

AI-Assisted Drone Localization of Arbitrary Objects using Aruco Markers

Daniel Czurko, Gabor Feher

Department of Telecommunications and Media Informatics
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics

Abstract—This conference paper presents a novel approach for localizing arbitrary objects using drones and AI assisted image processing. The system utilizes aruco markers, which are small square markers that are easily identifiable by image processing algorithms, in combination with AI to accurately determine the location of the objects in real-time. The system was tested in a variety of environments and demonstrated high accuracy and robustness. The results of this research have potential applications in areas such as warehouse management, search and rescue, and precision agriculture.

Keywords—object localization, object detection, AI, aruco codes, Drone

I. INTRODUCTION

In this work we will present a method to determine the spatial position of an arbitrary object using a camera and a special fiducial marker, the aruco code. The method can work with any camera as long as the camera has a good view of both the object and the aruco code [1]. The method has three main components. The aruco code recognition-, the object recognition-, and the spatial coordinate computation component. In this paper we will describe the exact operation of the method and then test its accuracy through measurements.

In this paper we would like to show that it is possible to calculate the position of an arbitrary object using an ordinary camera and an aruco code. Later we would like to optimize this method and examine if we can improve the accuracy of the localization component by changing the algorithms, like the triangulatePoints, camera calibration algorithm or using a newer version of YOLO.

II. RELATED WORK

3D reconstruction is the process of creating a 3D model of an object or scene using multiple 2D images or other data. There are several sensors and algorithms that can achieve 3D reconstruction and object localization. Algorithms such as the SFM [2]–[5] (Structure from Motion) algorithm uses image sequences to create 3D structures and consists of multiple steps and algorithms. The stereo camera, which uses two or more lenses, can capture 3D images. Sensors such as ToF (Time of Flight), LIDAR (Light Detection and Ranging) and structured light can also be used to obtain 3D data. (more info in the Learning OpenCV 3 book: [6]).

Our method relies on an object detection algorithm to identify objects on an image. Among machine learning-based object recognition algorithms, there are three common solutions:

- R-CNN (Region proposal Convolutional Neural Network) [7]
- YOLO (You Only Look Once) [8]
- SSD (Single Shot multiBox Detector) [9]

In the article describing YOLOv4 [8] a comparison shows that R-CNN is very slow, while YOLO and SSD are fast. If the accuracy values are taken into account, however, SSD is very inaccurate while YOLO and Faster R-CNN are similarly accurate. Based on these results, the YOLOv4 algorithm is an optimal solution and we have chosen it to implement object detection in our work. You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm. YOLO uses the one-stage detector strategy, which means it simultaneously tries to figure out the bounding box coordinates and corresponding class label probabilities. This object detector is able to identify 80 different objects. While developing our method the YOLOv7 [10, p. 7] was the latest YOLO version. We have tested YOLOv7 and there was minimal difference compared to YOLOv4 so we decided to use the well-tried YOLOv4 to develop our method.

The aruco detection algorithm is another important part of our method. This algorithm calculates the precise location of the camera relative the the aruco code. Aruco (Augmented Reality University of Cordoba) codes [1] are small black and white markers similar to the QR codes. Figure 1 shows two aruco codes and figure 2 shows how the aruco detection algorithm detected the aruco codes.

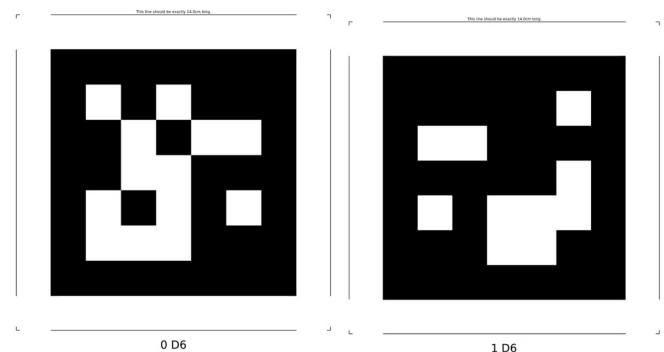


Figure 1: Aruco codes

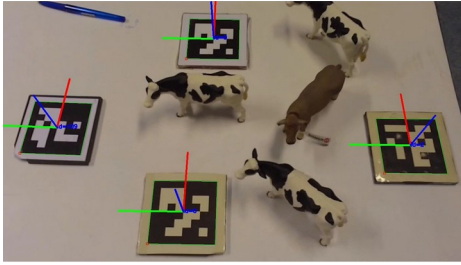


Figure 2. The detected aruco codes

Given the physical parameters of the camera, the aruco detection algorithm can very accurately determine the spatial position of the camera relative to the aruco code. This is done by calculating the relation between the four corner of the real aruco code and the four corner of the aruco code detected on the camera image.

