

Real-Time Tracking and Distance Measurement of Opencv Aruco Marker Using Webcam

Ali Shuja Sardar

Department of Electrical Engineering and Computer Science, University of Stavanger Stavanger, Norway

DOI : <https://doi.org/10.51583/IJLTEMAS.2025.1401034>

Received: 26 April 2023; Revised: 04 February 2025; Accepted: 06 February 2025; Published: 18 February 2025

Abstract: Object tracking and distance measurement play a vital role in robotics and drones. It is often challenging to measure the distance of a target object in an environment by just using a single-vision camera. This paper discusses the development of a fiducial marker-based object tracking and distance measurement system. Fiducial marker detection uses the ArUco method based on the OpenCV library and Python 3.x. The hardware consists of an Arduino, a single-vision camera, and two servos as an actuator for tracking. A mathematical equation is derived to measure the real-time distance of the marker by using a single camera and adjusting the frame size and the camera output colors to increase the detection method's performance. OpenCV is used to find the center coordinates of the bounding box, and a tracking algorithm is applied to give pan/tilt angles to the servos. Finally, to stabilize the tracking mechanism, an acceptable error is defined. The accuracy of the system is measured by performing 100 trials, and the results show a good accuracy for the system when tracking the ArUco marker. The system is highly beneficial for indoor mobile robot navigation and drone applications.

Keywords: OpenCV, Fiducial marker, Python, Computer vision, Robotics, Aruco marker.

I. Introduction

Real-time distance measurement and tracking are of significant importance in the field of robotics and computer vision. The distance of the object is very useful, not only for robot navigation and localization [1] but also for drones to track objects. Distance measurement is also a very useful module for modern autonomous and dynamic systems [2].

There are many distance-finding sensors available in the market, for example, ultrasonic, LIDAR, and infrared sensors, etc. Some of them are expensive, and only a single sensor can increase the cost of the whole project. The output of the ultrasonic sensor can change with temperature. The wave of the infrared sensor can damage eyes at high power and LIDAR sensors are very expensive.

Ultrasonic and infrared sensors can measure the distance of an object but fail to provide the distance of a specific or a known object in a real-time environment. Multiple previous works have been done to track the known object in the environment. Tracking is processed by fetching the features of the target from an image. The features include points or lines [3]. Fiducial markers are useful in applications such as augmented reality, virtual reality, and robot location. Fiducial markers are used as a unique reference that can be easily located in any environment [4]. There are multiple marker detection algorithms, which can be used to detect markers [5] such as ARToolkit [6], [7], AprilTags [8], and ArUco [9]. So, fiducial markers were selected as a target in the environment. Single, stereo, and multiple camera arrangements can be used to perform vision actions [10]. Jun et al. [11] made a planer-based marker tracking system for a large working space. It consists of two fixed cameras that give robust pose estimation, but distance estimation is still a challenge, and fixed camera setup can only track marker within their field of view. Ababsa and Mallem [12] used the corner information of the marker to estimate the pose of the camera, but the system gets worse when the direction of the camera and the marker is almost perpendicular. Latifah et al. [13] proposed a pan/tilt tracking mechanism. Object placed 10 cm away from the camera and frame size 320x240px chosen. But this is a less robust approach because if the distance between the object and camera increases or decreases, then the system may not work correctly. Continuously tracking marker in a real-time environment using a single camera is still a challenge.

In this paper, a single vision camera-based solution is proposed that detects the fiducial marker using ArUco, measures the distance of the marker from the camera lens, and also tracks it continuously by using pan-and-tilt mechanism without any specified distance approach. The pan and tilt mechanisms have 2 degrees of freedom. A pan/tilt platform is designed to keep the ArUco marker in the camera's field of view [14] by using two servo motors. A mathematical model is designed to measure the distance of the marker from the camera lens.

ARUCO Marker Detection algorithm

ArUco markers consist of the black border (as shown in fig. 1) that contains an inner white binary matrix that provides a unique identifier. The black border allows for fast detection in any environment. The size of the marker determines the size of the internal matrix. The detection of a marker is the very first step. In this step, the camera image output was analyzed to find the square shape in the field of view. All that begins with threshold the image. After that, contours extract from the threshold image and discards which are not approximate to square and not convex. Moreover, extra filtering was also applied to the image to remove very small or too large contours, and to remove contours that are very close to each other. After squares detection, the next step is to find out if they are actual markers. For this purpose, inner codification analyzes, and perspective transformation are applied to get the marker in its canonical form. After this step, the threshold applies to the canonical image to separate the white and black bits. The image is

divided into different cells according to their size, and black and white pixels count, to know if the image is black or white. Finally, bits are analyzed to find if the marker belongs to a specific dictionary.

