EE590 – Directed Research – April 23, 2019

A Report on

Using fiducial markers overlaid on retro-reflective materials

For enhancing detection and user experience

All the codes were implemented in C Sharp and Unity

Submitted by:

ANURAG SYAL

MASTERS IN ELECTRICAL ENGINEERING

MULTIMEDIA AND CREATIVE TECHNOLOGIES

USC ID: 9954-6583-64

Abstract and Motivation

The problem statement comprises of designing a method for recognizing fiducial markers over a long range, with the predicate that the method should work in relatively darker or low-lit environments.

Retro-reflective materials, also called reflective material, are materials that reflect light back to its original path. Such materials are mainly used for roads, traffic safety clothing, automobiles, ships, fairways, stages, various passageways and related mechanical equipment.

Figure 1: Example of retro-reflective materials being used in traffic-policeman's clothing
Image Source: http://www.longoureflective.com/product/retro-reflective-material.htm

Approach and Procedures

For the method to work, it was quintessential to perform the following steps:

1. *Finding a good prototyping platform*: For this purpose an Android phone was chosen with Unity as the proto-typing software. Reason for this being that a lot of documentation exists for both Unity and Android as well availability of Android phone.
2. *Experimenting with various Computer Vision libraries*: For this step, the following libraries were picked because of availability of documentation and their nature of being open source –
   a) OpenCV – Extremely customizable but does not come with built-in fiducial marker capability
   b) ArUco – The oldest library for this purpose but does not come with a documented Unity wrapper
   c) Void AR – Another library with its roots in ArUco library, has a Unity wrapper as well but scarce documentation
   d) Vuforia – Most robust library to be compatible with all major Android and iOS devices along with many wearable devices like HoloLens, Vuzix M300, Epson Moverio BT-200 and ODG R-7, has a well documented Unity Wrapper and comes with lots of tutorials

   NOTE: For the purposes of developing the proof-of-concept, Vuforia's free version was used
3. *Testing performance with paper-based image targets:* This step is equivalent to finding the perfect conditions in which there is a maximum chance of getting detection. An image target is any image which has enough features so that it is 'detectable' and 'tractable' in a real-time environment

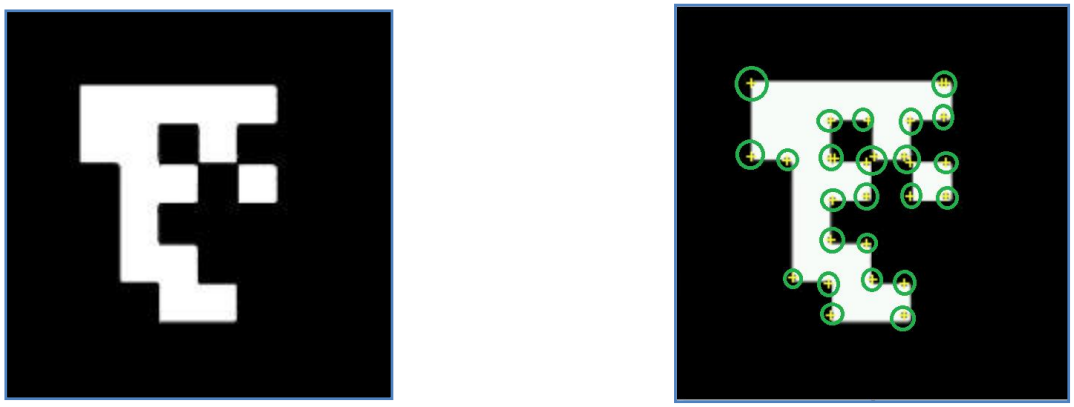

Figure 2: (Left) A fiducial marker uploaded to Vuforia engine for feature detection. (Right) A reference created by the Vuforia engine identifying the features present in the fiducial marker

4. *Testing performance with paper-based fiducial markers / VuMarks:* Vuforia has a whole different sub-library for fiducial markers called as the VuMarks library. Unlike image targets, VuMarks are uniquely assigned and can be identified with an ID. This step is equivalent to finding the perfect conditions in which there is a maximum chance of getting detected for a VuMark.

