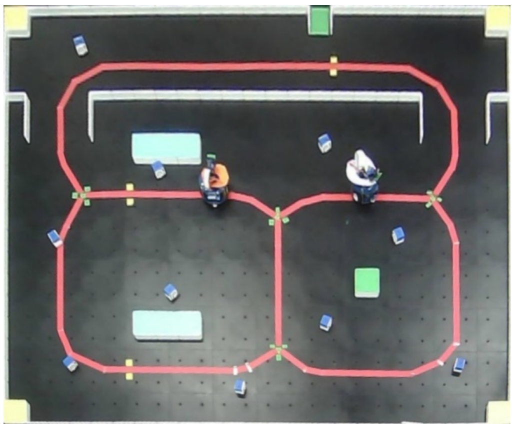
4.1 Case1: Manual detections



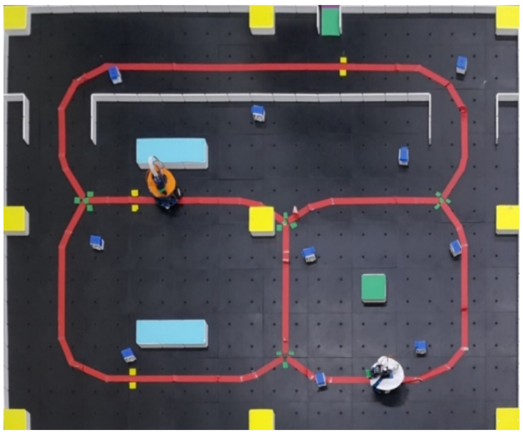
(a) Camera1 Original



(b) Camera2 Original



(c) Camera1 Undistorted



(d) Camera2 Undistorted

Fig. 9. Perspective transformed image using four manually selected corner points

In this case, we actually want to see the performance solely for the perspective transformation. The corner points are selected from the image manually then perspective transformations are made and finally the object positions in the image are also selected manually so that there will be a minimum possible error due to detections, **Fig.s 9**, **10** show the transformed outputs.

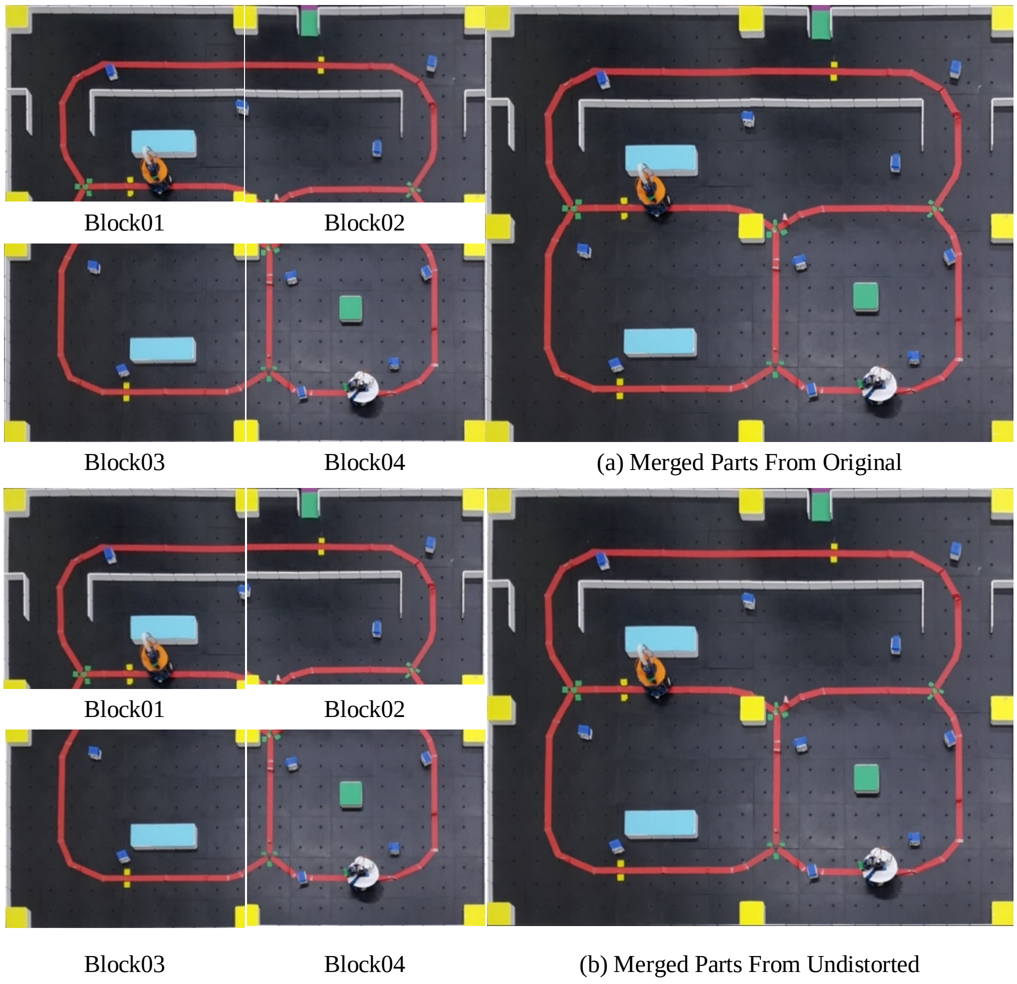

Block01
Block02
Block03
Block04
(a) Merged Parts From Original
Block01
Block02
Block03
Block04
(b) Merged Parts From Undistorted

