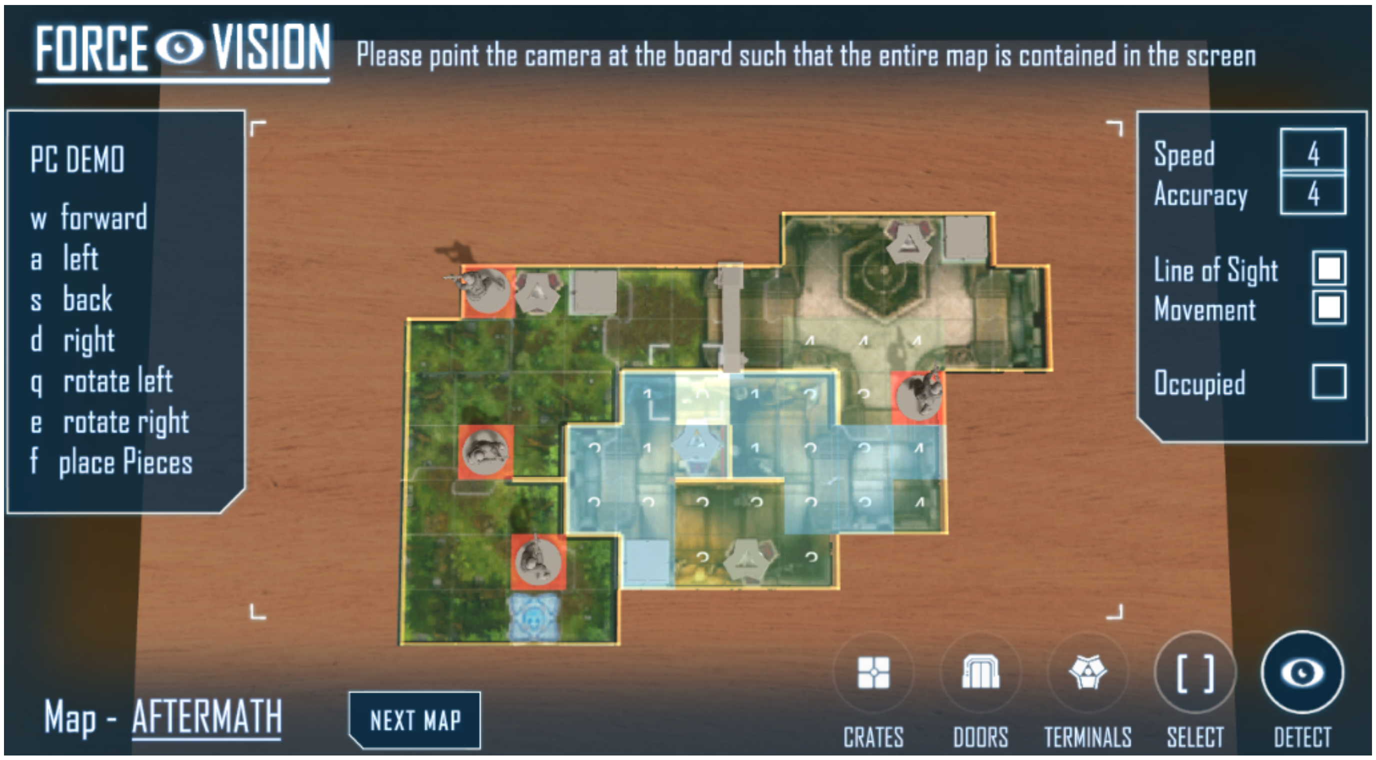
# Research & Development Report

## Enhancing Tabletop Miniature Strategy Game Experience with AR and Object Recognition (Computer Vision)

(Force Vision: Imperial Assault AR Companion App)

Manny Kwong 2022



.

# 1 Introduction

## 1.1 Purpose

The main purpose of this project was to create a minimal viable product that would enhance the gameplay of the tabletop miniature strategy game Star Wars: Imperial Assault using computer vision and augmented reality (AR). Imperial Assault is turn based game that can be played with 2-5 players, where one player acts as the Empire and the rest control the Rebels. Imperial and Rebel units are represented as miniature figurines where for each turn, a unit may perform 2 actions.

The most common actions are to move the unit and to attack another unit. Each unit has a maximum number of spaces it can move represented by a speed number which is affected by the board map features such as walls, objects and doors. Similarly, units can only attack other units for whom they have a line of sight affected by the board map features and with a range determined by an accuracy number. The main function of the app is to improve the player experience by automating the calculation of movement and line of sight and displaying it over a video of the board map in the form of highlighting available spaces. This required the detection of a board map from a mobile camera feed and overlaying such information over the video. The app can also detect the positions of the units and highlight the space occupied by the units which also affects calculations and clearly shows which units are attackable.