III. METHOD

The method we designed in this work aims to be able to determine the spatial position of a pre-selected object using two or more cameras and aruco codes. OpenCV provides us with a function called `triangulatePoints` [11], which can triangulate the spatial equivalent of points on the camera image. To do this, it only requires the points of the camera images that are connected and the projection matrix associated with the camera images.

Finding the points in the camera images that are connected is done by detecting common objects in the images. The objects will be recognized by the YOLOv4 object recognition algorithm and this algorithm will draw the bounding boxes as well. The center of this bounding box will represent the object in the calculations. The centers of the objects found in each shot will be the related points that are used in the OpenCV `triangulatePoints` function. There may be a problem if more than one of the same object is found in the image. In such a case, it is a complex image processing task to connect the related objects due to the arbitrary and possibly time-varying positions of the cameras. Solving this problem is not part of this paper. The method is designed so that only one object of a given species will be present.

A projection matrix is calculated using the camera calibration algorithm (Chapter 18, Learn OpenCV 3 book [6]) and the aruco detection algorithm [1], [12].

A big advantage of this method is that the relative position of the camera shots taken can be arbitrary and can change during operation. There are no constraints like in stereo cameras, where the position of the two cameras must be fixed. The way the method works is that for each frame we know the position of both cameras relative to a fixed point in the aruco code (and can therefore easily calculate the relative position of the two cameras to each other). Hence it does not matter how the two cameras are positioned relative to each other.

On the other hand, this method requires at least two viewpoints. One option is to use two cameras and place them in an arbitrary position, for example next to each other, so that they have a good view of the objects and the aruco code. A difficulty of this implementation may be the synchronization of the images from the two cameras.

The other option is to use one camera and have that camera take a shot first from one position and then from another position. In this case, an important criterion is that the environment, the objects under investigation, should not change between the two shots. If you have a relatively fixed, non-moving environment, this is a simpler solution.

In both the two-camera and single-camera implementations, the cameras must always see not only the object, but also at least one aruco code. This is a difficulty of this method and can lead to the placement of many aruco codes.

Figure 3 shows a high level architectural overview of the method.

The method consists of three main components. The aruco code detection -, the object detection - and the spatial coordinates calculating component. A camera, (in figure 3. it is mounted on a drone) takes a video of the object (in the example below it is the cow) and the aruco code. This recording is sent to the localization component, which is composed of three subcomponents: aruco detection, object detection and spatial coordinates calculating component. The video recording is processed in parallel by the aruco detection and object detection components and the computed data is passed to the component responsible for the computation of the spatial coordinates. This component calculates the spatial coordinate of the object (the cow in the example) in the coordinate system of the aruco code.

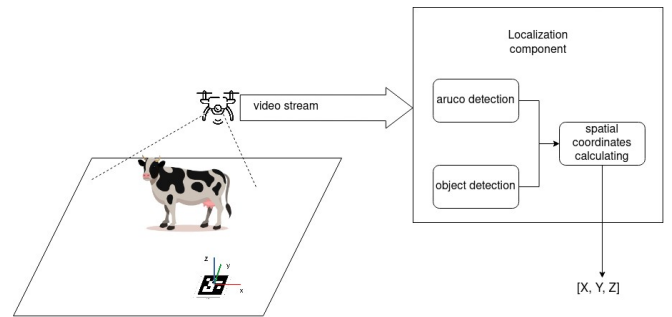


Figure 3: An outline of how the method works

IV. MEASUREMENTS

The accuracy of the method is tested using two different measurements. These are shown in the list below:

Measurement 1, How accurately can the method determine the four objects on the ground. (In this case, we manually matched the objects in the two images.) Between the two camera shots, the drone is slightly moved sideways in one direction only.

Measurement 2, How accurately can the method determine the position of the object on the ground if the two camera images were taken by the drone from two arbitrary positions.