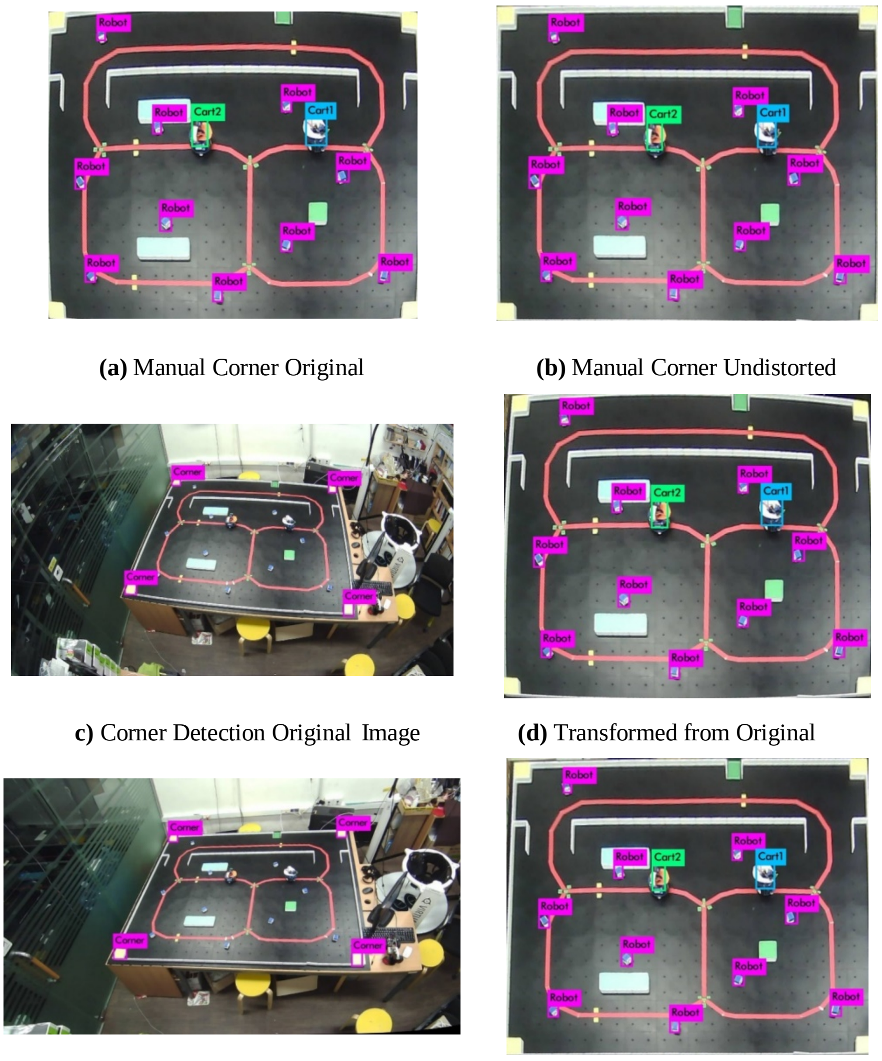
Fig. 10. Perspective transformed image using four manually selected corner points

In **Fig. 9**, we present the transformed output using 4 corner points for both arrangements. When we use the original images as input, the transformed versions also contain the distortions, **Fig.s 9a**, **9b** also show the manual labels for the robots and carts. **Fig. 4a** is taken by camera1 and has barrel distortion, as a result, the transformed version of Fig. 9a clips out the surrounding distorted parts. So, the relative positions of objects inside the image vary accordingly than the real space. Similarly, **Fig. 4b** contains small pincushion distortion which causes little bend at the top edge of the transformed image as in **Fig. 9c**.

However, using the images after removing the distortion results transformed version nicely aligned (**9b, 9d**). All full-size images are 1920×1080 pixels. The real board coordinates are calculated in millimeters whose size is $2070 \times 1710(mm)$ and the transformed image size is 1024×846 pixels. The correspondence of corner points is shown in **Table 3**.

Using all of the 9 boundary points in arrangement2 as described in **Table 3**, the ROI is actually divided into four blocks. As a result, *warpPerspective* is applied to each block individually and then merged into one as shown in **Fig. 10**. For the original image, the distortion is minimized when we use 9 corners, but the improvement for undistorted version is not clearly noticeable.

4.2 Case2: Object detection by the system



(a) Manual Corner Original

(b) Manual Corner Undistorted

c) Corner Detection Original Image

(d) Transformed from Original

(e) Corner Detection Undistorted Image

(f) Transformed from Undistorted

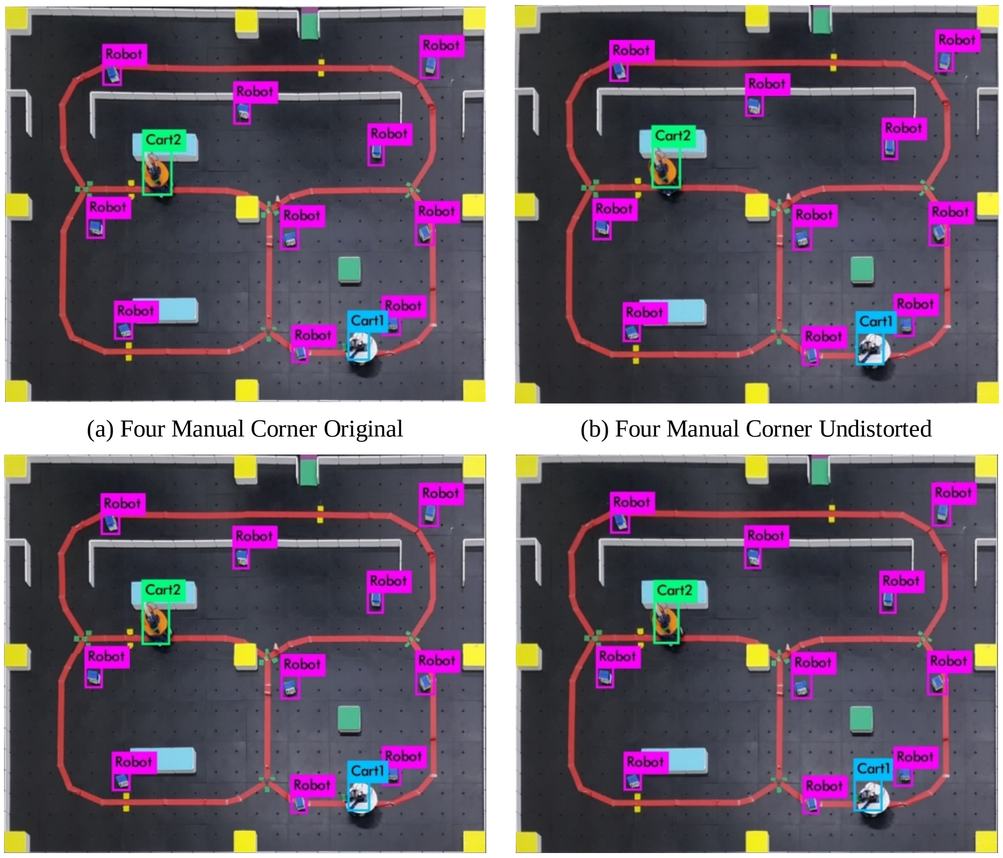
Fig. 11. Object detections on Arrangement1 (Camera1)

