Low-Cost Computer Vision Based Real-Time 3D Localization of Object for Robotic Applications

Vikram Rao
Indian Institute of Information Technology Design & Manufacturing, Chennai
rao.vikram@iiitdm.ac.in

Munesh Singh
Indian Institute of Information Technology Design & Manufacturing, Chennai
munesh.singh@iiitdm.ac.in

***Abstract*—In this paper, we try to solve this problem by proposing low-cost robust 3D vision for robotics. This system helps robot automatically detect and locate 3D coordinates of an object in space at real-time. In manufacturing this is very useful to automate repeated object pick and place operations. Initially we improvised 2D object detection. Later, we estimated object depth using Stereo Vision techniques. Overall we combine stereo vision techniques, object detection techniques, geometric calibration techniques, inverse kinematics into one model avoiding redundant computations and greatly simplifying underlying equations to achieve real-time speed and good-accuracy. We reduce redundant computations by replacing standard stereo matching with our own object detection model. We accurately localize detected objects by performing calibrations, improvising real world 2D coordinate equations and improvising depth estimation equation. This single framework is practically implemented and developed using open source platforms. From experimental results, we have observed the proposed model shows superior performance (Above 99% object detection rates, Accuracy upto 8mm, Only ∼23% CPU utilization, Only ∼155MB of RAM consumption) on an average and is cost-effective (Under $20). This low-cost model can be useful for industrial purpose, small businesses, STEM education or for training and skill development.**

***Index Terms*—Smart Manufacturing, Machine Vision, Robotics**

I. Introduction

A. Motivation

Developing countries are getting prepared for Smart Manufacturing. India being the sixth-largest manufacturing country is taking steps to improve its productivity through robotics. As per a report [1], India is estimated to have 24% CAGR growth by 2020 and expected to contribute 25% of GDP by 2022 in industrial manufacturing sector. Upskilling of workforce and training was found to be a major challenge. Robots were also found to be used rarely in factories [1]. This inspires us to identify and look for feasible solution for such problem.

We focus on three aspects Low-Cost, improvisation of existing technology and develop practical models useful for industrial purpose, small businesses, STEM education or for training and skill development.

B. Background

In this paper, we introduce a robotic model enabled with 3D vision to automatically detect object in 3D space. We build a system as shown in Fig. 1 which has two camera's Fig. 1(1,2) mounted onto a fixed frame for 3D vision purpose. Our model as shown in Fig. 1 is further explained in detail in upcoming sections.

Notable amount of research has been already done in the area of 3D vision for robotics. Roland et al. [2] proposed a visual system using stereo vision to control a robotic arm in 3D space. Their model was able to determine amount of movement needed by each of the motors to reach the desired object. Roland et al. [2] did develop Low-Cost stereo apparatus but on the other hand, they had to use RoboRealm which is currently commercial software and license exceeds $\$500$ per machine. Author has mentioned that system needs some improvements and how much precise their model is yet to be determined. Similarly, real-time processing and depth error analysis of their model is yet to be done. We develop similar Low-Cost stereo model as proposed by Roland et al. [2] with further improvements and analysis.

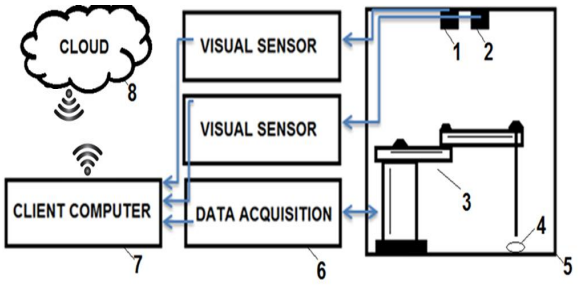

Fig. 1. Structure of our model: Stereo vision setup at (1,2), robotic manipulator (3), robotic cell (5), object of interest (4), data acquisition device (6), host PC (7) and internet(8).