To make the measurements, we first took the camera shots and then measured the position of the objects manually from the center of the aruco codes using a ruler. Then the spatial coordinate of the object was also calculated using the method we developed and these two values were compared. During the comparison, the two values were subtracted and the absolute value was taken. This absolute value became the accuracy of the method used in the measurement.

During the measurements we decided to use toy cows as the objects which will be localized since this object fits well to a possible agriculture use case. It is important to mention that in these measurements the cow was just an example we could have used other objects from the available 80 objects that YOLO can detect. The orientation of the chosen object is not relevant as long as the object detection algorithm, YOLO could detect it.

A. Measurement 1

In the first measurement, we tested how accurately the method could detect objects placed in different positions on the ground. Between the two camera shots, we moved the drone sideways by only about 10 centimeters.

Figure 4 shows the measurement setup with the drone, the aruco code and the four cows



Figure 4: First measurement setup

Figures 5 and 6 show the objects detected by the object recognition algorithm in the two camera images.

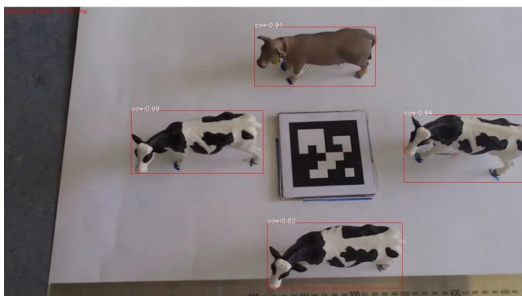


Figure 5: The first image processed by the object detector

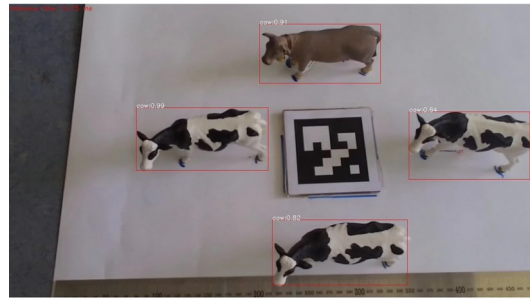


Figure 6: The second image processed by the object detector

The Table 1 shows the accuracy values. The columns of the table show each coordinate while the rows correspond to each object.

Table 1: The accuracy (difference) for each toy cow in the first measurement

	x difference [meter]	y difference [meter]	z difference [meter]
1	0.0064	0.0122	0.0078
2	0.0056	0.0078	0.0129
3	0.0080	0.0129	0.0009
4	0.0007	0.0166	0.0117

The table shows that in most cases there are only errors of the order of millimeters, therefore the position of the objects does not affect the accuracy.

B. Measurement 2

Finally, in the second measurement, we tested the accuracy of the method when the object is on the ground, but the two camera shots were taken in arbitrary position (of course both the aruco code and the object are clearly visible in the image). Figures 7 and 8 show the measurement setup at the time of the first and second shots, and Figures 9 and 10 show the images processed by the object detector. It can be observed that the two images were taken from completely different angles.

