DOI: 10.2507/32nd.daaam.proceedings.077

USING COLOUR-BASED OBJECT DETECTION FOR PICK AND PLACE APPLICATIONS

Moritz Abdank, Mohamed Aburaia & Wilfried Woeber

This Publication has to be referred as: Abdank, M[oritz]; Aburaia, M[ohamed] & Wöber, W[ilfried] (2021). Using Colour-based Object Detection for Pick and Place Applications, Proceedings of the 32nd DAAAM International Symposium, pp.0536-0541, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-33-4, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/32nd.daaam.proceedings.077

Abstract

Pick and place tasks are one of the most typical applications in robotics. Conventional robots perform a fixed task which restricts the workspace in terms of flexibility. With development in mobile robotics [1] and automation it is necessary to introduce sensor systems that enable a more flexible approach. This paper demonstrates a computer vision system which uses a colour-based approach to object detection and estimation of its 3D world coordinates based on a 2D image. The system is used to enable a robot arm to pick up an object and place it on a CNC-mill. Based on the use case the vision system can tell the robot if an object has been detected and where it is relatively to the camera. Using the robot operating system (ROS) and OpenCv, image thresholding is performed for detecting the object. The Canny edge detection algorithm is deployed to mark the object with a bounding box. The retrieved information from those operations is used for the 3D pose estimation. Using a linear calibration, the error of the system can be reduced, which results in an accuracy of 0.09cm.

Keywords: Computer vision; linear regression; object detection; HSV; ROS

1. Introduction

Widely available and cheap cameras enable the opportunity for different computer vision systems. Open-source libraries such as OpenCV [2] and ROS [3] provide a solid basis for developing robotic systems. Therefore, vision systems can be used to expand typical pre-coded robotic tasks such as pick and place applications to a more flexible setup which is not restricted to a predefined pick-up location [4]. In this paper, a colour-based object detection approach is implemented using image thresholding and linear calibration for error compensation. The vision system estimates the 3D world coordinates of a red metal cylinder based on the 2D image stream from a conventional webcam. To test the system, a robot arm is used to pick up the defined object and place it in a CNC-mill. The proposed system is developed with an autonomous task in mind, where a mobile robot transfers the object to the stationary robot arm. To compensate for the inaccuracy of the mobile robot, a camera observes the delivery zone. When the object is placed in the workspace of the robot, a vision system estimates the 3D position of the object and therefore the accuracy of the mobile delivering robot does not have to be exact. The addressed problem is the calculation of the 3D pose based on the 2D stream of the image. The contribution of this paper is the accuracy analysis of a computer vision framework and the object detection without machine learning.

2. State of the art

One of the more challenging tasks with computer vision systems is object detection. Khan, et. al. [5] especially mention viewpoint changes, scale, and illumination as some of the major variables which result in challenges for object detection algorithms. Since colour can change based on the illumination and the surroundings, colour information is often discarded [5] as an option for object detection [6]. managed to use colour-based information for the estimation of the location of traffic signs. The location was then used in combination with a support vector machine (SVM) [7] to classify the sign.

Image thresholding is one of the fastest methods for object detection [6] and was successfully implemented by [8] for object tracking with autonomous unmanned aerial vehicles (UAV). Using the HSV colour spectrum instead of the RGB spectrum, [8] implemented a threshold-based colour detection for tracking their defined object. To reduce the complexity mentioned in [5], they defined certain parameters and restrictions for the system.

State of the art object detection system can also utilise convolutional neural networks. YoloV3 [9] is one of the fastest real time object detection systems, with an image processing rate of 30 FPS on a Titan X. Such implementations can be used for various applications. An UAV approach to detect forest fires with this network resulted in 6FPS [10] and another YoloV3 based detector mentioned in [11] managed to reach up to 10FPS for a real-time traffic sign recognition. As mentioned by [18] the limitations of a real-time Yolo system is highly dependent on the hardware.