Secondary features of the app are displaying and tracking interactable objects such as doors, crates and computer terminals. Doors can be opened or closed which affects calculations, crates can be indicated as being picked up or not and computer terminals turned on or off.

Figurines also affect the line of sight and movement calculations and I aimed to use computer vision techniques to detect figurines from the camera feed from a mobile device. Collecting and processing training image sets for machine learning is time-consuming and laborious. The secondary purpose of this project is to experiment with supplementing photographic training images with synthetic images generated from a virtual scene.

## 1.2 Constraints

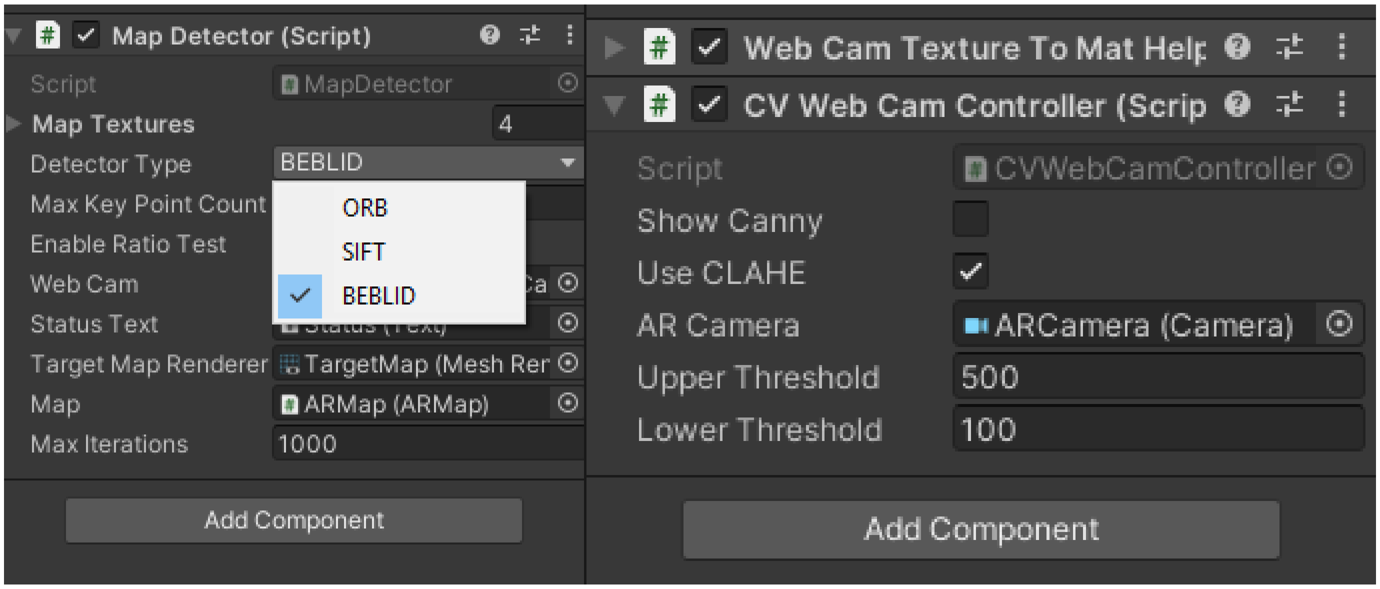
The project duration was from 21/3/2022 to 30/6/2022 (~100 days). The main software used was Unity Game Engine (for modelling map behaviour and the user interface), Vuforia Engine (for augmented reality) and Open CV (for object detection). Other software used were Visual Studio (for coding), Trello (for project management), and Github (for backup and version control). All software used was free as the budget of this project was $0. The target device is mobile phones running Android 10 or greater.

# 2 Development

## 2.1 Board Detection

Initially, feature description methods such as SIFT (Scale-invariant feature Transform), ORB (Oriented Fast and rotated BRIEF) and BEBLID (Boosted Efficient Binary Local Image Descriptor) were implemented to detect the game board map and recognize which type of map it was (from a selection of 3). The detected features were then used to calculate the orientation of the map using homography. SIFT was the most accurate and stable in terms of map orientation but ran at a very low framerate on mobile devices. ORB ran at a better speed up to about a maximum of 2000 key features but at the expense of detection success and orientation calculation, with the overlay being very jittery.

After struggling to make the above run well on mobile devices, the AR software development kit Vuforia was chosen instead. Vuforia performed very well on mobile devices and was compatible with the engine used for the project.