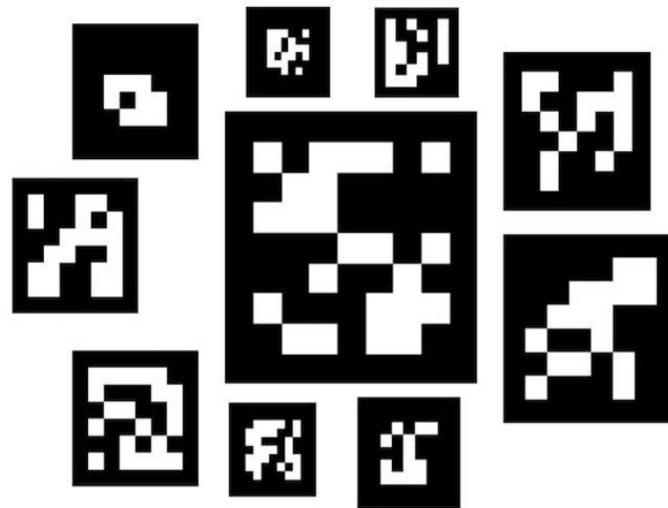


Fig. 1: Aruco Marker

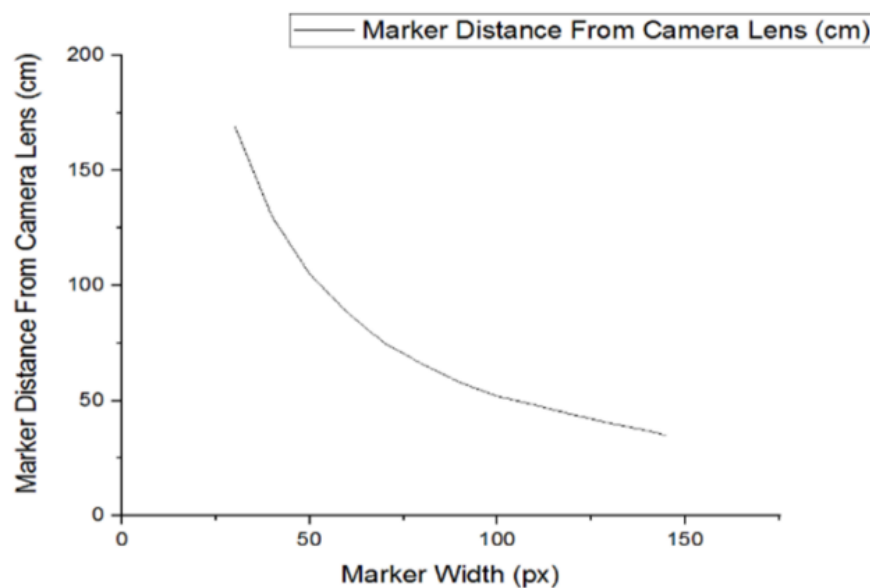


Fig. 2: Relationship between the distance of Aruco marker and marker's width

II. Methodology

This research divided into three sections, one is fiducial marker detection using OpenCV ArUco, the second is distance measurement, and finally real-time marker tracking. All these sections are integrated with each other and call each other back one by one. Marker detection is performed using the OpenCV [15] library, distance can be measured using a mathematical model, and finally, tracking can be performed using a center base method in which coordinates of the detected marker rectangle and bounding box can be used.

Data Acquisition

It is found that a relationship exists between the marker and the camera lens. The sample data show that marker width with different perspectives is directly proportional to the distance of the marker from the camera lens. Data collected manually using the different widths of the marker. Marker width and distance of the marker from the camera lens shown in fig 2. It is noted that the graph of marker width and marker distance from the camera lens is significantly identical.