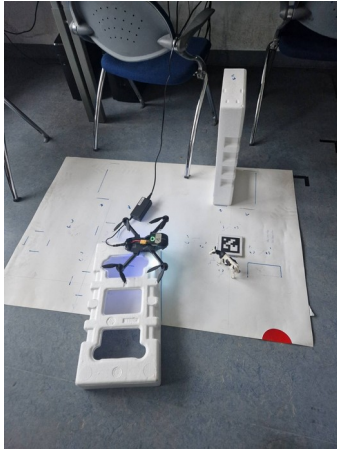


Figure 7: Arrangement of the second measurement when taking the first picture

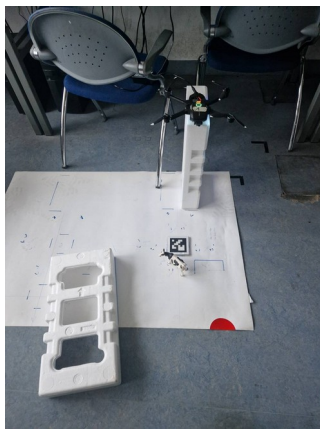


Figure 8: Arrangement of the second measurement when taking the second picture

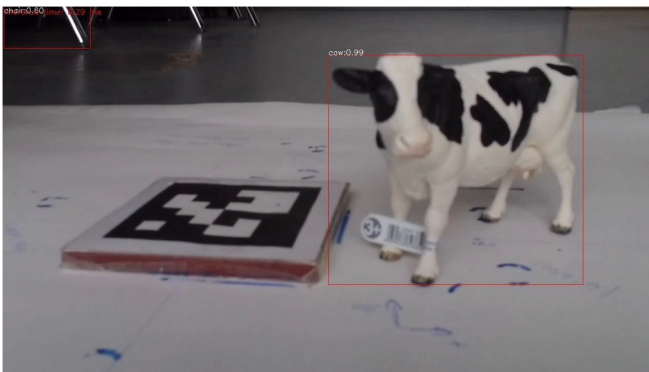


Figure 9: The first image processed by the object detector for the second measurement

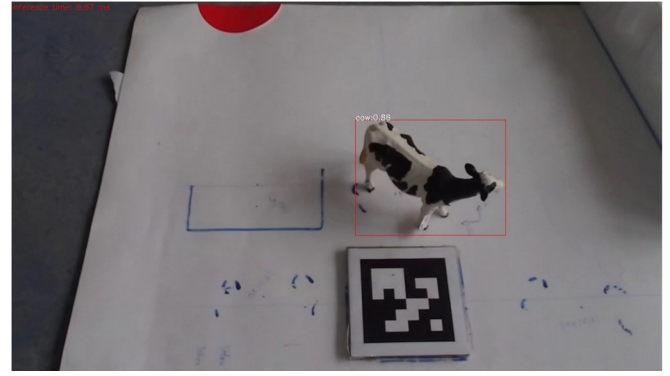


Figure 10: The second image processed by the object detector for the second measurement

The differences (accuracy values) are shown in Table 2. The method was accurate in this case as well.

Table 2: Accuracy (difference) of the method for the third measurement

x difference [meter]	y difference [meter]	z difference [meter]
0.0039	0.0159	0.0163

It can be seen from the above measurements that neither the number of objects, the spatial position (planar and elevation) nor the position of the cameras have a significant effect on the accuracy of the method.

CONCLUSION

In this paper, we have developed a method that can determine the spatial position of an arbitrary object using a camera and an aruco code. Through several measurements, we have shown that the method is able to work accurately and robustly with changes in the number of objects, their position, and the position of the cameras.

REFERENCES

- [1] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, ‘Speeded up detection of squared fiducial markers’, *Image Vis. Comput.*, vol. 76, pp. 38–47, Aug. 2018, doi: 10.1016/j.imavis.2018.05.004.
- [2] J. L. Schonberger and J.-M. Frahm, ‘Structure-from-Motion Revisited’, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 4104–4113. doi: 10.1109/CVPR.2016.445.
- [3] R. Gherardi, M. Farenzena, and A. Fusiello, ‘Improving the efficiency of hierarchical structure-and-motion’, in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 1594–1600. doi: 10.1109/CVPR.2010.5539782.

- [4] V. M. Govindu, ‘Combining two-view constraints for motion estimation’, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 2, p. II–II. doi: 10.1109/CVPR.2001.990963.
- [5] S. Ullman and S. Brenner, ‘The interpretation of structure from motion’, *Proc. R. Soc. Lond. B Biol. Sci.*, vol. 203, no. 1153, pp. 405–426, Jan. 1979, doi: 10.1098/rspb.1979.0006.
- [6] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, 1st edition. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly Media, 2016.
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ‘Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation’, in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.
- [8] A. Bochkovskiy, C.-Y. Wang, and H. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020.
- [9] W. Liu *et al.*, ‘SSD: Single Shot MultiBox Detector’, in *ECCV*, 2016. doi: 10.1007/978-3-319-46448-0_2.
- [10] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, ‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors’. arXiv, Jul. 06, 2022. doi: 10.48550/arXiv.2207.02696.
- [11] ‘OpenCV: Triangulation’. https://docs.opencv.org/4.x/d0/dbd/group__triangulation.html#ga211c855276b3084f3bbd8b2d9161dc74 (accessed Oct. 25, 2022).
- [12] D. Jurado-Rodriguez, R. Munoz-Salinas, S. Garrido-Jurado, and R. Medina-Carnicer, ‘Design, Detection, and Tracking of Customized Fiducial Markers’, *IEEE Access*, vol. 9, pp. 140066–140078, 2021, doi: 10.1109/ACCESS.2021.3118049.