3. Methods

To implement the vision system, a Logitech C270 RGB-Webcam is used. After calibration, the intrinsic matrix can be used to get the first parameters for estimating the 3D pose. With the premise of the pinhole camera model [12], the relation of similar triangles is the only restriction left, requiring that the object has a plane surface and is parallel to the camera. This setup resembles the premise from the UAV object tracking setup to reduce the complexity of the setup and therefore enable a fast and accurate detection method. The overall process flow is presented in Fig. 1.

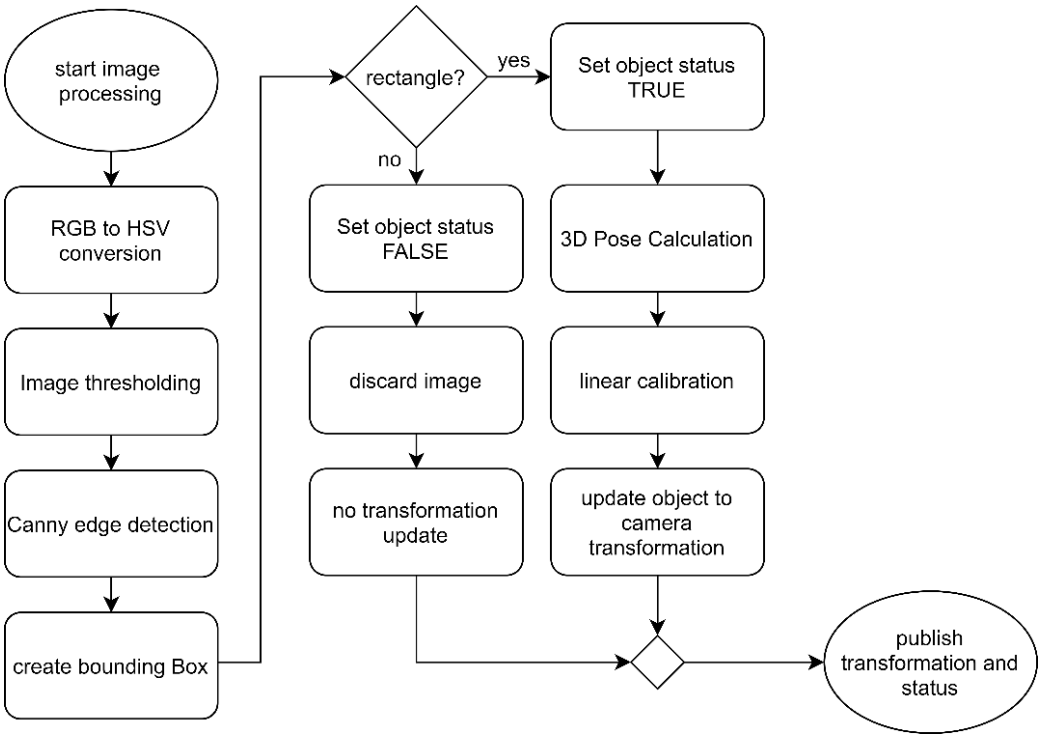

Fig. 1. Program flow diagram showing the different stages of the image processing and decision if the object is truly detected or if a false recognition is the case

For the object detection algorithm, which defines the range for the 3D pose estimation, a colour-based approach has been chosen to see if it can perform comparably to state of the art detection systems like the mentioned YoloV3 implementations [10] and [11]. The complete process of the colour-based system is shown in Fig. 1.In the first step, the rectified image is converted from the RGB-colour spectrum to the HSV-colour range. Since colour-based detection models are highly dependent on the lighting conditions, this enables the option to adapt the desired colour intensity and saturation more easily.

To detect the object, a threshold must be defined from experiments. An erode and dilate filter was implemented to eliminate noise [13] and prevent false detection to a certain degree. With the threshold image in place, the Canny algorithm [14] is deployed to find the edges of all the visible objects and the corresponding bounding boxes. The biggest bounding box is selected as the parameters of the detected object.