Formulation of the distance measurement equation

The polynomial provides the best approximation of the relationship between the dependent and independent variables. ArUco marker width is an independent variable, and distance is the dependent variable. The system was tested with linear regression and a 2-degree polynomial equation, but the results were not very precise and accurate. Finally, it is found that the 3-degree polynomial

is a perfect solution to this problem. To formulate the distance equation from the measured data, we have used various values as input of the matrix in Table I.

Table 1: Numeric Data Table

| A | Value | A | Value |
|-----------------|------------|-----------------|----------------|
| A ₁₁ | 13 | A ₃₁ | 88713 |
| A ₁₂ | 947 | A ₃₂ | 10261301 |
| A ₁₃ | 88713 | A ₃₃ | 1367095653 |
| A ₁₄ | 10261301 | A ₃₄ | 198334000000 |
| A ₂₁ | 947 | A ₄₁ | 10261301 |
| A ₂₂ | 88713 | A ₄₂ | 1367095653 |
| A ₂₃ | 10261301 | A ₄₃ | 198334000000 |
| A ₂₄ | 1367095653 | A ₄₄ | 30275900000000 |

Using system of linear equations:

$$Ax = B, \quad x = A^{-1}B$$

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1165 \\ 67765 \\ 4927155 \\ 459621145 \end{pmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 282.637090 \\ -5.209615 \\ 0.038970 \\ -0.000101 \end{bmatrix}$$

Using the general form:

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$$

$$D = 282.6371 - 5.2096x + 0.03897x^2 - 0.0001x^3 \quad (1)$$

Camera to Marker Distance Measurement

Marker-to-camera distance measurement can be accomplished by getting the predefined distance values according to the width of the marker. These values can be used to generate a 3-degree polynomial equation. Marker-to-camera distance measurement algorithm as shown in Fig. 3.

- Turn on the webcam or the external camera.
- Convert the RGB frame to a gray scale frame.
- Detect the marker in the frame and calculate the width of the detected marker.
- Finally, calculate the distance using the distance equation 1.

Hardware for Tracking

For tracking of the ArUco marker, the Arduino UNO is used. Output pins 4 and 5 used for two Mg996r servomotors as shown in Fig. 4b. MG996r is used mainly in robotics applications. In this research, two servos were used to cover the x and y axes. Python programming is used to send input signals to Arduino UNO via USB. A simple 5-megapixel USB webcam was chosen due to the low price and availability. Finally, a computer used has Arduino IDE. Fig. 4 shows the flow of the input.

Aruco Marker Tracking

After detection and distance measurement, it is necessary to track the marker in a real-time environment to make it more effective and applicable in robotics and drones.