Map Detector (Script)

| | |
|---|---|
| Script | MapDetector |
| Map Textures | 4 |
| Detector Type | BEBLID |
| Max Key Point Count | |
| Enable Ratio Test | |
| Web Cam | Ca |
| Status Text | Status (Text) |
| Target Map Renderer | TargetMap (Mesh Rer |
| Map | ARMap (ARMap) |
| Max Iterations | 1000 |

ORB
SIFT
✓ BEBLID

Web Cam Texture To Mat Help

CV Web Cam Controller (Scrip

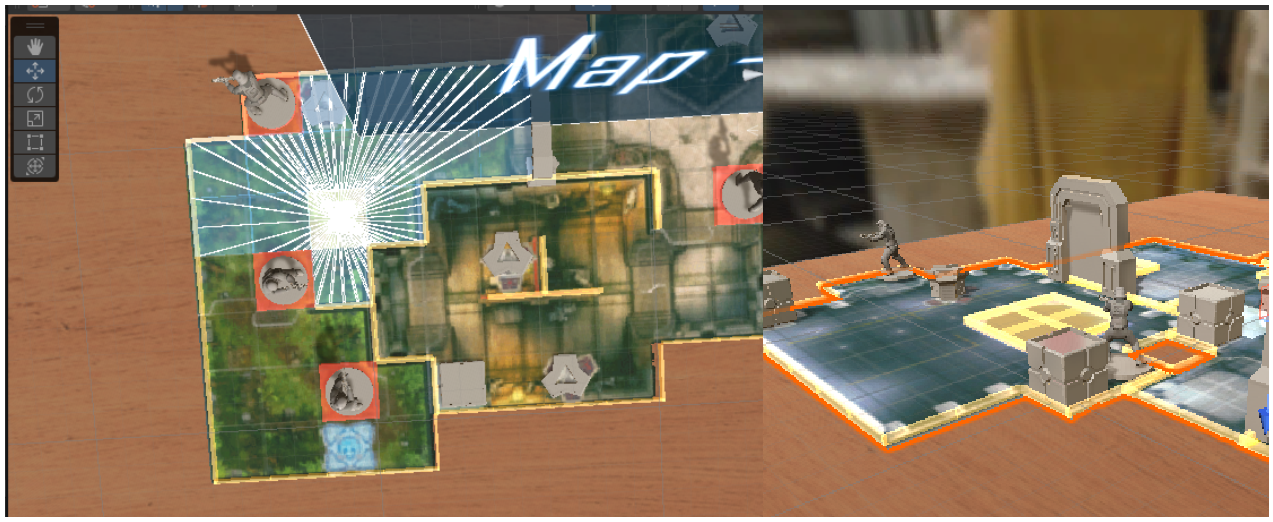
| | |
|---|---|
| Script | CVWebCamController |
| Show Canny | ☐ |
| Use CLAHE | ✓ |
| AR Camera | ARCamera (Camera) |
| Upper Threshold | 500 |
| Lower Threshold | 100 |

## 2.2 Map Features and Calculations

The features of the game board map were modelled using the Unity Game Engine. Each square space of the game board map was implemented as a prefab and spawned on a grid.

Box colliders were used for collision detection for walls, occupied spaces and blocking objects that can be activated on the space. This was then used for line of sight calculations by emitting rays from a selected space of a hypothetical attacking unit to each space along the perimeter of the grid. All spaces intersecting these rays within the range of the attacking unit can then be highlighted to show where a character may attack.

The state of each space used for movement calculation consisted of whether it was traversable, and whether it was possible to traverse to any of its neighbouring spaces. and if a unit was occupying the space. For the movement calculation, a boundary flood fill algorithm with 8 directions from the attacker's



position was used.

## 2.3 Piece Detection

Again SIFT, ORB and  BEBLID (Boosted Efficient Binary Local Image Descriptor) algorithms were tried for detecting the unit figurines on the board map. In addition, the Hough Circle detection algorithm was also tested as the base of each figurine was circular. These worked well enough when the figurines were placed on a playing background, however, the detailed illustrations on the board map made detection using these methods very inaccurate, plagued with false positives. An effort to reduce the false positives by reducing the camera image to outlines using Canny edge detection also failed.

Object detection algorithms such as Cascade Classifiers, SVM, and CNN were considered.
 and photos. Target base of game pieces for calculation of board position and all the different pieces have the same base. The object detection method chosen was a HAAR ()  Cascade Classifier using OpenCV for its relatively fast speed of training and performance on a mobile device. Though newer, more accurate methods are available such as SVM (Support Vector Machine) and CNN (Convolutional Neural Network), further research indicated that they were not performant on mobile devices. These methods take significantly more time and effort to implement and it was decided that it was not feasible to do so under the time, cost and equipment constraints of this project.