Here, object points are taken only from the detection; not by manual selection. At first, for arrangement1 we perform detection in four scenarios as shown in **Fig. 11**. For original and undistorted input images, objects are detected using manual corner points as well as detected by the system. It can be clearly observed again that using original camera image with distortion, loses some ROI part around the boundary walls, actually clips off because camera1 has barrel distortion (**Fig. 11**). Also, if we use detected corner points then the detection bounding boxes are not always perfectly aligned with the corner edges leaving some detection error in the transformed images.

As discussed earlier, arrangement2 have nine corner points and we can use either four or all nine corners (four blocks). The following two sub-cases consider both 9 and 4 corner points for perspective transformation.

4.2.1 Case2.1

Fig. 12 shows the scenarios using manual corner points. We see that object detection bounding boxes are similar in fashion but as pointed out earlier, four corners with original image retain the distortion and we expect more performance gain from undistorted images for four corners (**Fig.s 12a**, **12b** than with nine corners (**Fig.s 12c**, **12d**).



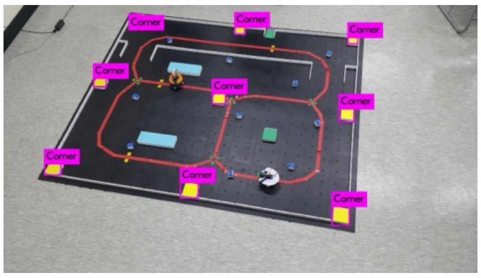
(a) Four Manual Corner Original (b) Four Manual Corner Undistorted

(c) Nine Manual Corner Original (d) Nine Manual Corner Undistorted

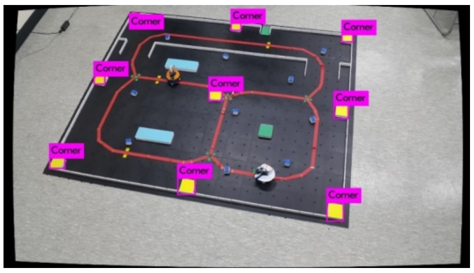
Fig. 12. Object Detections With Manually selected Corner Points on Arrangement2 (Camera2)

4.2.2 Case2.2- Using detected corner points

Here, both corners and objects are detected, not selecting anything manually. **Fig. 13** shows corner and object detections using input images (original and undistorted) considering both four and nine corner points. In original images, four corner keeps the small pincushion distortion of camera2 but nine corner removes that in original images (**Fig.s 13c**, **13e**).

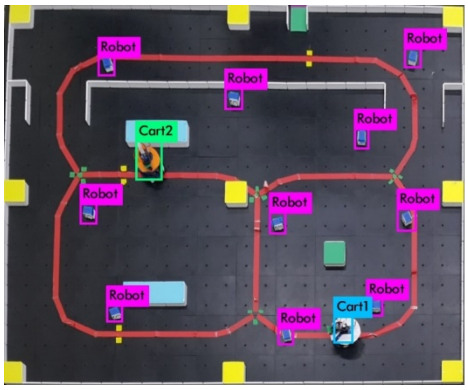


(a) Corner Detection Original Image

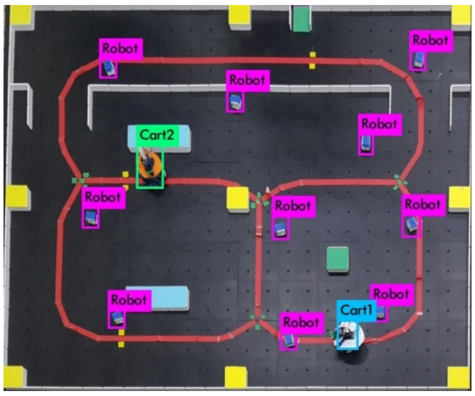


(b) Corner Detection Undistorted Image

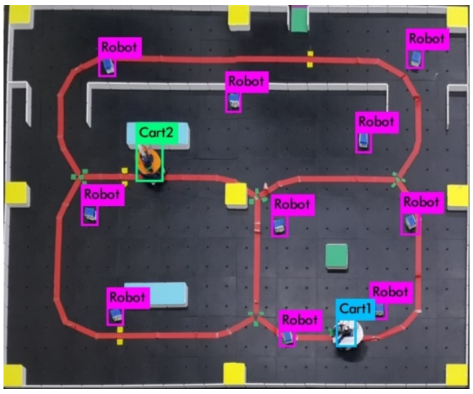
(c) Four Detected Corner Original



(d) Four Detected Corner Undistorted



(e) Nine Detected Corner Original



(f) Nine Detected Corner Undistorted

Fig. 13. Object Detections With Detected Corner Points on Arrangement2 (Camera2)

5. Results and Analysis

As already mentioned, object detection results in the bounding boxes for every object inside the ROI. The middle points are determined and the image points are converted into real space points using equation (14) where image and real dimensions are $1024(px) \times 846(px)$ and $2070(mm) \times 1710(mm)$ respectively, as specified in **Table 3**. There are 12 objects; 10 robots (R1,R2, ... ,R10) and 2 Carts (Cart1, Cart2). We calculate the Euclidean distances between all 66 pairs ($12C2$) of objects {(R1, R2), (R1, R3) ... (R2, R3)........(R10, Cart1), (R10, Cart2),(Cart1, Cart2)}. The distances in real space were recorded earlier using measurement tapes. Finally, we find out the errors in distances between real and calculated for every pair.