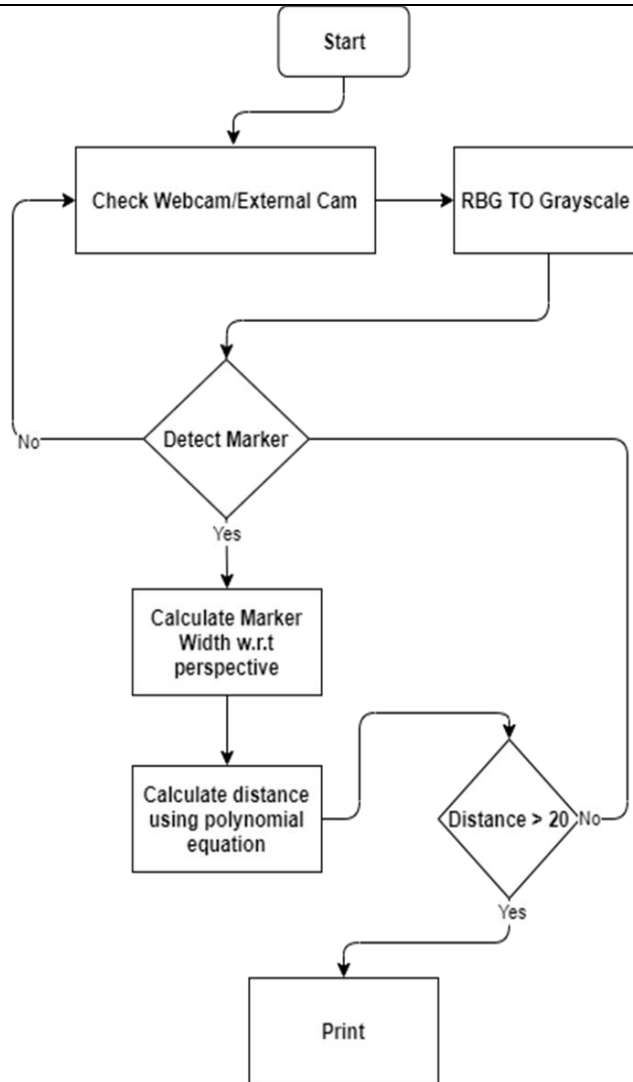


Fig. 3: Distance measurement flow chart

The resolution of the captured video must be set to track the marker. So, in this research paper, the frame set is 640 x 480 pixels. Marker tracking can be achieved by using the bounding box provided by the marker detection method. This bounding box is ROIW (regions of interest) [16]. A bounding rectangle is drawn around the detected marker so that it can be used to calculate the current coordinates of the marker. OpenCV function boundingRect() is used to get the width, height and x, y coordinates of the bounding box to calculate the center coordinates of the detected marker.

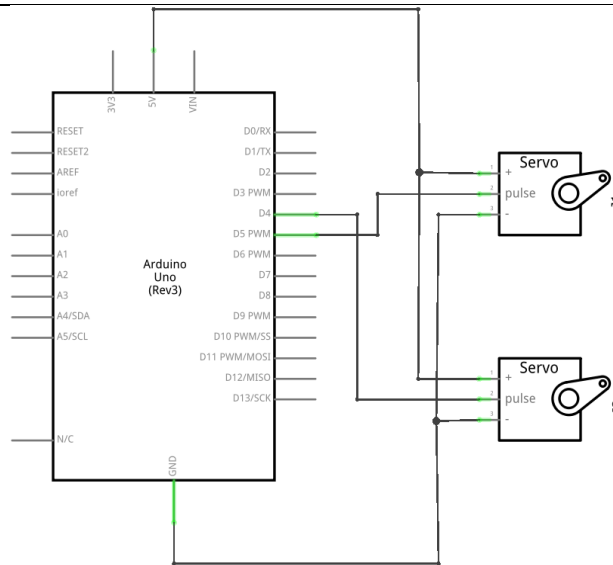
Where, units in pixels, x_1 is the initial marker coordinate on horizontal, w_1 is the width of the detected marker, y_1 is the initial marker coordinate on vertical and h_1 is the height of the detected marker. HorizontalCenter is the center value of the marker coordinate on horizontal, and VerticalCenter is the center value of the marker coordinate on vertical. The center is the center coordinate of the detected marker. In this research, the maximum and minimum angle is set for tilt and pan servos. The maximum and minimum values of pan servo are 160 and 20 respectively and the angle is 20 to 100 for tilt servo. Video frame size is divided into middle x and middle y: The middle screen x is 320px and the middle screen y is 240px. The video frame is divided into four main portions. Top left, bottom left, top right and bottom right as shown in Fig. 5. The mechanism is that the Python code calculates the center point of the marker using 2 and the center point is compared to the center of the screen.

$$\text{Horizontal Center } (x) = \text{int} \left(x_1 + \frac{w_1}{2} \right)$$

$$\text{Vertical Center } (y) = \text{int} \left(y_1 + \frac{h_1}{2} \right)$$



(a)



(b)

Fig. 4: Hardware: (a) Input flow (b) Hardware Design

$$\text{Center} = (x, y) \quad (2)$$

If the marker center is in the left portion of the frame, then Arduino receives the instruction to increase the angle of the pan servo. If the marker center is at the right side of the frame, then the pan servo angle decreases. The same happens for tilt servo, which works for the top and bottom field of view. If the vertical center of the marker is at the left or right side of the center of the screen, the tilt servo angle value will be added or subtracted respectively to follow the marker. If the horizontal center of the marker is at the top or bottom side of the center of the screen, then the pan servo angle will be subtracted or added, respectively, to follow the marker. The program sends angle data for the tilt / pan servo to Arduino after recognizing the change in position as shown in fig 7. Data transfer using serial communication between the python and Arduino board.

Marker Detection and Tracking Code

ArUco algorithm is used to detect the markers using a webcam. Python code was written using Spyder IDE which supports Windows OS. To write the code for Arduino, the Arduino IDE is used, which supports the C programming language. Listing 1 shows the way to detect the marker in a real-time environment. Listing 2 was used to find the distance and to track the marker. The tracking algorithm communicates with Arduino to provide a suitable direction for tracking purposes.