## 2.4 UI

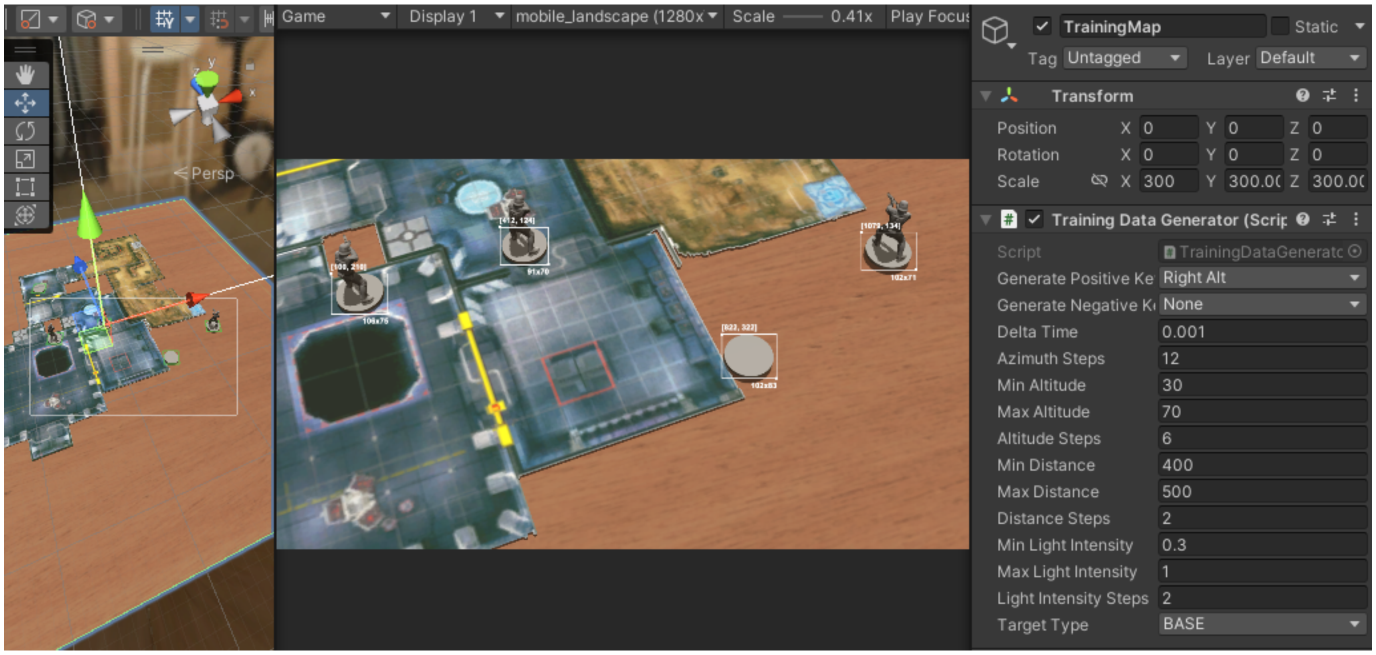
A Star Wars-themed presentation was used. Tapping on the logo will bring up the about screen. In the middle of the screen is the target pointer that highlights a space. Along the bottom right are buttons that toggle the visibility of doors, crates and terminals, unit figurine detection, and the selected space button, Pressing the selected space button will toggle the visibility of a selected space panel. The selected space panel contains toggles for the selected space's line of sight, movement and whether it is occupied by a blocking object or unit.

## 2.5 Virtual Training Image Set Creation

A model of the game board map was created in Unity.3D models of figurines were downloaded and placed randomly on valid spaces of the game board map. Each figurine projected a bounding box around its base onto the virtual camera for annotation. Initially, the bounding boxes were around the whole figurine, but that made it difficult to calculate the space it was occupying. There were many different types of figurines but the common element was the circular base. It was much simpler to detect the base rather than train multiple classifiers for each type of figurine.

Images were captured by the virtual camera at different camera angles, lighting conditions and figurine positions. Each image captured would write onto a text file with the image file location, top left corner coordinates and size. Similarly, a negative set was created with the pieces absent.