Mohamed et al. [3] developed a visual tracking control system for a Movable Robot Head. Using fuzzy logic, they were successfully able to enable a robot to move its head like humans and track a moving object at the same time. They have practically experimented and results indicate the head movement is quite smooth as expected. Mohamed et al. [3] attempted to perform coordinate transformation from camera to real-world coordinates on which we go further by replacing their complex matrix multiplications with simple slope-intercept equations. Mohamed et al. [3] used Bumblebee2[19] stereo camera apparatus which is good choice as they are accurate with average depth error as low as $\sim 6mm$ but it

costs as high as $1500. Author mentions further work needs to be done to automate, as per the model user has to manually select the object by drawing a polygon around the desired object.

Bindu et al. [4] has proposed a successful object detection model to detect objects in complex backgrounds. Using Circular Hough Transform (CHT) and Image preprocessing techniques such as color processing, they were successfully able to achieve high success rates. Author pointed out that it can be used in robotics to pick/pluck objects. In our model, we use similar objection detection model and take a step further to improvise model and practically implement it in robotics.

Several other similar model exists and other interesting low-cost stereo models such as [5,6] were also developed using lens but they turned out to be relatively robust for low-ranges upto $1 feet (\sim 30cm)$ only but not for ranges upto 2 meters.

We have combined existing state of the art models to develop a new efficient and cost-effective model. In the entire experiment, Low-Cost stereo setup was used. Our main contribution includes 1) Improvising object detection method, 2) Improvised formulation for Real-World 2D coordinate and depth estimation equations and 3) Real-world application. The rest of the paper is organized as follows: Section 2 presents the Model, Section 3 presents the Experimental Results and Section 4 Conclusion.

II. The Proposed Model

In this section, we discuss the development phases of our proposed model:

A. Geometric Camera Calibrations
B. Improvised Object Detection
C. Estimation of Real-World coordinates
D. Improvised Depth estimation
E. Inverse Kinematics
F. Application in Real-world Robot

A. Geometric Camera Calibrations

Radial and Tangential distortion [7] are two major types of lens distortion. Radial occurs when light rays bend more near the edges of a lens than they do at its center. Tangential distortion occurs when the lens and the image plane are not parallel. To rectify and undistort raw images, we use camera calibration techniques as proposed by [9]. Zhang [9] uses geometric camera calibration parameters such as intrinsic, extrinsic and distortion coefficients to rectify distortion errors. The calibration is carried out using pre-defined 2D image pattern such as chessboard pattern. To calculate these parameters, we need to have multiple position and orientation of this pattern in real world and correspond it with 2D image points. After several iterations, using this chessboard calibration pattern, we obtain set of 2D Image pixel correspondences of this pattern and store it in array. Using these correspondences, we solve for the described camera parameters as per Zhang [9] approach. Mohamed et al. [3] and Heikkila et al. [10] have used similar process in their model to correct distortion errors. The extrinsic parameters consist of a rotation (R) and translation(T) that define the location and orientation of the camera with respect to the world frame. The intrinsic parameters consist of focal length f, optical center O, and the skew distortion coefficient S for each camera. This intrinsic parameters bring both camera image plane into the same plane. See [3,9] for more detailed explanation. We use the same technique and camera calibration is one-time process. So, we calibrate once and save these parameters to a special file and load this file whenever we want to rectify images. Raw stereo image is shown in Fig. 2(1a,1b), and after performing calibration rectified stereo image is shown in Fig. 2(2a,2b). Deviation in the stereo images is measured with the help of epipolar lines [20]. Epipolar lines are indicated by horizontal parallel line in Fig. 2. Epipolar lines are guiding lines they show how well stereo left and right images are horizontally aligned before and after calibration process. Observe how closely circular object are aligned in Fig. 2(2a,2b) vs Fig. 2(1a,1b). The calibrated images are vital and serve as input for estimating Real-World 2D coordinates and depth. This step completes geometric camera calibration process.