Stabilization

It is found that the system is not stable due to small changes from the center of the screen to the center of the bounding rectangle. So, the acceptable error for the center of the screen is 100. Stabilizes and smooths the movement of the servo and tracking. The servo angle value, which added

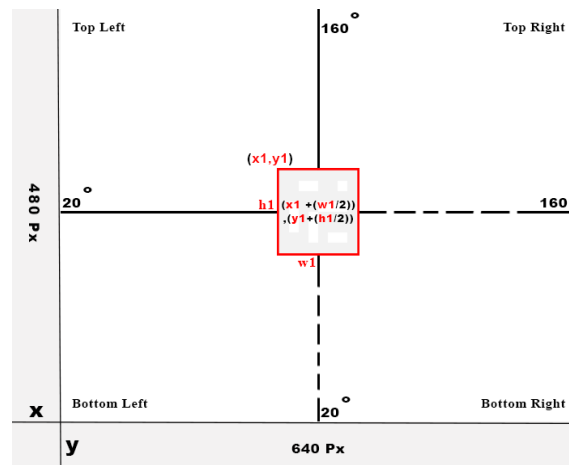


Fig. 5: Distance measurement flow chart

or subtracted every time position change, is set to 4 which makes tracking more stable.

Listing 1: ArUco Marker Detection

```
capture = cv2.VideoCapture(0, cv2.CAP_DSHOW);
parameters = cv2.aruco.DetectorParameters_create()
dictionary=cv2.aruco.

    → Dictionary_get(cv2.aruco.DICT_6X6_250)
while True:
    NoFrame, Frame = capture.read()
    FrameGray = cv2.cvtColor(Frame, cv2.COLOR_BGR2GRAY)
    MarkerCorners, MarkerIds, RejectedCandidates
        → = cv2.aruco.detectMarkers(FrameGray,
        → dictionary, parameters=parameters)
    cv2.aruco.drawDetectedMarkers(Frame,
        → MarkerCorners,MarkerIds,(255, 128, 10))
```

Listing 2: Real-time ArUco Marker distance measurement and Tracking

```
capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
midScreenX =320
midScreenY =240
stepSize = 4
servoTiltPosition =70
servoPanPosition =90
sendcommand=False
while True:
    if len(MarkerCorners) !=0:
        x,y,w,h = cv2.boundingRect(MarkerCorners[i])
            → NoFrame, Frame = capture.read()
        xx = int(x+(w/2))
        yy = int(y+(h/2))
        if yy<(midScreenY-midScreenWindow):
            if(servoTiltPosition<=100):
                servoTiltPosition+=stepSize
                sendcommand=True
        elif yy>(midScreenY+midScreenWindow):
            if(servoTiltPosition>=20):
                servoTiltPosition-=stepSize
                sendcommand=True
        if xx>(midScreenX+midScreenWindow):
            if(servoPanPosition>=20):
                servoPanPosition-=stepSize
                sendcommand=True
```

```






elif xx<(midScreenX-midScreenWindow):
    if(servoPanPosition<=160):
        servoPanPosition+=stepSize
        sendcommand=True
    if sendcommand:
        data = "X{0:f}Y{1:f}Z".format(
            → servoPanPosition,servoTiltPosition
        arduino.write(data.encode())
        sendcommand = False
    distance = 282.63 -5.20*w +0.038*(w**2)
            → -0.000101*(w**3)
    
```

III. Experimental Results

In this research, Python 3.7 and OpenCV 4.4 were used to process real-time video and send position data to Arduino. Arduino controls the servos, by getting the position indications from the Python program.

- Marker detected using OpenCV ArUco marker
- For distance calculation, a 3-degree polynomial equation derived
- For tracking of the detected marker, center of bounding box and center of screen used
- Maximum resolution of screen setup 640x480, with the middle being 320x240
- Maximum and minimum servo rotation for tilt: 100 and 20, respectively
- Maximum and minimum servo rotation for pan: 160 and 20, respectively.