Tables 4 and **5** show the errors of few distance IDs for all 18 scenarios and the **Fig.s through 14-18** present the plots for all distance IDs with their corresponding errors.

$$X_{real(mm)} = \frac{X_{image(px)} \times RealWidth(mm)}{ImageWidth(px)}$$
$$Y_{real(mm)} = \frac{Y_{image(px)} \times RealHeight(mm)}{ImageHeight(px)} \quad (14)$$

Table 4. Absolute errors i.e differences between real and calculated from system for arrangement1 (Camera1). Camera1 has barrel distortion and here we only used four corner points. Orig = Original Image, Und = Undistorted Image

Distance IDs	**Real Distance (cm)**	**Manual Detection**		**Detection by System**			
				Manual Corner		**Detected Corner**	
		Orig	**Und**	**Orig**	**Und**	**Orig**	**Und**
1(R1-R2)	81.05	0.06	0.60	2.34	0.81	0.61	2.41
4(R1-R5)	108.32	1.13	0.31	2.78	0.34	1.12	4.16
12(R2-R3)	52.00	0.32	0.95	1.70	0.35	0.54	0.55
14(R2-R5)	52.91	0.16	0.80	0.91	0.01	0.43	0.38
22(R3-R4)	88.78	3.00	1.33	3.56	0.04	3.67	0.67
23(R3R5)	51.60	1.32	1.05	0.82	0.44	2.01	0.80
31(R4-R5)	51.34	1.33	0.60	2.90	0.44	2.04	0.52
39(R5-R6)	48.41	1.69	0.49	1.57	0.26	1.86	0.66
40(R5-R7)	89.46	3.15	1.35	5.68	1.11	3.85	1.52
41(R5-R8)	65.06	3.34	0.37	3.88	0.23	3.34	0.16
42(R5-R9)	97.94	3.33	1.31	6.28	0.46	5.09	0.55
43(R5-R10)	122.80	2.77	0.39	2.99	0.82	1.88	0.09
44(R5-Cart1)	92.93	5.22	0.32	5.49	1.34	3.77	0.94
45(R5-Cart2)	49.72	4.01	0.56	3.09	0.94	1.44	0.72
46(R6-R7)	107.60	3.58	1.56	6.23	0.73	4.85	1.42
52(R7-R8)	74.70	1.44	1.00	3.84	0.80	1.41	1.92
57(R8-R9)	48.31	0.10	1.82	3.52	0.21	1.93	0.20
61(R9-R10)	59.82	0.74	1.35	1.14	0.01	0.56	0.49
64(R10-Cart1)	89.18	0.61	0.04	0.08	1.45	1.90	2.25
66(Cart1-Cart2)	60.54	2.19	0.00	3.93	0.51	1.92	0.41
Average		1.97	0.81	3.14	0.57	2.21	1.04

Table 5. Absolute errors i.e differences between real and calculated from the system for arrangement2 (Camera2). Camera2 has pincushion distortion. Orig = Original Image, Und = Undistorted Image

Distance IDs	**Real Distance (cm)**	**Manual Detection**				**Detection by System**							
						Manual Corner				**Detected Corner**			
		4 Corner		**9 Corner**		**4 Corner**		**9 Corner**		**4 Corner**		**9 Corner**	
		Orig	**Und**	**Orig**	**Und**	**Orig**	**Und**	**Orig**	**Und**	**Orig**	**Und**	**Orig**	**Und**
1(R1-R2)	77.65	0.65	0.56	0.25	0.04	1.04	0.66	0.71	0.46	1.81	0.04	0.92	0.61
4(R1-R5)	82.37	0.64	1.38	0.06	0.47	0.23	1.02	0.76	1.13	0.54	1.47	0.71	1.15
12(R2-R3)	41.91	0.25	0.20	0.20	0.39	0.64	0.31	0.38	0.47	1.25	0.19	0.00	0.46
14(R2-R5)	50.25	0.76	0.42	0.07	0.03	0.74	0.60	0.22	1.03	0.27	0.92	0.00	0.09
22(R3-R4)	136.77	2.15	0.28	0.98	0.66	2.56	1.21	2.08	1.79	3.16	1.38	2.34	1.60
23(R3R5)	58.58	1.02	0.37	0.16	0.32	0.52	1.03	0.71	0.25	1.87	0.69	0.97	0.61
31(R4-R5)	85.15	1.33	0.30	0.75	0.32	1.07	0.54	0.65	0.37	1.35	0.04	0.94	0.62
39(R5-R6)	59.05	0.56	0.58	0.19	0.39	1.22	0.70	1.11	1.13	1.81	1.02	1.23	1.03
40(R5-R7)	105.44	0.28	0.45	0.99	0.17	1.39	0.58	1.40	0.73	2.85	1.08	1.59	1.23
41(R5-R8)	59.46	0.22	0.34	0.03	0.37	2.15	0.89	0.19	0.44	2.45	0.20	0.03	0.65
42(R5-R9)	54.09	0.39	0.18	0.25	0.54	3.13	0.57	2.64	2.78	1.64	0.57	1.35	1.50
43(R5-R10)	96.99	0.20	0.49	0.30	0.12	2.46	0.21	2.14	2.38	1.90	1.03	1.33	1.40
44(R5-Cart1)	57.56	1.66	0.47	0.03	0.64	2.16	2.15	2.45	3.08	3.26	1.36	1.91	1.62
45(R5-Cart2)	63.96	0.05	0.96	0.60	0.61	1.91	3.77	2.23	2.60	2.09	2.54	2.02	2.58
46(R6-R7)	152.95	0.90	1.07	0.80	0.20	1.52	0.42	1.66	0.62	3.92	1.34	1.82	1.33
52(R7-R8)	59.55	0.23	0.34	0.84	0.34	0.85	0.19	1.65	0.86	0.17	0.50	1.53	1.43
57(R8-R9)	60.58	0.87	0.44	0.36	0.14	0.10	0.50	0.11	0.56	2.46	0.32	0.43	0.06
61(R9-R10)	43.41	0.59	0.66	0.04	0.48	0.98	0.91	0.62	0.63	0.21	0.40	0.24	0.27
64(R10-Cart1)	127.21	1.35	1.06	0.90	0.56	3.57	0.57	3.59	3.58	3.36	1.60	2.63	1.97
66(Cart1-Cart2)	116.98	1.01	0.69	0.89	0.03	0.30	2.54	0.41	0.31	0.85	1.69	0.51	1.55
Average		0.76	0.56	0.43	0.34	1.43	0.97	1.29	1.26	1.86	0.92	1.13	1.09

The bar-charts on the right-sides of the **Fig.s 14 - 18** present another interpretation for the corresponding plots on the lefts. We could use the average errors for each scenario to plot the bar charts but it does not always provide the proper interpretation in this regard. If most of the detection bounding boxes are nicely aligned except a single which have big displacement then the average can be affected. As a result, for every figure, we calculate normalized scores and the average of those are plotted in the bar charts.