Attributes	Differences between Image Targets and VuMarks in Vuforia	
	Image Targets	VuMarks
Does not need a lot of features for recognition	✘	✓
Uniquely identifiable with an ID	✘	✓ Supported ID Data types: string, integer, alpha-numeric.
Has to asymmetric for better chances of getting a detection	Yes. The image needs to have features evenly distributed across the area for better detection. Being or having symmetric features results in bad detection	Yes, but can design elements that are symmetric and appealing to the eye **[NOTE: VuMark needs to designed separately in a SVG format]**
Web-based API for generating more images similar to fiducial markers	✘	✓
Total number of images in a database	**Storing images in a database on-the-device:** 100 in free version, 1000 in paid. **Storing images in a database based on cloud:** 1 million	

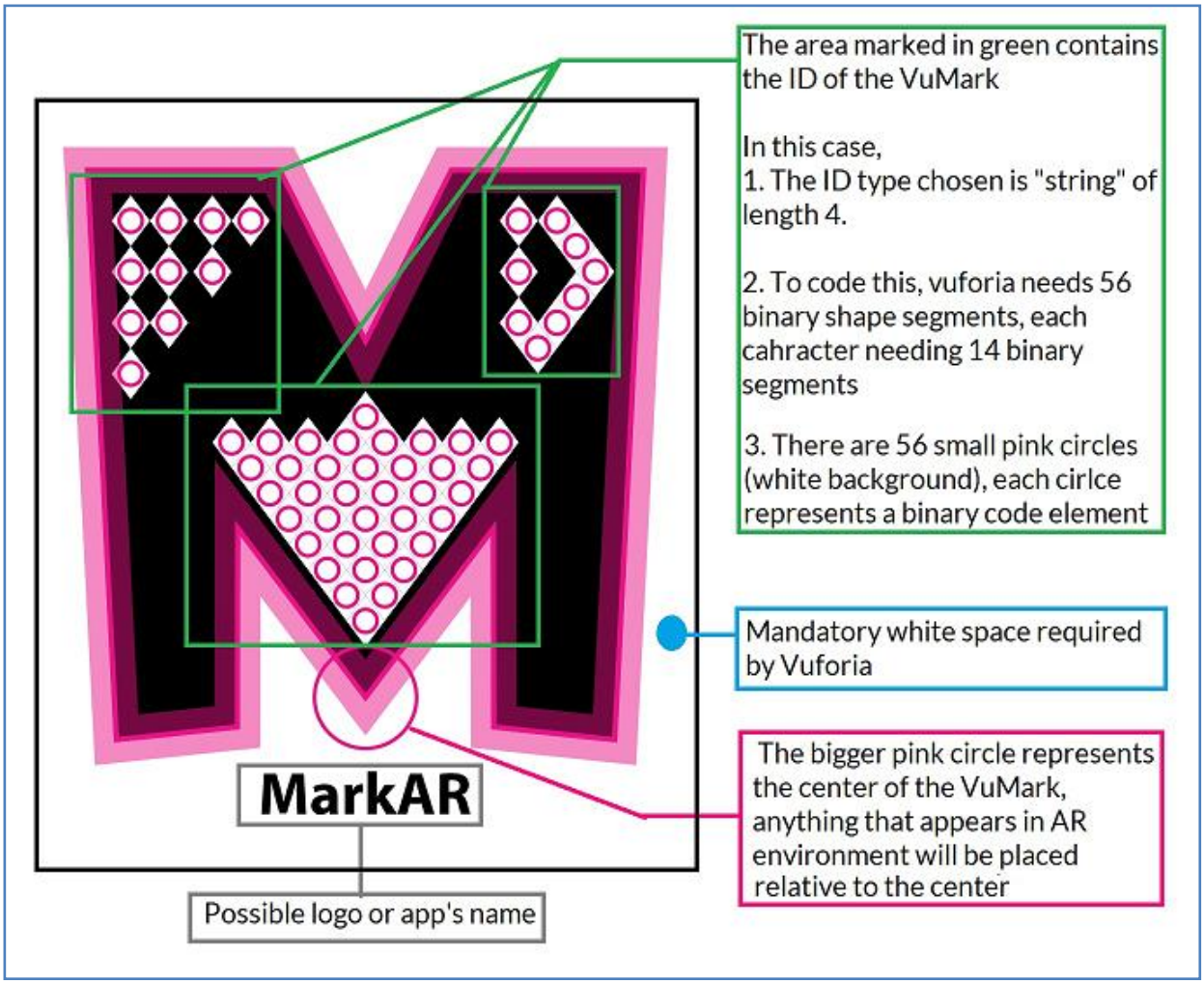

Figure 3: VuMark designed for the purpose of this project, design using Adobe Illustrator, as per the design guide