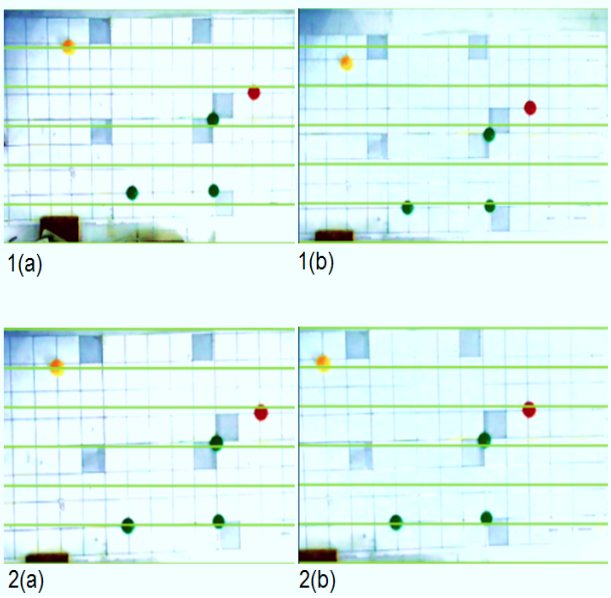

Fig. 2. Geometric Camera Calibration: 1(a,b) represents raw images from stereo left and right camera respectively and 2(a,b) represents calibrated images from stereo left and right camera. Here Epipolar lines represented by parallel lines, show how well stereo left and right images are aligned before and after calibration.

B. Improvised Object Detection

Object detection being one of the major challenging problems in the areas of Machine Vision[7]. Automatic object detection in robotics is vital and makes robots more autonomous. Our scope here is to detect and locate coordinates of a desired object using object detection techniques. This detected coordinates of object will be used in the upcoming section to determine the real-world coordinates of the object. In our model, our object of interest is a red circular object. There are various techniques for circle detection but we found

Circular Hough Transform (CHT) to be more efficient, reliable and has been widely implemented in numerous research areas [4,11,16]. To demonstrate its effectiveness, the experiments have been done to detect object even in complex backgrounds [4]. CHT is well-known to perform best for detecting any parametric curves such as circles, conics, parabola, etc. However, limitations of CHT [4,11] is when there is noise, illumination changes and complex background in the image. We use CHT but focus only on CHT limitation and improve it. We improve it by introducing white background, maintain constant illumination. Illumination is a major issue, hence occasionally we use 50W LED Flood light to maintain constant lighting conditions. We chose camera in such a way that noise such as color noise, salt-pepper noise is quite less. By doing so we succeed in setting ideal conditions for CHT to work. So, unlike [4] we relatively consume lesser Image preprocessing steps to achieve our goal. Our proposed Image preprocessing flow is shown in Fig. 3.

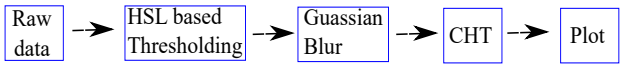

Fig. 3. Proposed image preprocessing flow

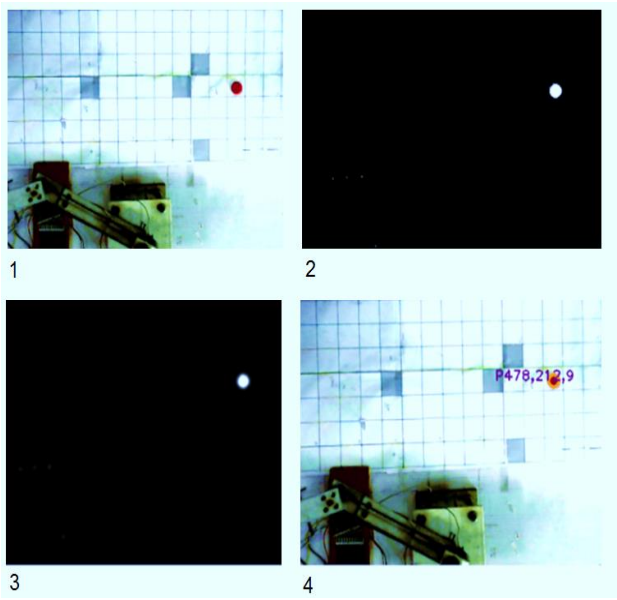

Fig. 4. Image preprocessing steps for object detection. (1) Raw input image, (2) Color Thresholding, (3) Guassian Blur and (4) Apply CHT and plot over input image