5.1 Average Normalized Error Calculation

Let, in a figure there are four comparing scenarios(S_1, S_2, S_3, S_4), for a specific distance ID X_1 the corresponding errors are $(E_1X_1, E_2X_1, E_3X_1, E_4X_1)$ then the corresponding normalized errors will be assigned from a sequence of four equally spaced constant values (C_1, C_2, C_3, C_4) where $C_1 > C_2 > C_3 > C_4$. If $(E_1X_1 > E_3X_1 > E_2X_1 > E_4X_1)$ then we get the normalized errors; $(N_1X_1 = C_1, N_2X_1 = C_3, N_3X_1 = C_2, N_4X_1 = C_4)$. Similarly, for another distance ID X_2, if $(E_3X_2 > E_1X_2 > E_4X_2 > E_2X_2)$ then $(N_1X_2 = C_2, N_2X_2 = C_4, N_3X_2 = C_1, N_4X_2 = C_3)$, and so on. Finally, the average normalized errors

$AVG(N_1X_1, N_1X_2 \ldots\ldots N_1X_n)$,

$AVG(N_2X_1, N_2X_2 \ldots\ldots N_2X_n)$,

$AVG(N_3X_1, N_3X_2 \ldots\ldots N_3X_n)$ and

$AVG(N_4X_1, N_4X_2 \ldots\ldots N_4X_n)$ are plot as bar chart. In our experiments, we use $C_1 = 20, C_2 = 15, C_3 = 10, C_4 = 5$ and accordingly the average normalized errors (ANE) lies between 5 and 20.

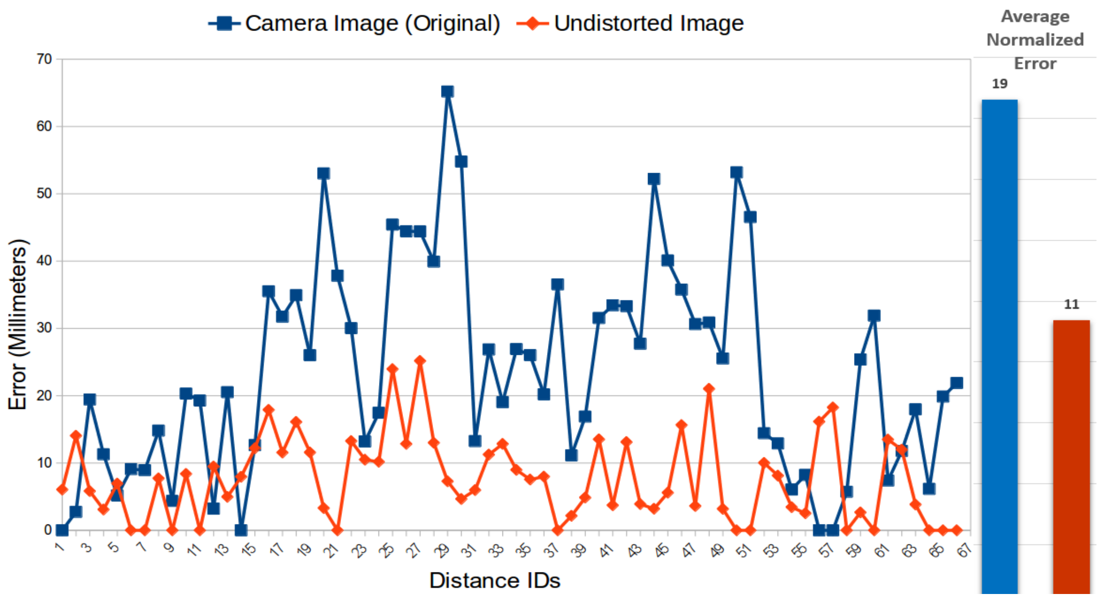


Fig. 14. Manual Object Detections using Manually Selected Corner Points with Original and Undistorted Images of Camera1

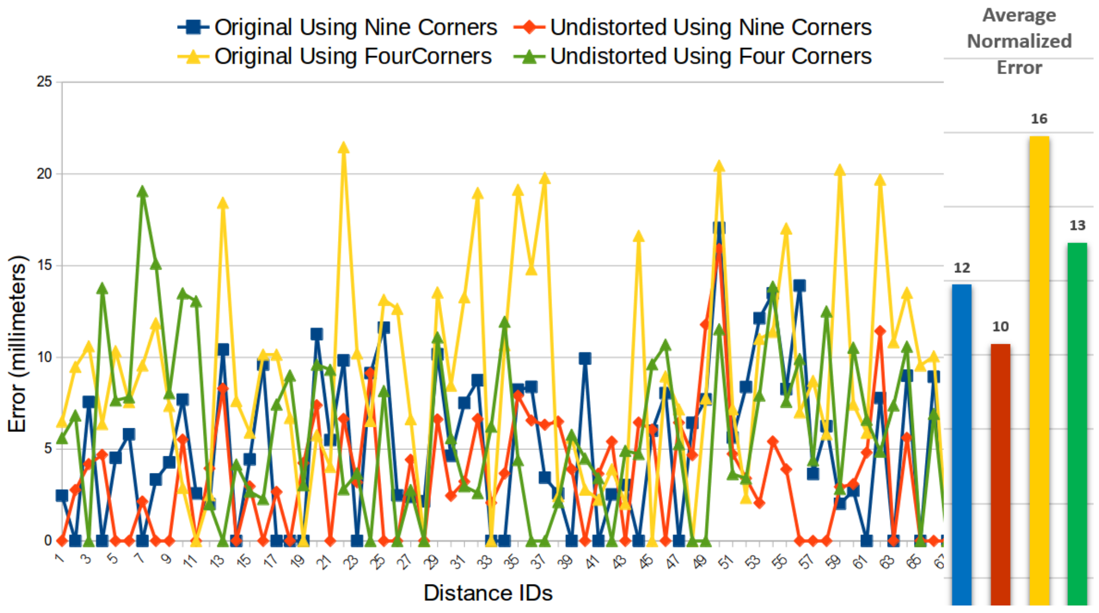


Fig. 15. Manual Object Detections using four and nine Manually Selected Corner Points with Original and Undistorted Images of Camera2

Fig. 14 shows the results for manual detection of arrangement1. Because camera1 has big barrel distortion, the errors for original input image significantly bigger than the undistorted version, the average normalized errors (ANE) are 19 and 11 respectively.