The cascade classifier was then trained with a random selection of 500 mixed generated and photo positive images and 500 negative images. From the



TrainingMap Static
Tag Untagged Layer Default

Transform

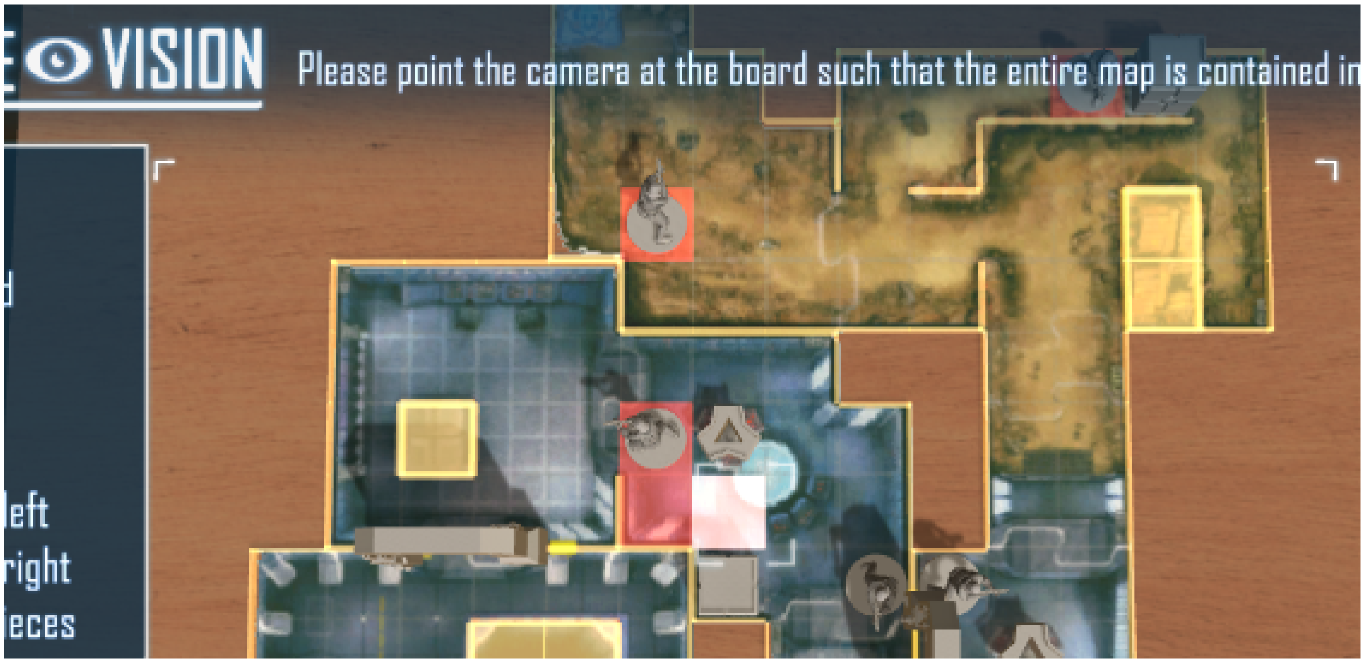
| | | | | | | |
|---|---|---|---|---|---|---|
| Position | X | 0 | Y | 0 | Z | 0 |
| Rotation | X | 0 | Y | 0 | Z | 0 |
| Scale | X | 300 | Y | 300.0( | Z | 300.0( |

Training Data Generator (Scrip

| | |
|---|---|
| Script | TrainingDataGenerato |
| Generate Positive Ke | Right Alt |
| Generate Negative K | None |
| Delta Time | 0.001 |
| Azimuth Steps | 12 |
| Min Altitude | 30 |
| Max Altitude | 70 |
| Altitude Steps | 6 |
| Min Distance | 400 |
| Max Distance | 500 |
| Distance Steps | 2 |
| Min Light Intensity | 0.3 |
| Max Light Intensity | 1 |
| Light Intensity Steps | 2 |
| Target Type | BASE |

## 3 Results

While functional in Unity Editor (emulation) on PC, it does not work well on mobile devices. Map detection works on mobile with the camera directly facing the board but struggles if viewed from more to the side. Figurine detection is poor and false positives are quite frequent due to the detailed illustrations of the game map.



Please point the camera at the board such that the entire map is contained in

## 4 Discussion & Conclusion

After experimenting with using feature descriptors and homography to detect, identify and orient the game board map in AR I relented and used Vufloria Engine instead. Line of sight and movement calculation works correctly on PC in Unity Editor but is buggy on mobile phones. For figurine detection, I tried feature descriptors and cascade classifiers with poor results. Improving the quality and quantity of training data should improve results. SVM and CNN would probably give better results though performance may be an issue. Overall this was a nice introduction to computer vision and machine learning but in the end, making a performant viable product was above my abilities.

## 5 Link to Project

Please open the unity project files and play the Demo scene if possible.
mansinh/ForceVision (github.com)