Initially we determine upper and lower bound of Hue, Saturation, Lightness (HSL) of our desired red colored object. This is applicable to any other primary or secondary colored object such as green, blue or yellow object. Based on the determined HSL color range from source image in Fig.4(1), we extract color which are in range and fade out rest of the pixels as shown in Fig.4(2). In Fig.4(2), we observe that the white region is in range and black region is out of determined HSL range Interestingly, we observed lesser noise in Fig.4(2) that

TABLE I
DETECTION RESULTS AND COMPARISON

No. of Snaps Collected	Existing vs Our Model Highest success rates		
	[16]	[4]	Our Model
10	95.2%	96.2%	100% (10/10)
100	95.2%	96.2%	99% (99/100)
250	95.2%	96.2%	99.2% (248/250)

shows our environment setup is near ideal condition. Later, we apply Guassian Blur onto Fig.4(2) to reduce noise (if exists) and enhance features as shown in Fig.4(3). Finally, we apply CHT over image Fig.4(3) to retrieve detected circle's center coordinate(2D) and radius(1D). We plot this 2D point over Fig.4(1) and obtain Fig.4(4), where $(P478, 212, 9)$ indicates detected circle's center point(478,212) and radius(9) in pixels. In brief, CHT uses 3D array containing center point of circle(2D) and its radius(1D). The values in the 3D array are increased every time a circle is drawn with a range of radii over every edge point. The 3D array keeps counts of how many circles pass through coordinates of each edge point. This process uses a voting strategy to find the highest count. The coordinates with highest count represent the center of the circle[11]. The high-level steps are presented in Algorithm. 1

Algorithm 1 Simplified Object Detection

Input: Calibrated image
Output: 3D array
Requires: White background with constant illumination.

1) Set upper HSL threshold of desired object
2) Set lower HSL threshold of desired object
3) Clear all pixels which are not in this range
4) Apply Guassian blur filter
5) Apply CHT
6) Return detected circle's center point(2D) and radius(1D)

Each model has its own criteria and problem statement. Table I shows comparison chart of number of snaps taken vs. success rate of existing vs our model. Image enhancement and setting right environment played a vital role in achieving high success rates, and we were able to detect circles 99% of the time. We implemented this phase with the help of open-source OpenCV[8] image processing library.

C. Estimation of Real-World Coordinates

Coordinates retrieved by object detection are in pixel coordinates, our scope here is to infer real-world coordinates in centimeters from retrieved pixel coordinates. By this we can actually sense the location of the object, and calibrate it further to get accurate results. Note that we use results of prior steps such as camera calibration, object detection steps in this phase. After object detection on calibrated images Fig. 5(1,2), we get respective pixel coordinates as shown in Fig. 5(3,4). We use left-stereo image Fig. 5(3) only to calculate real-world

2D (x, y) coordinates. We apply slope-intercept form Eq. (1) to calculate x and Eq. (2) to calculate y, where in x_p, y_p are pixel coordinates of left-stereo image as shown in Fig. 5(3). While, m, c, m', c' are scalar constants and x, y is the desired real-world coordinates. We deduce constants m, c, m', c' from 4 samples and crosscheck with another 4 samples. Computed x, y using Eq. (1,2) is shown in top left corner of Fig. 5(3) (Values: $x = 15cm$ and $y = 20cm$). By correcting most of the distortion errors in previous steps, we were able to estimate real-world coordinates using a linear equation itself accurately. Accuracy analysis presented in results sections.