Fig. 15 shows the result of arrangement2 where there is a small pincushion distortion. Here the undistorted version shows smaller errors as well; 10 and 13 average normalized errors for nine and four corners respectively. However, the original image with nine corners reveals similar (bar chart shows better) result as the undistorted image with four corners (ANE = 12 and 13).

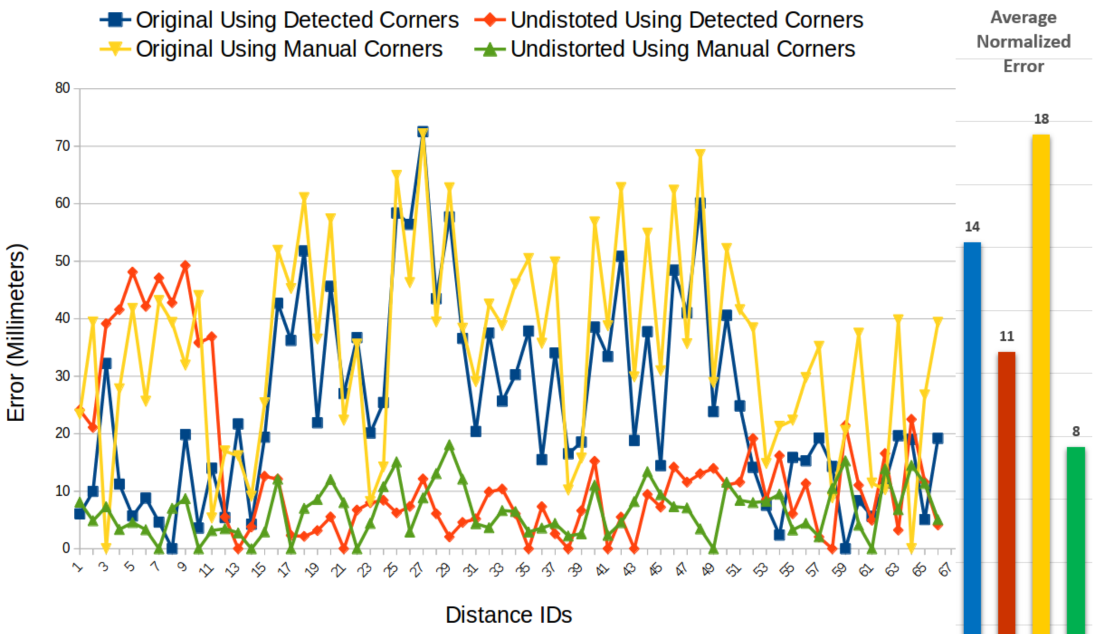


Fig. 16. Detections using Detected and Manually Selected Corner Points with Original and Undistorted Images of Camera1

We can define the total error as in equation 13. All errors are independent of each other but how one kind of error affects the other is uncertain.

$$totalError = \sum(cornerDetectioError,\ transformationError, objectDetectionError) \quad (15)$$

Now let's see the results for detection by the system, **Fig. 16** shows the result for arrangement1; here only four corners are used. We see that using an undistorted image with manual corner provides the best result (ANE=8). Surprisingly, original image with manual corner generates the worst result (ANE=18) than detected corners (ANE=14). This is because camera1 has big barrel distortion and manual corner keeps them all whereas error from detected corners actually decreases the total error in this case. In case of object detection using manual corners on arrangement2 where no corner detection error, we see the improvement using nine corner points for the original image; average normalized errors 13 and 15 with nine and four corners respectively (**Fig. 17**). However, four corners again provide very good accuracy (ANE=10) for undistorted version. When both corners and objects are detected, using nine corners we find similar accuracy for both original and undistorted images (ANE=12)

(**Fig. 18**). It happens due to the uncertain combination of different detection errors. Either manual corner or detected, four corners with original and undistorted images have the highest and lowest errors respectively (ANE= 15,10 and ANE=16,10). Nine corners always improve the accuracy and seem more reliable with an original image.

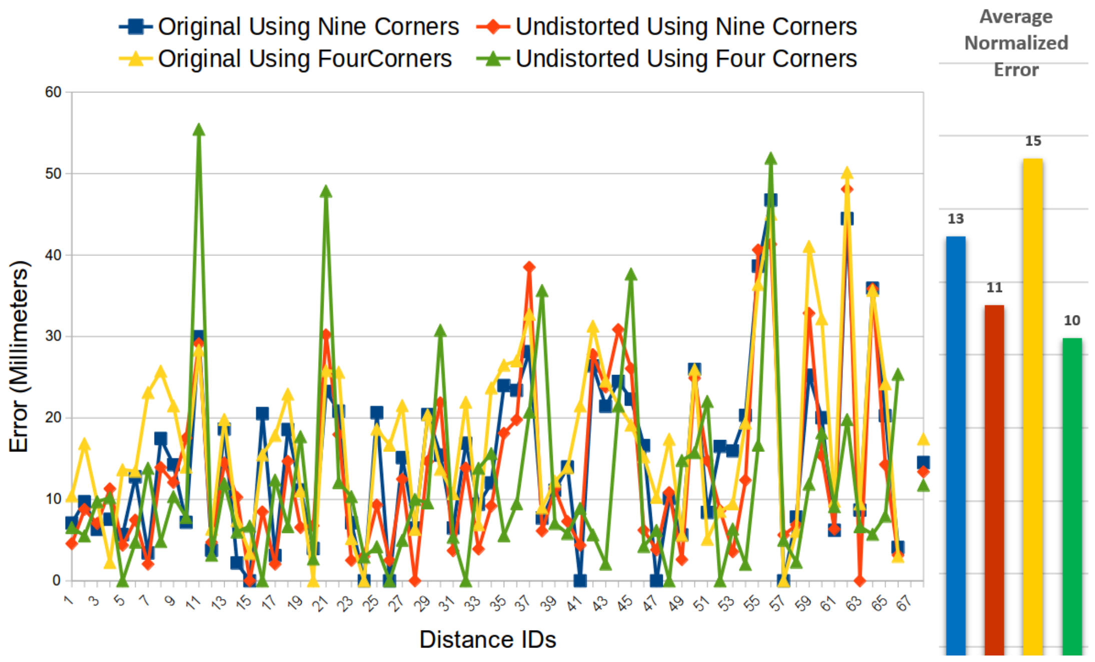


Fig. 17. Detections using four and nine Manually Selected Corner Points with Original and Undistorted Images of Camera2

