AR Tag Detection

Team MelonMusk

May 2, 2021

A.D. Vikas
EDM19B004

A. Srinadh Reddy
EDM19B003

Utkarsh Verma
EDM19B038

V. Pradeep Kumar
EDM19B018

***Abstract*—This project aims to implement a real-time AR tag decoding algorithm in MATLAB. It will take in a video feed, such as a webcam and will output the docoded AR tag as a video too.**

***Index Terms*—augmented reality, computer vision, image processing**

I. Introduction

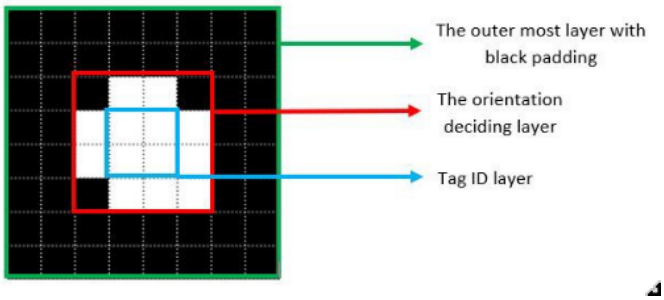


Fig. 1: A sample AR tag.

An AR tag is nothing but an easily recognizable point of reference in an augmented reality(AR) application which helps to calibrate and track camera position in a scene. These markers generally look like QR code sequences and have tag ID, orientation information and, to make them easy to recognize black padding attached to the outer layer.

II. Motivation

The use of AR tags has been increasing rapidly in the recent years thanks to the recent developments in augmented reality and computer vision. Robots are becoming more and more autonomous and aware of their surroundings, and AR tags help in passing information to it efficiently and in a robust manner. They are used as checkpoints to help autonomous robots navigate.

Fig. 2: AR tags being used as markers for augmented reality. (Courtesy: www.artag.net)

Apart from that, augmented reality applications have spanned commercial industries such as education, communications, medicine, and entertainment. In education, content may be accessed by scanning or viewing an image with a mobile device or by using markerless AR techniques.

At first sight, AR tags had us fascinated and hence we decided to take this as a project because it presented a great opportunity for us to dip our fingers into image processing and computer vision in MATLAB. We went with MATLAB because MATLAB doesn't have an official implementation of AR tag detection algorithms at the time of writing, and we wanted to do things from scratch.

III. Existing solutions

AR tags are widely used and hence there are already a lot of pre-existing implementations of this algoirithm. Some of the open-source implementations are:

- sudrag/Perception-and-Computer-Vision-in-MATLAB
- harshkakashaniya/AR-tag-detection
- urastogi885/ar-tag-detection
- anubhavparas/ar-tag-detection

IV. Methodology

The overview of the algorithm is described above. Firstly, the webcam feed is taken as an input and for each frame of the feed, the outlined process is followed.

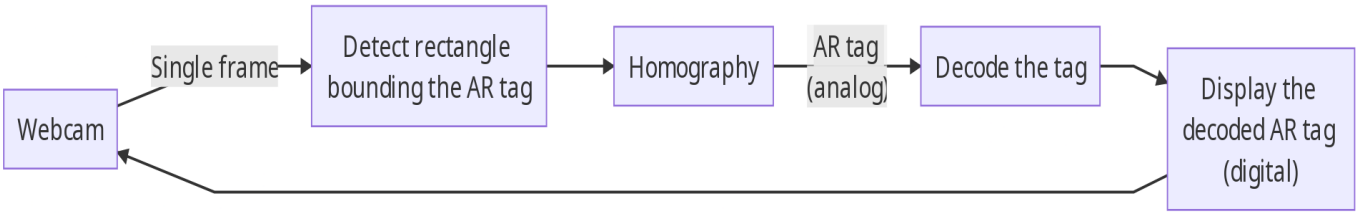


Fig. 3: Overview of the process.

Fig. 4: Dectection of rectangle bounding the AR tag.