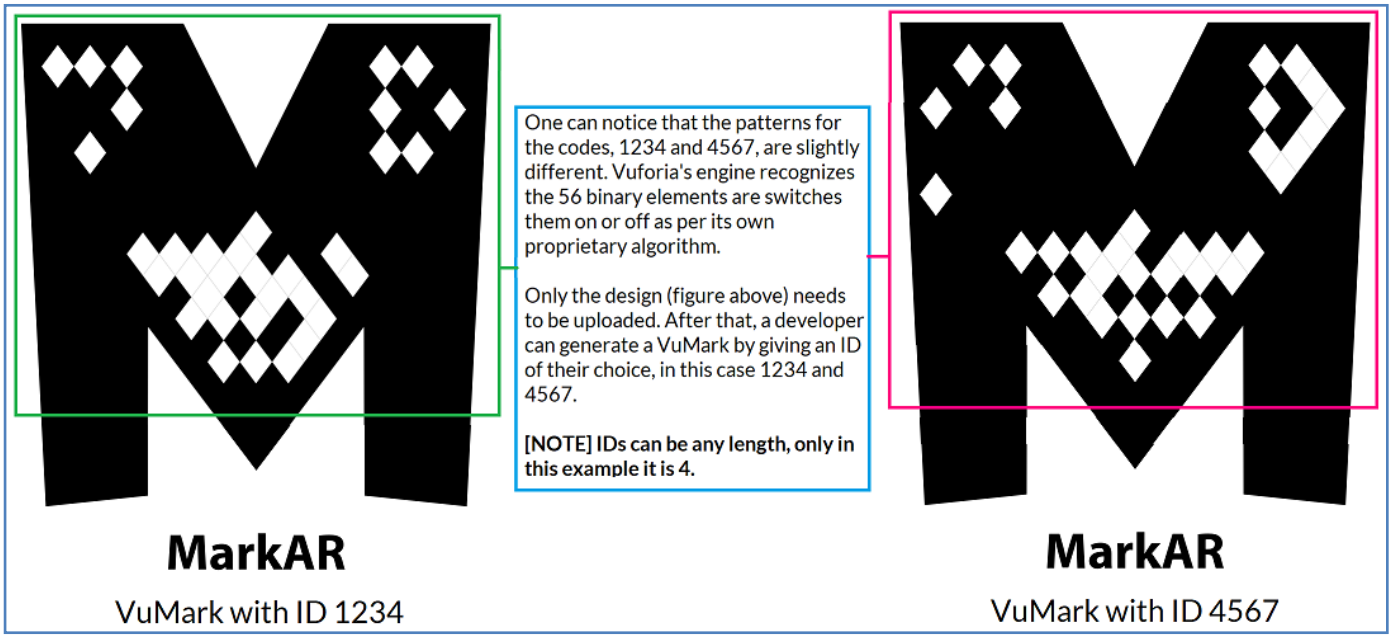

Figure 4: VuMarks generated by Vuforia engine

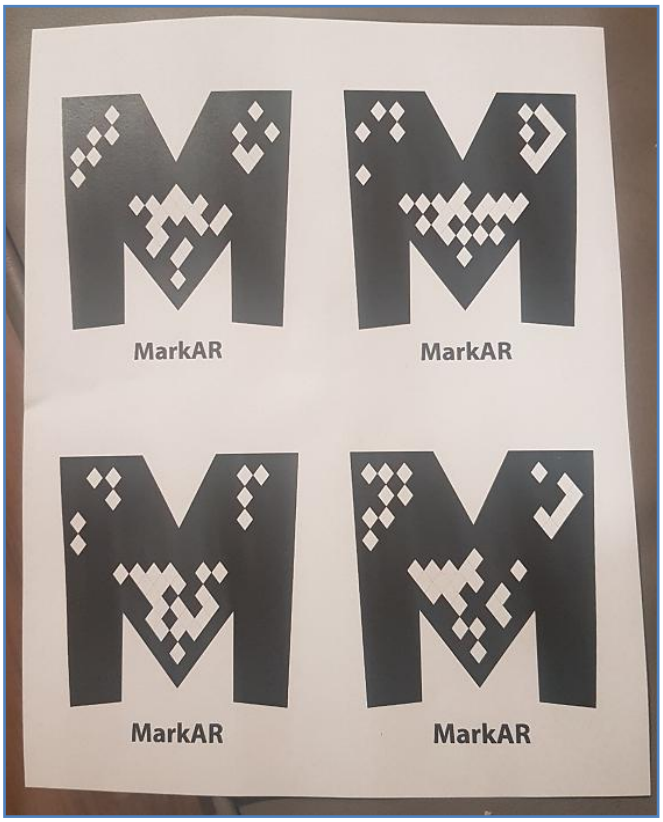

Figure 5: VuMarks printed on paper, each of size 3 inches

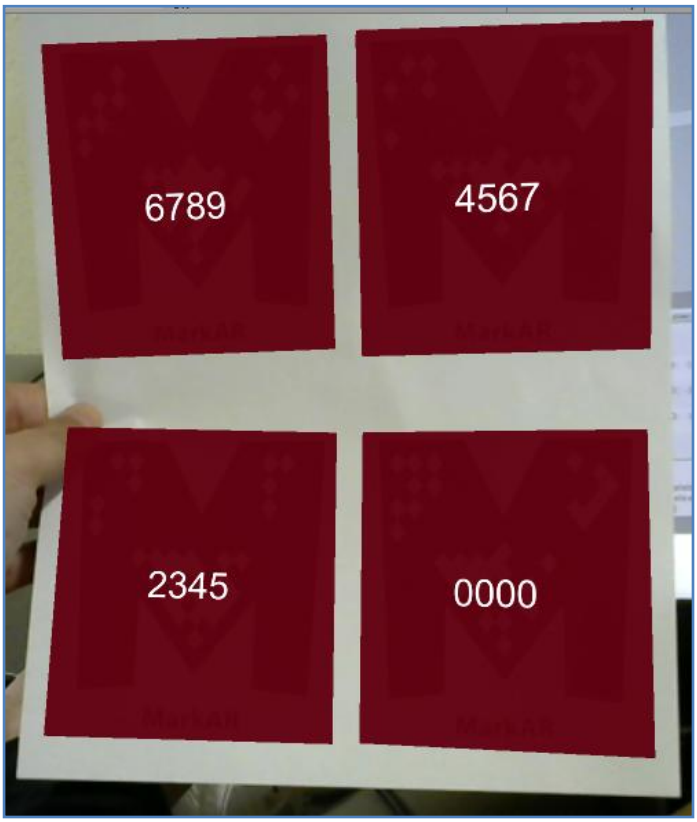

Figure 6: VuMarks identified in Unity using paper shown in figure 5

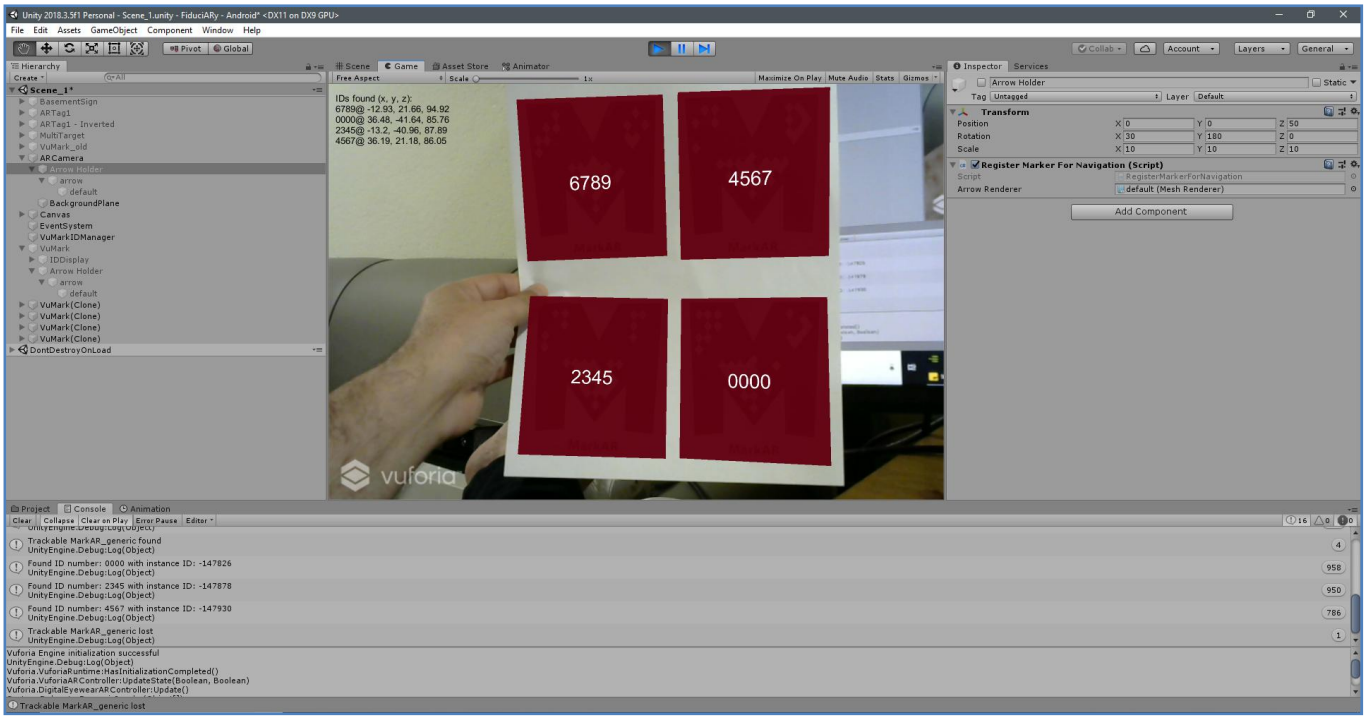