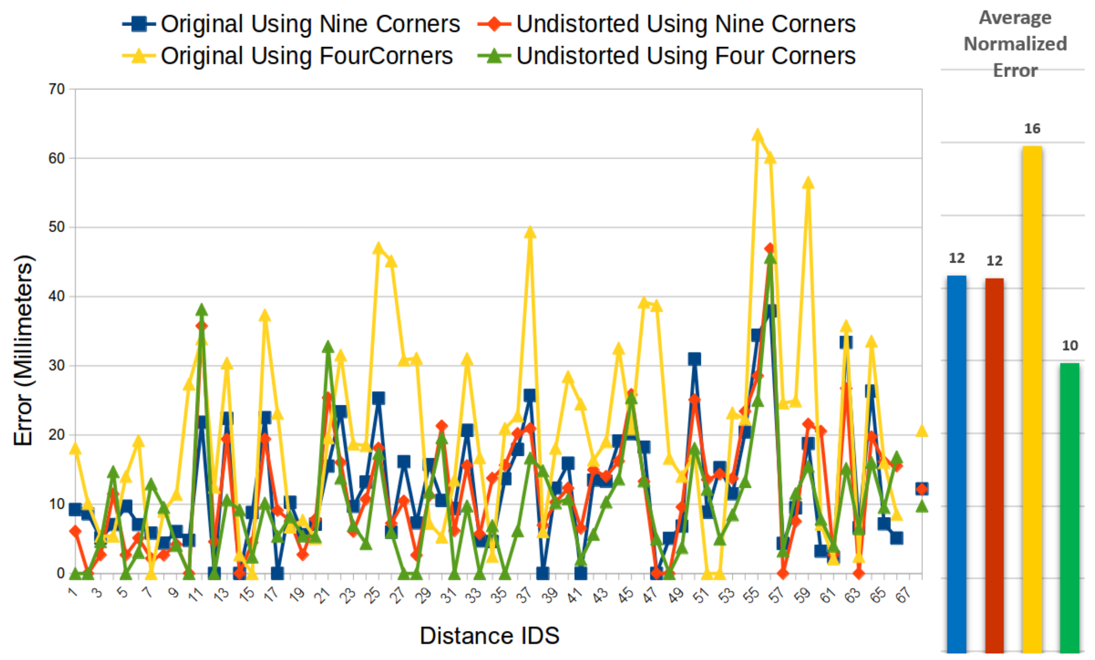


Fig. 18. Detections using four and nine Detected Corner Points with Original and Undistorted Images of Camera2

From all of the results discussed above, we can conclude that perspective transformation gives very good accuracy in locating objects from images. If we can undistort the image then only four corner points improve the accuracy significantly. Dividing the ROI into more blocks achieves some sort of auto-calibration and un-distortion becomes less important. Although manual corner selection on undistorted image performs even better, it is not practical. Both removing the distortion with calibration or using more boundary points in order to achieve auto-calibration gains accuracy improvement with the cost of some complexity increase. However, using more control points seems to have more advantages than camera calibration and undistortion because it does not depend on a specific camera and thus can help us to develop a general system using multiple cameras with motion.

Table 6. Comparison of errors between our method and Mobile Robot Self-Localization System [27]

Method of [27]		**Our Method**		
Camera	**Error (cm) (undistorted image)**	**Camera**	**Error (cm)**	
			undistorted image	**original image**
Camera1	6.86	Camera1	1.08	1.05
Camera2	12.11	Camera2	1.53	2.18
Camera3	9.01			
Average	**9.33**	**Average**	**1.30**	**1.62**

The closest method which we can compare with our work is the Mobile Robot Self-Localization System [27] where they experiment with three webcams. They measured the displacements of specific positions whereas we measured the distances between objects. To compare with them, we calculated our results in a similar way and the results are shown in **Table 6**. The reference method measures the displacements on the calibrated images only whereas we consider several cases along with both calibrated and original images. Above table shows the average error of that method is 9.33 cm but our method has 1.30cm and 1.62cm for calibrated and original images respectively. Moreover, our proposed method can be trained to detect any kind of objects which gives it a wider range of applicability.

6. Conclusion

In this paper, we proposed a system to detect objects from a single remote image frame and then locate and measure the distance between them in real space. The detection is done by deep learning and performed in two steps, the boundary marks, and object detection. Perspective transformation is used to crop the ROI from image and make it proportional to the real space. Experiments are done using original images, after removing their distortions by camera parameters, with four and nine corners. Results prove that combining perspective transformation and object detection provides a simple solution for locating objects in real space and using more corner points improves the accuracy significantly for original images. However, if we remove the distortion from images prior to the transformation then four corner points are enough to provide very good accuracy.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436-444, May, 2015. Article (CrossRef Link)

[2] C. Szegedy and V. O. Vanhoucke, "Processing images using deep neural networks," *United States patent* US9904875B2, February, 2018. Patent Link

[3] J. Redmon, "Darknet: Open Source Neural Networks in C,". Web Link.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, June 2016. Article (CrossRef Link)

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, April, 2018. Article (CrossRef Link)

[6] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-21, January, 2019. Article (CrossRef Link)

[7] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517-6525, July, 2017. Article (CrossRef Link)

[8] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-YOLO: Real-time 3D Object Detection on Point Clouds," *arXiv preprint arXiv:1803.06199*, March, 2018. Article (CrossRef Link)