$$y = m * y_p + c \tag{1}$$

$$x = m' * x_p + c' \tag{2}$$

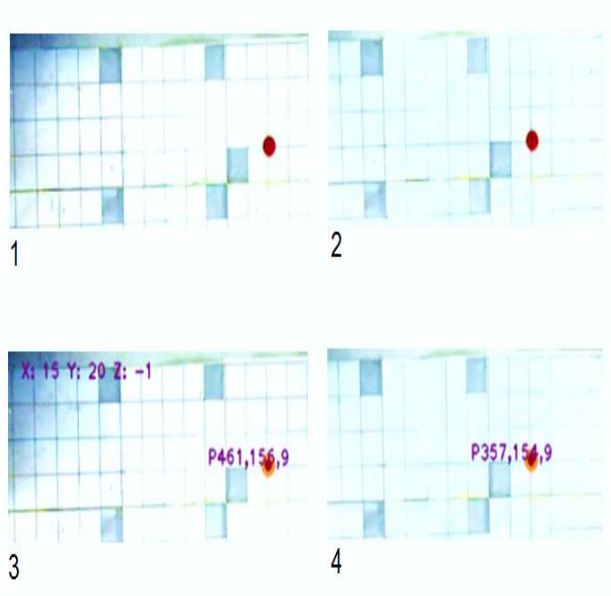

Fig. 5. Estimation of real-world coordinates from pixel coordinates. (1,2) Calibrated Stereo Left and Right Images. (3,4) Plotted Detected Pixel points on calibrated Stereo Left and Right Images respectively

D. Improvised Depth Estimation

Depth is key spatial dimension that enables robotics in many $3D$ applications. Depth can be obtained by using Kinect, Karmin2[17], ZED[18], Bumblebee2[19] and other stereo vision systems. In the proposed model, we use two simple web camera's to sense depth. By depth estimation from stereo vision models [12,13], we can get $3D(x, y, Z)$ coordinates of desired object in space. We continue from previous steps retrieved pixel coordinates from left and right calibrated image as shown in Fig.5(3,4). Using x-coordinates (in pixel) of left and right calibrated image, we determine depth using our proposed depth estimation equation Eq. (4). Computed (x, y, Z) is shown on top-left of Fig. 5(3). From Fig. 6 we observe a triangle indicating position our left camera at O, right camera at O' and the point of interest P. The left camera O perceives image of point P at focal length f distance i.e at x, similarly right camera O' perceives P at x'. Triangles POO' and Pxx' are similar and by applying trigonometry on Fig. 6 we obtain Eq. (3). Equation (3) is equivalent to Fig 6. Equation (4) is derived from Eq. (3), where B is base distance between cameras and focal length f of both camera are identical. Equation (3) is used to find depth under ideal condition. In practice, we had to add extra offset parameters to make it viable in real-life condition. Hence, we use simple heuristics logic and derive Eq. (4) from Eq. (3). The coefficient's in Eq. (4) $m''', n''', c''' > 0$ are deduced from 3 samples and we cross verify with another 3 real-world samples. Introduction of heuristics coupled with previous stereo calibration steps has led to improve accuracy in depth estimation. Results and accuracy are plotted in further sections.

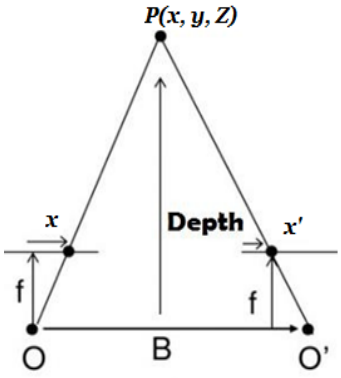

Fig. 6. Depth estimation from stereo images: Our setup with left camera at O, right camera at O', the point of interest at P and Depth(Z) is our desired parameter