Fig. 5: Input image

1. The image is converted from RGB to grayscale, and then the Gaussian and Binarise filters are applied to get a black and white image.

Fig. 6: Black and white image

2. Then Sobel edge-detection is performed to get information about the shapes in the image.

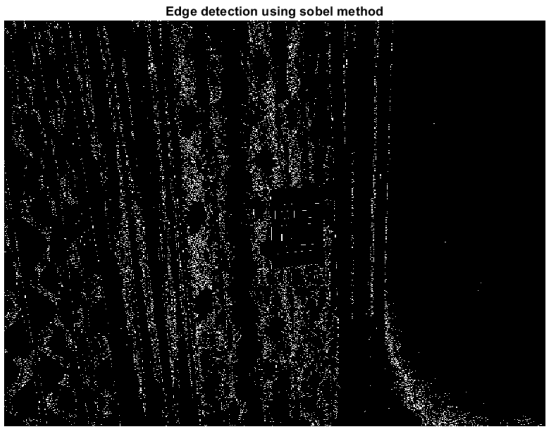


Fig. 7: Image w/edges

3. After this, the edges are made thicker using erosion and dilation filters, and then bounded hollow polygons are filled using `imfill()`. At this point, we only fill those shapes which are above a certain threshold. As the end result, we obtain an image with solid polygons.

Fig. 8: Image w/solid polygons

4. Now we finally apply the Douglas-Peucker algorithm to simplify the solid-polygons and select out shapes which have 4 edges, because that is what we are interested in.
5. Now, we have the AR tag, but it can be skewed, so we apply homography on it to get the proper analog form of the AR tag.

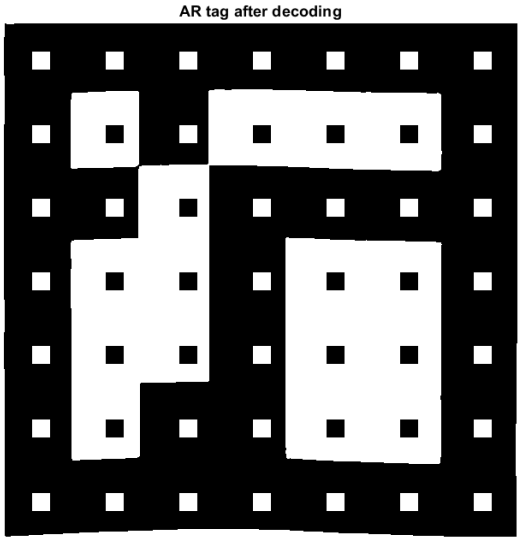
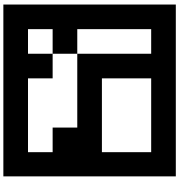


Fig. 9: AR tag

6. Now that we have the AR tag, in a non-skewed form, we go ahead and decode it and output the final digital AR tag and the final output.

(a) Output image



(b) Decoded AR tag

The above described procedure for one frame is repeatedly performed for each frame of the webcam feed and each output is passed to an output video feed, which makes our algorithm perform detection in real-time.

Our algorithm is also able to handle multiple AR tags as shown below.

(a) Input image

(b) Output image

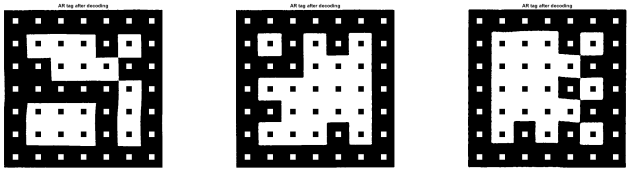


Fig. 12: Detected AR tags

V. Future Plans

We have the following future plans for this project:

- Making the code efficient enough to run on low-power devices like Raspberry Pi.
- Improve the algorithm to perform well in low-lighting as well.

VI. References

We have referred to the following sources for this project:

- https://github.com/sudrag/Perception-and-Computer-Vision-in-MATLAB
- https://github.com/harshkakashaniya/AR-tag-detection
- https://web.archive.org/web/20120814082107/http://www.artag.net/