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks.," in *Proc. of CVPR*, 2017, vol. 1, pp. 2261-2269, July, 2017. Article (CrossRef Link)

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the 2016 IEEE conference on computer vision and pattern recognition*, pp. 770–778, June, 2016. Article (CrossRef Link)

[11] X. Hao, "A FAST OBJECT DETECTION METHOD BASED ON DEEP RESIDUAL NETWORK," *INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH*, vol. 7, no. 2, February, 2018. Article (CrossRef Link)

[12] C. N. Aishwarya, R. Mukherjee, and D. K. Mahato, "Multilayer vehicle classification integrated with single frame optimized object detection framework using CNN based deep learning architecture," in *Proc. of 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–6, March, 2018. Article (CrossRef Link)

[13] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330-1334, November, 2000. Article (CrossRef Link)

[14] "Measuring Planar Objects with a Calibrated Camera - MATLAB & Simulink," 2018. Web Link

[15] M.-C. Lu, W.-Y. Wang, and C.-Y. Chu, "Image-based distance and area measuring systems," *IEEE Sensors Journal*, vol. 6, no. 2, pp. 495–503, April, 2006. Article (CrossRef Link)

[16] Y. M. KlimKov, "A laser polarimetric sensor for measuring angular displacements of objects," in *Proc. of Lasers and Electro-optics Europe, 1996. CLEO/Europe., Conference on*, pp. 190–190, September, 1996. Article (CrossRef Link)

[17] A. Carullo and M. Parvis, "An ultrasonic sensor for distance measurement in automotive applications," *IEEE Sensors journal*, vol. 1, no. 2, p. 143, August, 2001. Article (CrossRef Link)

[18] Y. M. Mustafah, R. Noor, H. Hasbi, and A. W. Azma, "Stereo vision images processing for real-time object distance and size measurements," in *Proc. of Computer and Communication Engineering (ICCCE), 2012 International Conference on*, pp. 659–663, July, 2012. Article (CrossRef Link)

[19] I.-H. Kim, D.-E. Kim, Y.-S. Cha, K. Lee, and T.-Y. Kuc, "An embodiment of stereo vision system for mobile robot for real-time measuring distance and object tracking," in *Proc. of Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pp. 1029–1033, October, 2007. Article (CrossRef Link)

[20] K. Murawski, "Method of Measuring the Distance to an Object Based on One Shot Obtained from a Motionless Camera with a Fixed-Focus Lens.," *Acta Physica Polonica, A.*, vol. 127, no. 6, pp. 1591-1595, June, 2015. Article (CrossRef Link)

[21] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, October, 2013. Article (CrossRef Link)

[22] M. Jüngel, H. Mellmann, and M. Spranger, "Improving vision-based distance measurements using reference objects," in *Proc. of Robot Soccer World Cup*, 2007, pp. 89–100, 2008. Article (CrossRef Link)

[23] H. Zhang, L. Wang, R. Jia, and J. Li, "A distance measuring method using visual image processing," in *Proc. of the 2009 2nd International Congress on Image and Signal Processing*, 2009, vol. 1, pp. 2275–2279, October, 2009. Article (CrossRef Link)

[24] A. Roberts, W. N. Browne, and C. Hollitt, "Accurate marker based distance measurement with single camera," in *Proc. of Image and Vision Computing New Zealand (IVCNZ), 2015 International Conference on*, pp. 1–6, November, 2015. Article (CrossRef Link)

[25] K. Chan, A. Ordys, and O. Duran, "A system to measure gap distance between two vehicles using license plate character height," in *Proc. of International Conference on Computer Vision and Graphics*, pp. 249–256, 2010. Article (CrossRef Link)

[26] M. Raza, Z. Chen, S. Ur Rehman, P. Wang, and J. Wang, "Framework for estimating distance and dimension attributes of pedestrians in real-time environments using monocular camera," *Neurocomputing*, vol. 275, pp. 533–545, January, 2018. Article (CrossRef Link)

[27] I. Li, M.-C. Chen, W.-Y. Wang, S.-F. Su, and T.-W. Lai, "Mobile robot self-localization system using single webcam distance measurement technology in indoor environments," *Sensors*, vol. 14, no. 2, pp. 2089–2109, January, 2014. Article (CrossRef Link)

[28] J. H. Shim and Y. I. Cho, "A mobile robot localization via indoor fixed remote surveillance cameras," *Sensors*, vol. 16, no. 2, p. 195, February, 2016. Article (CrossRef Link)

[29] Tzutalin, "LabelImg," *Git code (2015)*. Code Link.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June, 2017. Article (CrossRef Link)

Md. Abu Layek received his B.Sc. and M.Sc. degrees from Information and Communication Engineering department, Islamic University, Bangladesh in 2004 and 2006 respectively. He is an Assistant Professor in the department of Computer Science and Engineering, Jagannath University, Dhaka, Bangladesh. At present, he is pursuing his PhD in Computer Science and Engineering, Kyung Hee University, Republic of Korea. His current research interest includes Cloud Computing, Machine Learning, Screen Contents Coding and Image Quality Assessment.

TaeChoong Chung received the B.S. degree in Electronic Engineering from Seoul National University, Republic of Korea, in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST, Republic of Korea, in 1982 and 1987, respectively. Since 1988, he has been with Department of Computer Engineering, Kyung Hee University, Republic of Korea, where he is now a Professor. His research interests include Machine Learning, Meta Search, and Robotics.

Eui-Nam Huh earned a B.S. degree from Busan National University in Korea, a Master's degree in Computer Science from the University of Texas, USA in 1995, and a Ph.D. degree from the Ohio University, USA in 2002. He is the director of Real-time Mobile Cloud Research Center. He is a chair of Cloud/BigData Special Technical Committee for Telecommunications Technology Association(TTA), and a Korean national standards body of ITU-T SG13 and ISO/IEC SC38. He was also an Assistant Professor at Sahmyook University and Seoul Women's University, South Korea. He is now a Professor in the Department of Computer Science and Engineering, Kyung Hee University, South Korea. His research interests include Cloud Computing, Screen Contents Coding(Cloud Streaming), Internet of Things, Distributed Real-Time Systems, Security, and Big Data.