The bounding box provides the needed information for the pose estimation of the object. First, based on the now known image-coordinates and size of the bounding box, the centre of the object can be defined and will also be the origin of the object coordinate system. These image coordinates in pixel size will later be converted to real life references. Afterwards, the size of the bounding box can then be used to calculate the distance between the camera and the object.

Depending on the use case and the object, the third parameter is the orientation of the object. To get the orientation, the bounding box must be adapted to enclose the smallest area of the detected contours. The work piece for this paper was defined as a cylinder, resulting in the orientation being irrelevant. With this parameter, it is possible to filter false detection based on the bounding box. If the detected bounding box is not a square, the centre of the coordinate system will not be in the centre of the object and therefore must be discarded. With the now known parameters, (1) can be used to calculate the Z-coordinate of the object which represents the distance between the camera and the surface of the object.

$$Z = \frac{f \cdot width_{real}}{width_{image}} \tag{1}$$

This equation is based on the knowledge of the size of the object ($width_{real}$) and the width of the object ($width_{image}$). The proposed linear calibration is performed using (2).

$$x = \frac{(y - d)}{k} \tag{2}$$

This results in the correction of the measurement y with the linear calibrated parameters k and y. To perform no linear calibration the default values are d=0 and k=1. To calibrate the system and to adjust the parameters, a polyfit trendline calculation from the NumPy library [15] is performed based on known measurements from an experimental setup. To calculate the X and Y-coordinates, (3) and (4) are used.

$$X = \frac{u \cdot Z}{c_x \cdot f_x} \tag{3}$$

$$Y = \frac{v \cdot Z}{c_y \cdot f_y} \tag{4}$$

The same linear calibration, that was deployed for the Z-coordinate, is also implemented for the X and Y calculation. To make the detection compatible with different robot systems, the ROS tf package [16] is used as an interface between different robot systems. This allows for the possibility to define the coordinate systems for the camera and the object. Knowing that, a camera model can be implemented in a running robot operating system and the tf package will take care of the transformations between the different coordinate frames, calculating a direct transformation between the object and robot coordinate systems. For example, this is used in the experimental setup to calculate the transformation between the now detected object and the robot's end effector.

For the use case, the camera can be mounted virtually in reference to the robot system using RVIZ [17]. Different coordinate systems represent the different parts of the setup. This can be helpful to align the robot coordinate frame with the camera coordinates. The system was implemented and tested on ROS Noetic Ninjemys [3], OpenCV 4.4.0 [2] and NumPy 1.16.6 [15].

4. Experimental setup and analysis

To test the accuracy of the vision system, the camera is mounted parallel to the tabletop (see Fig. 2). The distance between the vertical camera and the use case object that must be recognized, resulting in a distance of 69cm. For the vertical (Z-Axis) measurements, the object was positioned at ten different places on the tabletop. At each position, 100 measurements were recorded periodically with a frequency of 10Hz using a tf-listener to intercept the frame transformations between object and camera. This data was then used for analysis and the proposed linear calibration. The same measurements as before were performed to evaluate the optimized system. For the X- and Y-axis, the object was placed along the desired axis. Then it was moved in 4cm intervals. Each position was recorded 100 times and evaluated.