Figure 7: Testing environment in Unity 3D

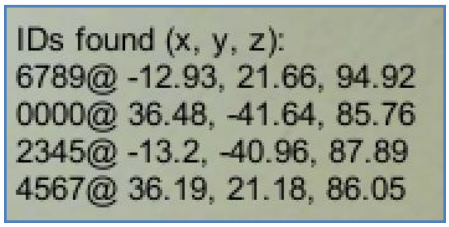

Figure 8: Location in world-coordinates identified in Unity (camera is origin)

5. *Testing performance with retro-reflective material:* The primary aim of this was to identify the correct size for detecting VuMarks (fiducial markers) printed on transparency-paper attached to retro-reflective material based cloth. For this step, 3 sizes were used as shown in the figure below.

Figure 9: VuMarks on reflective cloth in bright environment

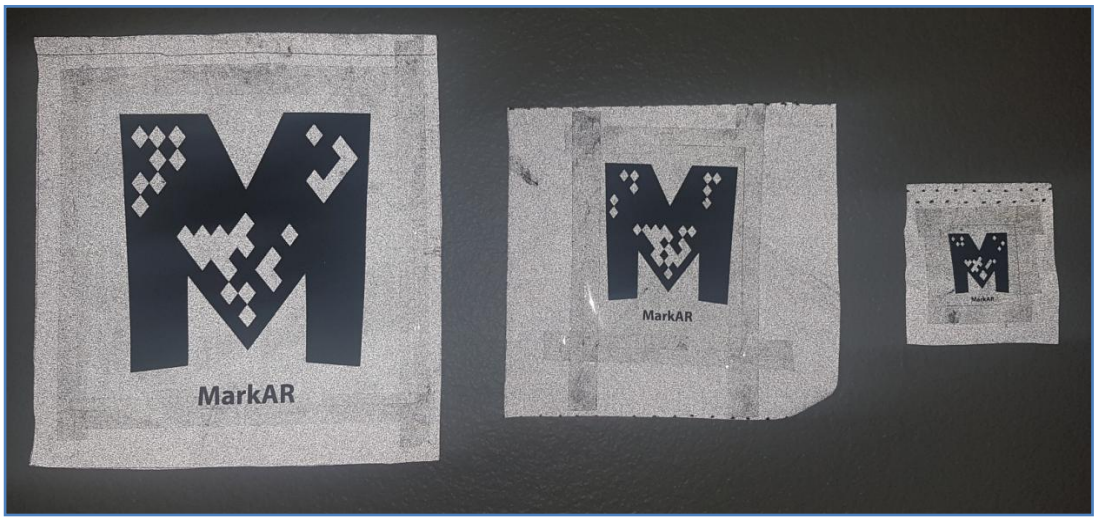

Figure 10: VuMarks on reflective cloth in low-lit environments using camera flash