$$x - x' = \frac{Bf}{Depth}, Depth = \frac{Bf}{x - x'} \tag{3}$$

$$Depth = m'''(x - x')^{-1} - n'''(x - x') + c''' \tag{4}$$

E. Inverse Kinematics Equations

Given any desired point in space, robot can manipulate its arm to the desired point with the help of Inverse Kinematics(IK) equations. IK reverses the calculation to determine parameters that achieve a desired configuration [14]. It is applicable to all robots of various arm lengths and sizes. The standard inverse kinematics Eq. (5,6) helps you evaluate θ_1 and θ_2. From Fig. 7, if we determine the spatial coordinates of desired point (x, y) using vision techniques then we can calculate the θ_1 using Eq. (6) w.r.t $x = 0$ and θ_2 using Eq. (5) w.r.t first link.

$$\theta_2 = cos^{-1} \frac{(x^2 + y^2) - (l_1^2 + l_2^2)}{2 l_1 l_2} \tag{5}$$

$$\theta_1 = tan^{-1} \frac{y}{x} - tan^{-1} \frac{l_2 sin\theta_2}{l_1 + l_2 cos\theta_2} \tag{6}$$

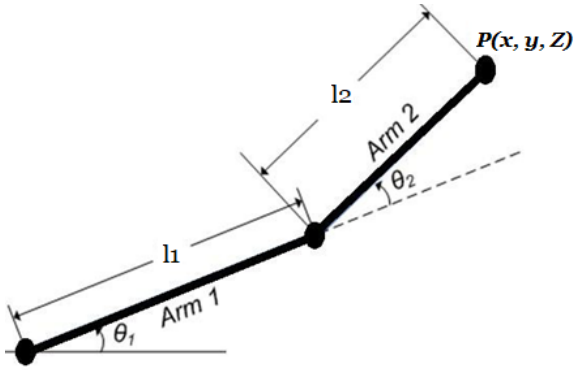

Fig. 7. Orthographic view of Robotic arm

F. Application in Real-world Robot

This section has been divided into three phases:

1) Hardware Setup
2) Software Setup
3) Motors Calibration

1) Hardware Setup: Our hardware setup of robotic cell as shown in Fig.1, where we use HP all-in-one workstation as a host PC shown in Fig.1(7). The workstation has 2GB RAM, Intel i7 3GHz processor. For computer vision purpose, we used 2 Logitech web-cams Fig.1(1,2) connected to host PC via USB cable. For better coverage area, these web-cams are mounted onto fixed frame Fig.1(5). To drive robotic arm motors, we have used industry-standard PITTMANN [15] DC motors Fig.1(3) with encoders. These motors were also mounted onto fixed frame as shown in Fig.1(5). To acquire motor encoder readings, we have used USB4 DIGITAL data acquisition system(DAQ) as shown in Fig.1(6). DAQ is connected to host PC via USB. The DAQ used here can capture samples up to 2 μs. Optionally we connect the system to Internet/Cloud Fig.1(8) to log all the hardware and software feedback useful for future research.

2) Software Setup: We extensively used Dot NET framework, OpenCV[8] framework, Visual Studio Community Edition IDE and C# language for developing our model. USB4 DIGITAL library was used to collect data from DAQ. Visual Studio IDE was used for development, testing, calibration, rectification, and to compute performance metrics.

3) Motor Calibration: In our experiment with DC motors, we have encountered hardware issues such as no two motors exhibited similar characteristics (we observed variable voltage ranges for clockwise, counterclockwise rotation with different motors). By doing calibration numerous times, we finally determine correct voltage readings for each of the motors. Another issue is that the DAQ sends out too many samples (pulses every $2\mu s$). Because of mismatch in clock-speed between DAQ and Host PC, which results in the loss of samples. To avoid such issues, we had to slow down the motor speed gradually much before it reaches destination angle (here we slow-down motors at 5 degrees before the destination angle). By this way, we were gradually able to gently maneuver robotic arm to the desired location.