Table 2: Table showing images, bounding box details, and results

| Image | Bounding box of Marker (x1, y1, w1, h1) | Horizontal center of bounding box | Vertical center of bounding box | Result (Position) |
|---|---|-----------------------------------|---------------------------------|-------------------|
|  | (256, 205, 121, 119) | 316 | 264 | Center |
|  | (68, 45, 153, 150) | 139 | 124 | Top left |
|  | (205, 297, 158, 155) | 484 | 374 | Bottom right |
|  | (67, 205, 135, 133) | 134 | 371 | Bottom left |
|  | (407, 63, 177, 172) | 492 | 150 | Top right |

A simple 5-megapixel webcam (T1 class USB video class) is used for video acquisition. ArUco marker 200x200 printed to embed in the object. Marker glued to a box to measure the distance from the camera to the box, camera attached with two servo motors for the pan-tilt mechanism as shown in fig. 6b. Arduino received the angles from the Python program using serial communication and gave

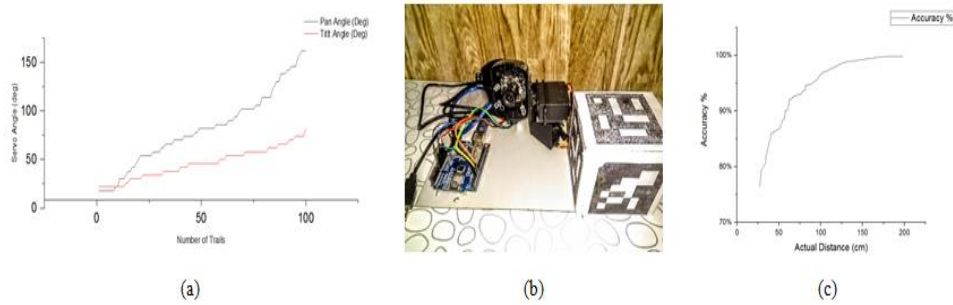


Fig. 6: System (a) 100 trails on tracking system (Pan angle and Tilt angle) (b) Tracking System (c) Accuracy after 100 trails

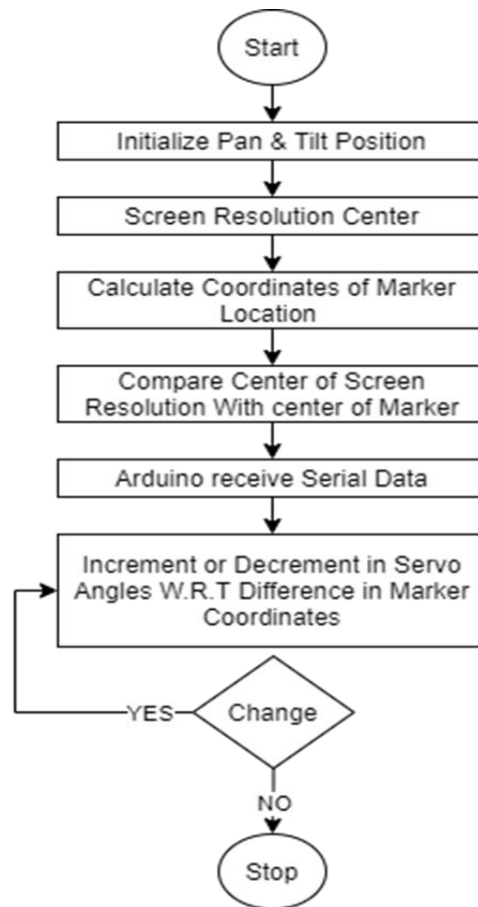


Fig. 7: Tracking flow chart

instructions to the servo to track the ArUco marker. The camera got the real-time video and detected the ArUco marker after the detection distance was measured from different ranges and the accuracy of the system was calculated. In addition, the real-time marker tracking was tested. To test the accuracy of the distance measurement system, 100 tests are performed as shown in Fig. 6c. In all these leads to 94%. During the 100 trails, the maximum distance 183 cm and the minimum distance 27 cm are measured. During the trails, the pan and tilt angles were also recorded, as shown in Fig. 6a. The rapid change in the pan/tilt angles can be seen. It is clear from Fig. 6a that the pan angle was range of 20-160 ° and the tilt angle was also within the defined range of 20-100°. Table II shows the results of the tracking positions. Five experiments were performed at different positions. The tilt / pan mechanisms worked accurately, and the camera detected and tracked the marker very smoothly. This process continuously works until the end of the tracking. Finally, the proposed system proves its simplicity, accuracy, and cost-effectiveness.