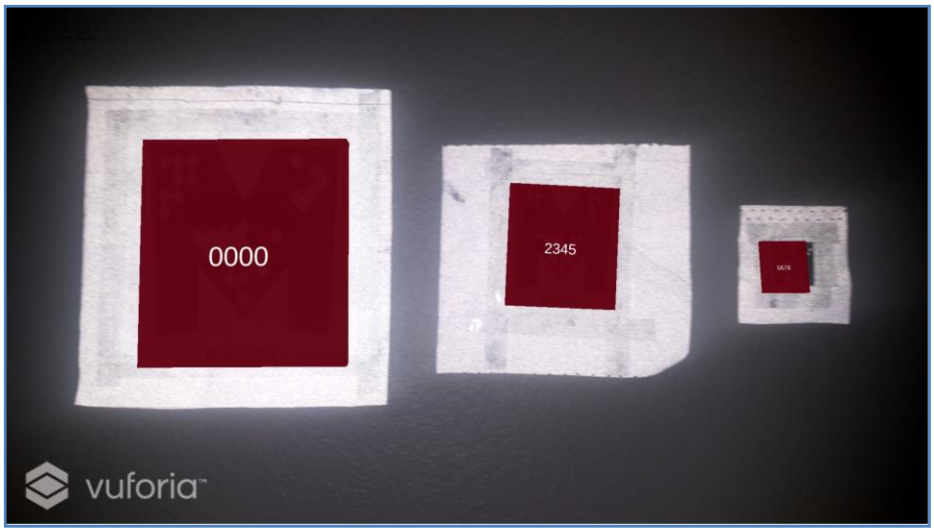

Figure 11: A screenshot from app running on Samsung Galaxy S7 (Android), VuMarks have Ids '0000', '2345' and '5678'

6. *Designing a guidance system (Partial implementation):* For this step, Vuforia's extended tracking system was tested. But the results with the subject android phone (Samsung Galaxy S7) were not up to the mark. The system had glaring errors upon visual inspection. A better approach would be use a SLAM library like OpenSLAM along with Vuforia's extended tracking system to get better results.
   Nonetheless, a semblance of guidance system was implemented using "Arrow Markers" which activate once any VuMark is recognized. The system currently works for only one target in view.

