

DESIGN OF AN AUGMENTED REALITY PROCESSING ALGORITHM FOR BALL DETECTION

Kwame Martin

B.S. Computer Science '11

Norfolk State University

Mentor: **Chad Jenkins, Ph.D**

Robotics, Learning and Autonomy Lab

Department of Computer Science

Brown University, RI

Providence, RI 2010

Table of Contents:

1. Introduction
2. ROS
3. AR ToolKit & AR_RECOG
4. Augmented Reality Processing
 - 4.1 Using Augmented Reality Recognition to see AR tags attached to ball
 - 4.2 Getting iCreate Robot to seek out AR tags i.e. Ball Detection
5. Conclusion & Future Work
6. References

1. INTRODUCTION

The aim of this research project was to design an algorithm for ball recognition which will be used in a bigger project – i.e. an outdoor soccer game using the iRobot Create, an Aus Eee PC and a camera. My paper focuses on both the locomotive and computer vision methods that were taken into account while achieving this aim. The main approach deals with Augmented Reality Tag Recognition. The paper contains my attempt and the extent to which the ultimate goal has been accomplished.

The programming language of choice in this project was C++ for the following reasons – it is one of the languages supported by ROS (including Python); and it is currently the programming language with which I feel most comfortable with.

2. ROS

ROS (Robot Operating System) is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. [1]

This is the main platform upon which all our experiments are run. It is installed on the Asus PC which is then connected to the iRobot Create and camera. It contains libraries for handling streaming camera images and also for communicating with the Create's motors and sensors.

In ROS, programs are called *nodes*. For one node to make certain data available to other nodes, it *publishes* such data to a *topic*. A node that needs such data then *subscribes* to the desired topic. When many nodes need to run at the same time, one can utilize *roslaunch*. By creating a *launch* file that contains all the nodes, all the process can start at the same time.

3. AR ToolKit & AR_RECOG

AR_RECOG processes ROS image-transport with ARToolkit.

ARToolkit is a C and C++ language software library that lets programmers easily develop Augmented Reality applications. Augmented Reality (AR) is the overlay of virtual computer graphics images on the real world, and has many potential applications in industrial and academic research. [2]

One of the most difficult parts of developing an Augmented Reality application is precisely calculating the user's viewpoint in real time so that the virtual images are exactly aligned with real world objects. ARToolkit uses computer vision techniques to calculate the real camera position and orientation relative to marked cards, allowing the programmer to overlay virtual objects onto these cards. The fast, precise tracking provided by ARToolkit played a major role in our project. [2]

4. AUGMENTED REALITY PROCESSING

4.1 Using Augmented Reality Recognition to see AR tags attached to ball

ARToolkit applications allow virtual imagery to be superimposed over live video of the real world. Although this appears magical it is not. The secret is in the black squares used as tracking markers. The ARToolkit tracking works as follows:

1. The camera captures video of the real world and sends it to the computer.
2. Software on the computer searches through each video frame for any square shapes.
3. If a square is found, the software uses some mathematics to calculate the position of the camera relative to the black square.
4. Once the position of the camera is known a computer graphics model is drawn from that same position.
5. This model is drawn on top of the video of the real world and so appears stuck on the square marker.
6. The final output is shown back in the handheld display, so when the user looks through the display they see graphics overlaid on the real world. [2]

Fig. 1 below summarizes these steps. ARToolkit is able to perform this camera tracking in real time, ensuring that the virtual objects always appear overlaid on the tracking markers. This AR Tag was mounted onto the ball.