IV. Conclusion

A cost-effective and simpler algorithm increases the importance of real-time implementation. A single camera can be used for real-time object tracking and distance measurement using fiducial markers. It can be used by a robot for navigation purposes within a room environment. 100 trials were done in the system and data collected proves that the overall system is 94% accurate. The

tracking algorithm checks whether the marker center is above, below, left, or right of the center of the webcam screen. This information helps to change the position by sending angle information to Arduino for pan and tilt servos. The coordinates and size of the detected marker were used to find the center of the marker. The maximum distance measurement range depends on the environment and camera specifications, which include megapixels and low-light performance.

The system is very adaptive and can be used with ARToolkitPlus and ARTag instead of the ArUco approach. Practical importance in the fields of robotics augmented reality and in drones.

References

1. Mustafah, Y.M., Noor, R., Hasbi, H., and Azma, A.W. (2012). Stereo vision images processing for real-time object distance and size measurements. 2012 International Conference on Computer and Communication Engineering (ICCE), 659-663.
2. Hossain, M. A., and Mukit, M. (2015). A real-time face to camera distance measurement algorithm using object classification. 2015 International Conference on Computer and Information Engineering (ICCIE), 107–110. <https://doi.org/10.1109/CCIE.2015.7399293>
3. Ye, Y., Tsotsos, J., and Harley, E. (2000). Tracking a person with a pre-recorded image database and a pan, tilt, and zoom camera. *Machine Vision and Applications*, 12(1), 32–43. <https://doi.org/10.1007/s001380050122>
4. Acuna, R., and Willert, V. (2018). Dynamic markers: UAV landing proof of concept. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR), and 2018 Workshop on Robotics in Education (WRE), 496–502. <https://doi.org/10.48550/arXiv.1709.04981>
5. Saez, J. M., Lozano, M. A., Escolano, F., and others. (2020). An efficient, dense, and long-range marker system for the guidance of the visually impaired. *Machine Vision and Applications*, 31, 57. <https://doi.org/10.1007/s00138-020-01097-y>
6. Kato, H. (2002). ARToolKit: Library for vision-based augmented reality. *IEICE Technical Report*, 101(652 (PRMU2001 222-232)), 79–86
7. Kato, H., and Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR '99)*, 85–94. <https://doi.org/10.1109/IWAR.1999.803809>
8. Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2011)*, 3400–3407. <https://doi.org/10.1109/ICRA.2011.5979561>
9. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2280–2292. <https://doi.org/10.1016/j.patcog.2014.01.005>
10. Dandil, E., and Cevik, K. K. (2019). Computer vision-based distance measurement system using stereo camera view. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 1–4. <https://doi.org/10.1109/ISMSIT.2019.8932817>
11. Jun, J., Yue, Q., and Qing, Z. (2010). An extended marker-based tracking system for augmented reality. *Proceedings of the 2010 Second International Conference on Modeling, Simulation and Visualization Methods (WMSVM)*, 94–97. <https://doi.org/10.1109/WMSVM.2010.52>
12. Ababsa, F., and Mallem, M. (2004). Robust camera pose estimation using 2D fiducials tracking for real-time augmented reality systems. *VRCAI '04*. <https://doi.org/10.1145/1044588.1044682>
13. Latifah, A., Saripudin, Aulawi, H., and Ramdhani, M. (2018). Pantilt modelling for face detection. *IOP Conference Series: Materials Science and Engineering*, 434, 012204. <https://doi.org/10.1088/1757899X/434/1/012204>
14. Torkaman, B., and Farrokhi, M. (2012). Real-time visual tracking of a moving object using pan and tilt platform: A Kalman filter approach. 20th Iranian Conference on Electrical Engineering (ICEE2012), 928–933. <https://doi.org/10.1109/IranianCEE.2012.6292486>
15. Intel. (2008, October). Intel Open Source Computer Vision Library, v1.1.0. <http://sourceforge.net/projects/opencvlibrary/>
16. Chakravorty, T., Bilodeau, G., and Granger, E. (2020). Robust face tracking using multiple appearance models and graph relational learning. *Machine Vision and Applications*, 31, 23. <https://doi.org/10.1007/s00138020-01071>