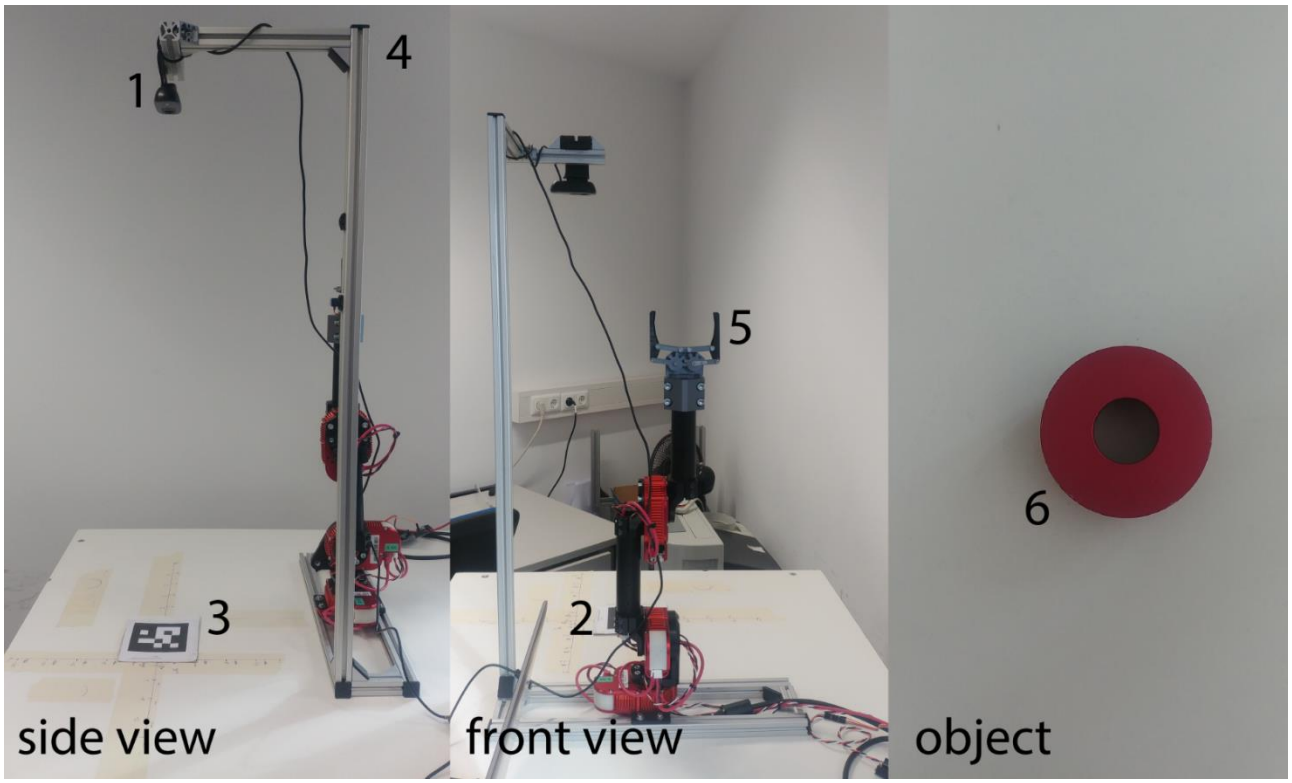

Fig. 2. Experimental Setup of the Vision System in reference to the designated Use Case. 1. webcam, 2. robot arm, 3. expected pick up location, 4. frame, 5. end effector, 6. object

These measurements were processed the same way as the Z-axis and then re-evaluated with the same measurements as before. The X-measurements were performed over a distance of 36cm and the Y-measurements were carried out over a distance of 32cm. Fig. 3 shows the calculated trendline (red) compared to a perfect system (green) that has no error.