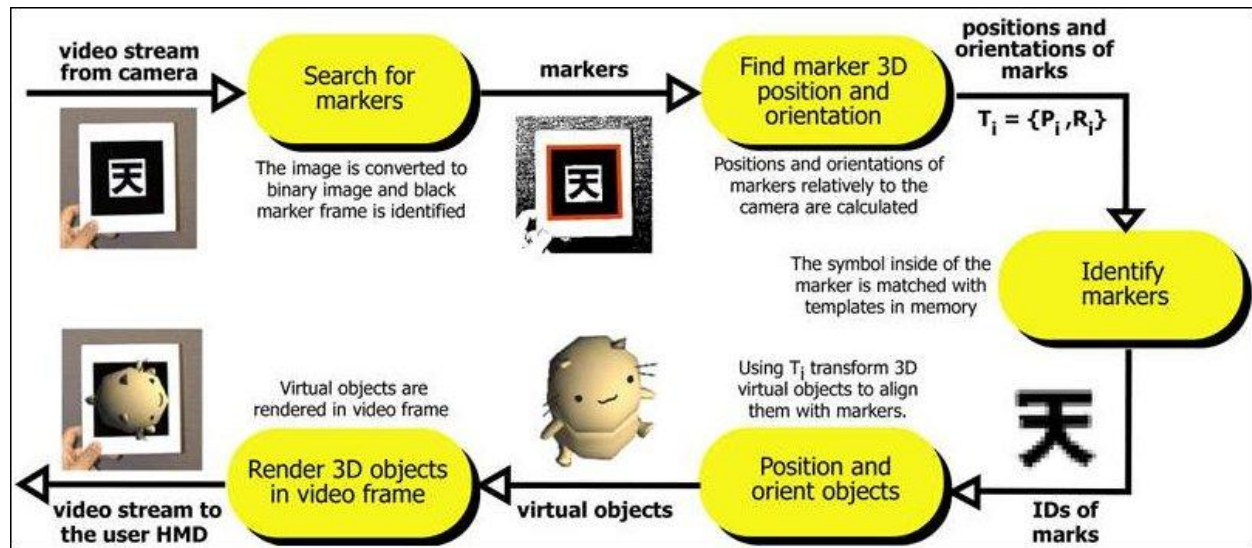


Fig. 1: How ARToolKit works.

The Asus Eee PC's webcam used to recognize the AR tags in the experiment, had to be calibrated specifically due to resolution correction issues. The code below helped resolve those issues perfectly:

```
export GSCAM_CONFIG="v4l2src device=/dev/video0 ! video/x-raw-rgb,width=320,height=240 ! ffmpegcolspace ! identity name=ros ! fakesink"
```

The camera was not initially calibrated to recognize and determine the distance of the AR tag, so I had to follow the “Calibrating Distance” instructions found on the ROS wiki. The instructions were as follows:

- Set up a clear, recognized AR tag at a known distance from the camera.
- Start the ar_recog node and monitor the /tags message
- Arrange the AR tag and camera so that the tag coordinates are as close as possible to the center of the image, and the xRot, yRot and zRot are as close as possible to zero. This will minimize distortion.
- Call the ar_recog/CalibrateDistance service, where dist is the distance between the AR tag and the camera in millimeters:

“rosservice call ar/calibrate_distance dist”

- The more accurately the distance is known, the better ar_recog is able to undistort the image. Thus you may have to repeat the service call a few times until the reported distance converges on the value you are providing.
- Save this aov value to provide as a ROS parameter next time you use ar_recog with this camera. [1]