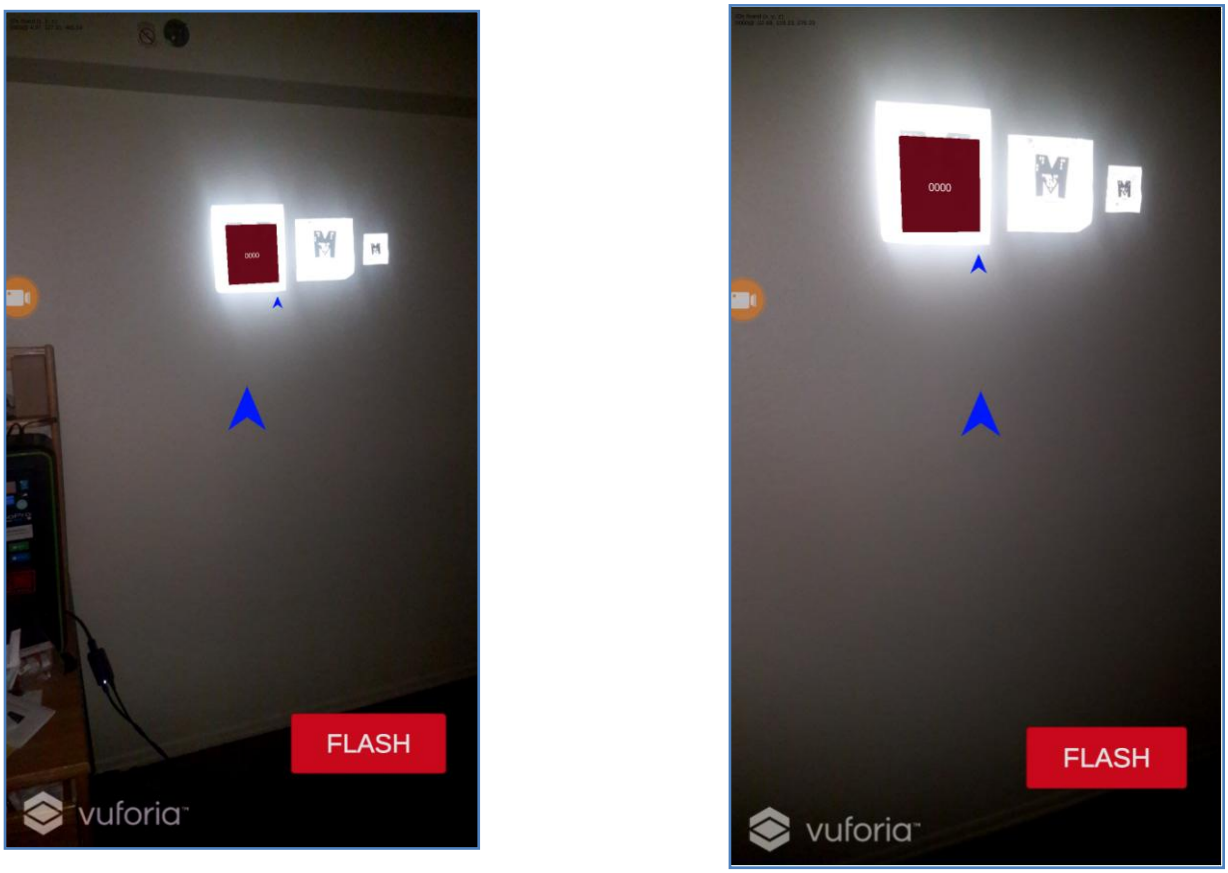

Figure 12: An elementary navigation system comprising of 2 arrows, one at the camera centre and one near the identified VuMark provides a rough sense of direction. (Left) At 4 meters from VuMark. (Right) At 2 meters from VuMark.

Results

The following data was collected with different sizes of fiducial markers (VuMarks). The experiments were conducted for both paper-based and reflective material based backgrounds.

Size of Marker	Minimum detection distance		Maximum detection distance	
	Paper	Retro-reflective	Paper (only in well-lit conditions)	Retro-reflective
0.75 inch	< 0.5 ft (0.2 meter)	< 0.5 ft (0.2 meter)	2 ft (0.7 meter)	2 ft (0.7 meter)
1.5 inch	< 0.5 ft (0.2 meter)	< 0.5 ft (0.2 meter)	2 ft (< 1 meter)	3 ft (1 meter)
2.0 inch	<1 ft (0.3 meter)	<1 ft (0.3 meter)	4 ft (< 1.4 meters)	6 ft (2 meters)
3.0 inch	<1 ft (0.3 meter)	<1 ft (0.3 meter)	4 ft (< 1.4 meters)	6 ft (2 meters)
4.0 inch	<1 ft (0.3 meter)	<1 ft (0.3 meter)	4 ft (< 1.4 meters)	6 ft (2 meters)
6.0 inch	1 ft (0.3 meter)	1 ft (0.3 meter)	6 ft (< 2 meters)	10 ft (3 meters)
8.0 inch	2 ft (0.7 meter)	2 ft (0.7 meter)	12 ft (< 4 meters)	20 ft (6 meters)

Discussion on Vuforia's performance

Vuforia has a limit of 100 VuMarks per database (unlimited for pro-version) with a maximum of 5 targets being tracked simultaneously in a frame. In most general cases, it would be physically improbable to have more than 5 targets in a single frame). The VuMark library treats the markers differently than general images by "faster-recognition" and assigning each VuMark a different instance-ID to be looked up into a database.

The larger a database gets, the tougher it for the developers to maintain such a large database and more importantly, it signifies that the targets are not well classified. Hence it is an imperative for the developer to create a system that can cater to more than one database. One may feel that how could more than one databases be necessary, well such needs may arise when there are a large of products and very categories. Like incoming shipments could have one database and outgoing shipments could have another.

Workarounds for increasing performance:

1. Time domain multiplexing for device database: As per documentation, Vuforia takes 3 frames (or 100 ms) for recognition and reporting, assuming 30 fps, so my suggestion is that we can scan through 10 databases in one second by deactivating and activating different databases every 3 frames. This will increase the count to 300.

2. Cloud database: Vuforia cloud database can hold up to 1 million images. But the frame rate is of course compromised. They have no guarantee on this, but assuming a 150 ms network RTT delay and 100 ms for recognition and reporting, the FPS drops to 4.

Conclusion

There are a couple of design decisions to make:

1. Desired FPS: Detecting 1 million VuMarks using cloud version @ 4 FPS or 300 VuMarks @ 30 FPS using device version
2. Size of VuMark: It was observed that the size of 6 inch was the best size for creating an environment conducive for detecting VuMarks. Below 6 inches, the range reduces drastically and above 6 inches gives us diminishing returns.