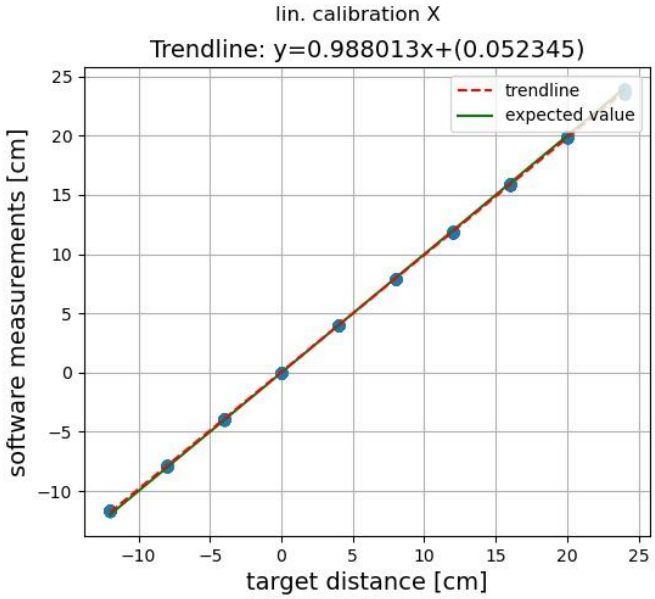

Fig. 3. Linear calibration diagram for calculating the corresponding parameters (d,k)

The difference in error is shown in Fig. 4. The trendline notation follows (3). The results are listed in table 1. The Z-axis calculation is straight forward since the measurement should always be 69cm. Therefore, the average was taken of the measurements and subtracted from the defined height. The raw calculation of the Z-distance had an average error of about 0,035mm with a scattering around 0.5565. The X and Y-coordinates had a bigger average error of 0.1194cm and 0.1531cm, but a smaller mean variation compared to the Z-axis.

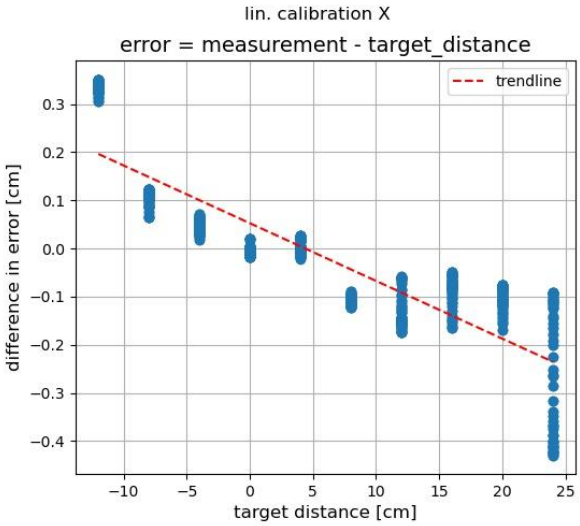

Fig. 4. Difference in error based on the linear calibration for the X-coordinates

measurement	result (mean)	difference	Standard deviation
Z	69,035cm	0,035cm	0,5565
calibrated Z	67,587	-1,413	1,9194
X	not calculated*	0,1194cm	0,08665
calibrated X	not calculated*	0,08889cm	0,06657
Y	not calculated*	0,15307cm	0,11442
calibrated Y	not calculated*	0,1465cm	0,11429

Table 1. Data Evaluation

*Mean measurements were not calculated because of the measurement type for X and Y. For the difference and scattering, the individual measurement was subtracted from the corresponding step (4cm intervals). The absolute value was used for comparison/further calculations. Corresponding Steps/Coordinates in X: [-12, -8, -4, 0, 4, 8, 12, 16, 20, 24], Steps/Coordinates in Y: [-20, -16, -12, -8, -4, 0, 4, 8, 12]

5. Discussion

This paper proposed a simple colour-based object detection for different robotic tasks, including a linear calibration method, with focus on a specific pick and place use case. A standard RGB-Webcam stream was used to detect a certain colour and obtain information which then was filtered and evaluated based on the predefined conditions of the system. ROS enables the processing of the calibrated camera stream and the interface to the pick and place robot arm. A coordinate transformation is used to pass the calculated world coordinates to the robot arm that places the detected object into a CNC-mill. The experimental setup was designed for a mobile robot to deliver the object that must be manipulated. Therefore, the vision system is needed to compensate for possible inaccuracy of the mobile robot, since the drop-off location can vary with each cycle.

During the tests an accuracy of around 0.12cm (optimized) was achieved in the X/Y-plane, which resembles the tabletop of the setup where the object rests. The average Z-distance accuracy for the system resulted in around 0.035cm without any optimization and a transformation publishing frequency of around 7Hz. The optimization for the Z-axis resulted in a false compensation and therefore distorted the measurements and had to be discarded. The overall accuracy for the system was sufficient to successfully grab the object and place it into the CNC-mill.