III. Experimental Results

Our proposed model is practically implemented and experimental results are analyzed in this section. We will be presenting actual values vs. our computed results. We analyze deviation in 2D coordinates, analyze deviation in depth, analyze real-time CPU/RAM performance, and finally present cost-effectiveness.

A. Deviation in 2D coordinates

We collected 40 samples, where each sample contains a ground truth value (3D coordinate) and computed 3D coordinates, then we scatter plot all the values as shown in Fig. 8. From Fig. 8, the green dot represents ground truth value and red box represents computed 3D coordinates of the object in centimeters. We observe very rare $+1$ cm deviation in $x-axis$ and frequent ± 1 cm deviation in the $y-axis$.

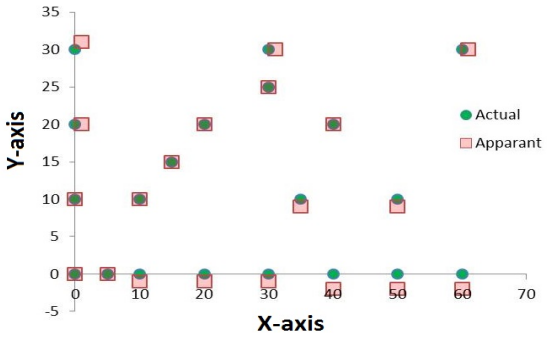

Fig. 8. Scatter plot for deviation in (x, y) position

B. Deviation in depth

In Fig. 9, the histogram bar indicates depth error in centimeter, where we observe average error ± 0.8 cm for over 40 samples. From the results, we have observed very rarely $+2$ cm error and -1 cm error. There are no error observed which is indicated by voids in the histogram. In contrast with high-end ZED[18] SVS whose average error is 0.6 cm, our low-cost model averages at 0.8 cm and this can be improved further by enhancing camera quality.

C. CPU/RAM Performance

Our model was set to run at constant frame-rate of 10 frame per second, over which we collected 653 samples containing system diagnostics parameters such as RAM and CPU usage. We observe peak RAM usage of 1.1 GB for 1.14 second, where the average RAM usage was about 155.71 MB. Similarly, CPU usage peak was at 57%, where the average CPU usage was about 23.37%, which means CPU was 76.63% idle most of the time.

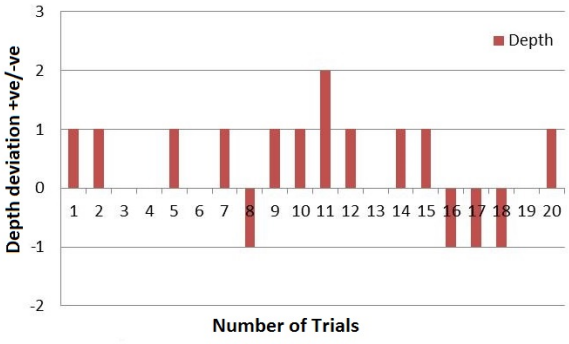

Fig. 9. Graph showing deviation in Depth; Actual vs Ground truth observation

TABLE II
ACCURACY VERSES COST-EFFECTIVENESS

Existing Systems	**Accuracy (0-2m)**	**Cost**
Bumblebee2[19]	*High ∼1cm error*	>$1500
ZED SVS [18]	*High ∼1cm error*	>$400
Karmin2 SVS[17]	*Good ∼2cm error*	>$400
Other low cost[6]	*Good low error*	>$100
Our Model	*Good ∼2cm error*	∼$50
3D Mini Camera Model[5]	No readings presented	<$20
Other low cost models		<$20

D. Review of cost-effectiveness

Cost-effectiveness w.r.t cost verses quality of our system and other existing systems is described in Tab. II. We observe that our system is low cost as well as effective enough in comparison with similar low-cost/high-cost stereo vision systems(SVS) models. On closer look we observe that many other low-cost model are yet to be tested and analyzed for ranges up-to 2 meters.