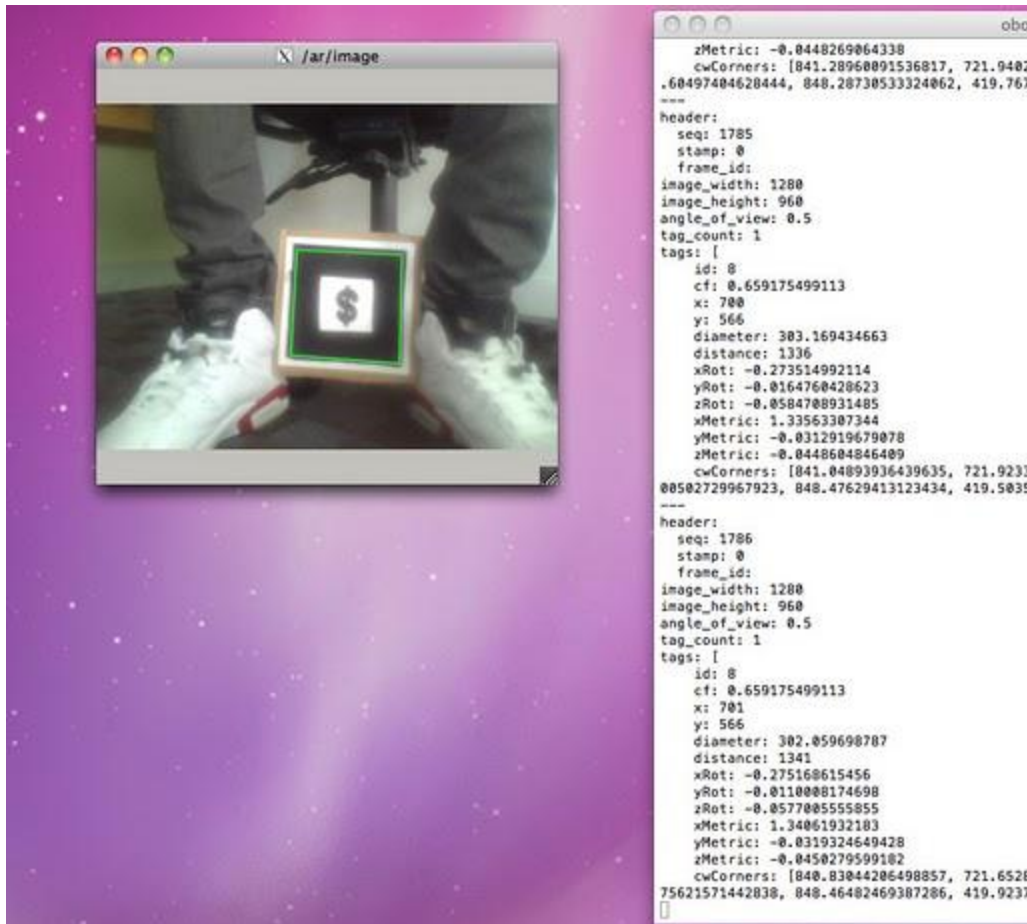


Fig. 2: Image to the left shows the view of the Asus Eee PC's webcam. Here you can see the green outline showing ar_recog finding the AR tag via corner finding. The image to the right shows the AR tag information once it has been recognized giving specific tag information and details.

4.2 Getting iCreate Robot to seek out AR tags i.e. Ball Detection

The main goal initially was to have the AR tag attached to the soccer ball and have the iRobot use the AR tag to seek out & tackle the/kick the ball but I found difficulty in devising a means of doing this without the AR tag toppling over and the webcam losing sight of the tag. So instead, I simply attached the AR tag to a box to display the iRobot functionality with the AR tag recognition as though the box was a ball, the box substitute, being a little better in this case, it being less likely to topple over.

With that said my next goal was to have the iRobot Create with the Asus Eee PC and its webcam mounted on top of it to seek out the AR tagged box and inevitably hit or "kick" the ball, i.e. the box with the AR tag attached to it.

Already in the works for AR Tag Detection and Following at the Brown ROS lab was an algorithm called Nolan.py, located in the Brown ROS Package's Experimental folder. The algorithm that I used to seek out and follow the ball was basically the Nolan.py algorithm with some of my own modifications made to it. The Nolan.py algorithm utilizes ar_recog and its tag recognition services and message publishing to locate an identified AR tag within its range of sight and angle of view, while moving towards it and simultaneously keeping track of the AR tags corners through the use of its corner finding algorithm. In this manner the iRobot Create with the Asus Eee PC and its webcam mounted on top of it appears to follow the AR tag around.

My modifications basically incorporated the manipulation of the ar/image and ar/tag's Y-Metric value to set up an algorithm that now has the iRobot Create:

- i. Rotate, till the Asus Eee PC's webcam identifies an AR tag
- ii. Stop and face the found AR tag
- iii. Move toward the found AR tag, and then
- iv. Increase its forward motion velocity within the closest range of the AR tag, mimicking a soccer "kicking" motion and hitting the box with the AR tag attached to it.

5. CONCLUSION & FUTURE WORK

In conclusion, the experiment shows that through the use of Augmented Reality Processing Algorithm, one can effectively detect the presence of an object (ball, box etc.) and maneuver around and interact with it, given that it has been properly calibrated and tagged with an AR tag. In future, a camera with a higher resolution better designed for sight and object recognition could be used in the location of the AR tag.

Also, through the process of this research, it has been observed that ball recognition will require more than Augmented Reality Processing. Future research work will include more detailed techniques such as Image Processing, HOG (Histogram of Oriented Gradients) and Sift (Scale-Invariant Feature Transform).

I would like to thank the Computer Research Association for this research opportunity and Brown University's RLAB (Robotics, Learning & Autonomy at Brown) for giving me the opportunity to perform research at their lab. Through this granted experience, I have gained much new knowledge in a very interesting field of Computer Science. The skills I have obtained in ROS will be relevant to my future career in the business world of Computer Science.

6. REFERENCES

[1] "ROS Wiki" <http://www.ros.org/wiki>

[2] Sinclair, Patrick. "ARToolKit Documentation." Web. 17 Oct. 2010.
<<http://www.hitl.washington.edu/artoolkit/documentation/>>.

[3] "Augmented Reality." *Wikipedia, the Free Encyclopedia*. Web. 17 Oct. 2010.
<http://en.wikipedia.org/wiki/Augmented_reality>.

[4] "iRobot Create." *Wikipedia, the Free Encyclopedia*. Web. 17 Oct. 2010.
<http://en.wikipedia.org/wiki/iRobot_Create>.