For more complex object shapes, the orientation might be needed. Simple orientation tests with a square object have been performed. The yaw orientation was successfully calculated but no accuracy measurements were performed, since the robot end effector was designed to grab a cylinder. Given the restricted parameters of the setup, the calculation of the orientation is promising and is worth pursuing in future work. Although the vision system and its accuracy are dependent on the surrounding lighting conditions, the advantages of a lightweight, flexible, and accurate setup should not be disregarded when designing a robot station.

6. Acknowledgments

This work was supported by the city of Vienna (MA23 – Economic Affairs, Labour and Statistics) through the research project AIAV (MA23 project 26-04)

7. References

[1] Gacovski, Z. (2011). Mobile Robots: Current Trends
[2] Bradski, G. (2000). The OpenCV Library Dr. Dobb's Journal of Software Tools
[3] Quigley, M.; Gerkey B.; Conley K.; Faust J.; Foote T.; Leibs J.; Berger E.; Wheeler R.; Ng A. (2009). ROS: an open-source Robot Operating System, ICRA workshop on open source software
[4] Andhare, P. & Rawat, S. (2016). Pick and place industrial robot controller with computer vision, International Conference on Computing Communication Control and automation (ICCUBEA)
[5] Khan, F. S.; Anwer, R. M.; Weijer, J. v. d.; Bagdanov, A. D.; Vanrell, M. & Lopez, A. M. (2012). Color attributes for object detection, IEEE Conference on Computer Vision and Pattern Recognition
[6] Soendoro, D. & Supriana, I. (2011). Traffic sign recognition with Color-based Method, shape-arc estimation and SVM, Proceedings of the 2011 International Conference on Electrical Engineering and Informatics
[7] Bishop, C. M. (2006). Pattern recognition and machine learning, New York, NY
[8] Kadouf, H. H. A. & Mustafah, Y. M. (2013). Colour-based Object Detection and Tracking for Autonomous Quadrotor {UAV}, {IOP} Conference Series: Materials Science and Engineering, Bd. 53
[9] Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement, arXiv
[10] Jiao, Z.; Zhang, Y.; Xin, J.; Mu, L.; Yi, Y.; Liu, H. & Liu, D. (2019). A Deep Learning Based Forest Fire Detection Approach Using UAV and YOLOv3, 1st International Conference on Industrial Artificial Intelligence (IAI), pp. 1-5
[11] Rajendran, S. P.; Shine, L.; Pradeep, R. & Vijayaraghavan, S. (2019). Real-Time Traffic Sign Recognition using YOLOv3 based Detector, 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-7
[12] Lu, X. X. (2018). A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation, Journal of Physics: Conference Series
[13] Jamil, N.; Sembok, T. M. T. & Bakar, Z. A. (2008). Noise removal and enhancement of binary images using morphological operations, International Symposium on Information Technology
[14] Topal, C.; Akınlar C. & Genç, Y. (2010). Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection, 20th International Conference on Pattern Recognition
[15] Harris, C. R.; Millman, K. J. & Walt S. J. v. d. (2020). Array programming with {NumPy}, Nature, pp. 357-362
[16] Foote, T.; Marder-Eppstein E. & Meeussen, W. (2017). Available from: http://wiki.ros.org/tf, Accessed: 2021 03-16
[17] PickNikRobotics. (2020). Rviz Visual Tools: C++ API wrapper for displaying shapes and meshes in Rviz
[18] Deac, G. C.; Deac, C. N.; Popa, C. L.; Ghinea M. & Cotet, C. E. (2017). Machine Vision in Manufacturing Processes and the Digital Twin of Manufacturing Architectures, Proceedings of the 28th DAAAM International Symposium, pp.0733-0736, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/28th.daaam.proceedings.103