CONCLUSION

We developed Low-Cost and robust 3D vision model for robotics by improvising existing methods and practically implementing a prototype. We proposed an improvised object detection approach with high success rates. We avoid redundant computations and greatly simplify underlying equations to achieve real-time speed. We accurately localize detected objects by performing geometric camera calibrations, improvising real world 2D coordinate and depth estimation equations. In conclusion, this model helps robot automatically detect and locate object in 3D space at real-time with good-accuracy. From experimental results, we observe our proposed model is efficient and cost-effective than existing models. In future we plan to use Machine Learning techniques for object detection and further improvise equation for more precision.

REFERENCES

[1] Times News Network,"Fourth industrial revolution: is india prepared?," https://timesofindia.indiatimes.com/india/fourth-industrial-revolution-is-india-prepared/articleshow/63439642.cms, May 2018.
[2] Roland Szab, Aurel Gontean, "Controlling a robotic arm in the 3D space with stereo vision," 21st Telecommunications Forum (TELFOR), 2013.
[3] Abdulla Mohamed, Chenguang Yang, Angelo Cangelosi.l, "Stereo vision based object tracking control for a movable robot head," 4th IFAC Conference on Intelligent Control and Automation Sciences, 2016.
[4] S Bindu, S Prudhvi, G Hemalatha, N Raja Sekhar, MV Nanchariahl."Object detection from complex background image using circular hough transform," Int.Journal of Engineering Research and Applications,4:2328, 2014.
[5] Minh Nguyen, Huy Le, Huy Tran, Wai Yeap, "Adaptive stereo vision system using portable low-cost 3D mini camera lens," International Conference on Mechatronics and Machine Vision in Practice, 2017.
[6] C. Ttofis, C. Kyrkou, T. Theocharides, "A low-cost real-time embedded stereo vision system for accurate disparity estimation based on guided image filtering," IEEE TRANSACTIONS ON COMPUTERS, 2015.
[7] Paul van Walreer. Distortion. "Photographic optics", 2009.
[8] Bradski G. and A. Kaehle. "Learning OpenCV: computer vision with the OpenCV library," Sebastopol, CA: OReilly, 2008.
[9] Zhengyou Zhang, "A flexible new technique for camera calibration," IEEE Transactions on pattern analysis and machine intelligence , 22(11):13301334, 2000.
[10] Janne Heikkila, Olli Silven, "A four-step camera calibration procedure with implicit image correction," Proceedings in IEEE, Computer Society Conference on Computer Vision and Pattern Recognition, pages 11061112. IEEE, 1997.
[11] Simon Just Kjeldgaard Pedersen, "Circular hough transform,"Aalborg University, Vision, Graphics, and Interactive Systems, 123:123, 2007.
[12] Gregory D Hager, Wen-Chung Chang, A Stephen Morse, "Robot feedback control based on stereo vision: towards calibration-free hand-eye coordination,"Proceedings In IEEE International Conference on Robotics and Automation, pages 28502856, 1994.
[13] Alexander Mordvintsev and Abid K., "OpenCV: depth map from stereo images," Online OpenCV, 2013.
[14] Richard P Paul, "Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators,"1981.
[15] Haydon Kerk Pittman, "Technical documents," 2018.
[16] Qiu Chen, Koji Kotani, Feifei Lee and Tadahiro Ohmi. "An accurate eye detection method using elliptical separability filter and combined features," Int. Journal of Computer Science and Network Security Vol.9 No.8, 2009
[17] Nerain Karmin2 Stereo Vision System, https://nerian.com/
[18] ZED Stereo Vision System, https://www.stereolabs.com/
[19] Bumblebee2, https://www.ptgrey.com/stereo-vision-cameras-systems
[20] Epipolar line, https://en.wikipedia.org/wiki/Epipolar_geometry