Evaluation of Augmented Reality Frameworks for Android Development

http://dx.doi.org/10.3991/ijim.v8i4.3974

I. Marneanu[1], M. Ebner[1] and T. Rößler[2]
[1] Graz University of Technology, Graz, Austria
[2] Evolaris Next Level GmbH, Graz, Austria

***Abstract*—Augmented Reality (AR) is the evolution of the concept of Virtual Reality (VR). Its goal is to enhance a person's perception of the surrounding world. AR is a fast growing state of the art technology and a variety of implementation tools thereof exist today. Due to the heterogeneity of available technologies, the choice of the appropriate framework for a mobile application is difficult to make. These frameworks implement different tracking techniques and have to provide support to various constraints. This publication aims to point out that the choice of the appropriate framework depends on the context of the app to be developed. As expected, it is accurate that no framework is entirely the best, but rather that each exhibits strong and weak points. Our results demonstrate that given a set of constraints, one framework can outperform others. We anticipate our research to be the starting point for testing of other frameworks, given various constraints. The frameworks evaluated here are open-source or have been purchased under Academic License.**

***Index Terms*—Augmented Reality, framework, Android, evaluation.**

I. INTRODUCTION

Augmented Reality is a very promising emerging technology, growing in popularity on mobile devices. A number of research studies were published in late 2013, forecasting the future of the Augmented Reality market [1]. For instance, Juniper Research has estimated that the number of mobile AR users worldwide will steadily grow to 200 million by 2018 [2]. The AR technology has made great progress on mobile phones, and Juniper Research further predicted in 2012 that over 2.5 billion mobile AR apps will be downloaded annually to smartphones and tablets by 2017 [3].

These premises lead to the research question – "Which is currently the best open framework for developing an Augmented Reality mobile application?" Towards this goal, we evaluated open AR frameworks for Android mobile development, due to the fact that the iOS platform has been evaluated over the years and clear results can be found in Dominik Rockenschaub's Master's Thesis [4]. Another incentive for researching the Android specific open AR frameworks is the recent job market growth registered in Android development [5].

Various AR systems are available today. Based on the hardware and technical capabilities of the testing device, three main tracking techniques are defined: marker-based tracking, markerless tracking, and GPS tracking. Our research evaluates the AR frameworks using visual tracking methods, specifically markerless tracking. Marker-based tracking is also an optical method, but has become obsolete [6]. Special fiducial marks have to be created in order to be tracked. They have to be maintained over time, which is also a costly operation. Markerless based tracking on the other hand, can target any part of the surrounding environment. Any image or object can be used as a target without being tied to a specific marker. Markerless tracking is gaining more and more ground against the marker-based tracking, as it does not require generating of invasive markers. These methods extract information and characteristics about the environment, which may be useful again later.

This publication is organized as follows: the methodology is presented in section II; section III describes the setup of the testing environment and the implementation process; the results per criterion are given in section IV while the discussion analyzing these results can be found in section V; and finally we conclude the paper in section VI.

II. METHODOLOGY

Six AR frameworks have been researched and evaluated after a careful evaluation of 35 frameworks. The 6 chosen frameworks support Android development, as well as markerless tracking. An important criterion in choosing the frameworks to be evaluated is the availability of the library. Several are available for commercial use only (for the purpose to sell and profit) or are simply not available to the general public. Others are accessible only for a trial period. Some are depreciated, like in the case of Popcode, or even present language barriers as Koozyt, where information is provided only in Japanese. Table I provides the six frameworks, the development country and start year as well as availability.

TABLE I.
EVALUATED FRAMEWORKS

Framework	Development	Availability
ARLab	Spain (2006)	Demo
ARToolKit	USA (2001)	Academic License
D'Fusion	France (1999)	Watermark
Vuforia	Austria (2011)	Free
catchoom	Spain (2011)	Demo
metaio	Germany (2003)	Watermark

For evaluation purposes, various criteria have been defined. The frameworks are actively tested and the testing times are recorded. Therefore, frameworks can be

compared against each other given a criterion or a set of criteria.

For a better visualization of the results, a test app has been developed. As a result, different frameworks can be tested in real time, and the results are saved in a local database. The results are then presented in different visualizations.

III. SETUP AND IMPLEMENTATION

For each criterion, we simulated a testing environment and each framework was evaluated under the same conditions. Several tests have been performed per framework for each criterion, thus the average testing time is used to compare the frameworks.

A. Criteria Categories

Two main criteria categories have been identified and actively tested: environmental criteria, and target criteria. Environmental criteria represent those constraints found in the immediate neighbourhood, which are conditioned by the environment and influence the recognition of the target image. Tested here are the influence of different light intensities or that of the frameworks' performance when dealing with dark backgrounds versus bright ones. Other considered criteria include differences in viewpoint, visible target area, the mirroring effect and various distances between the testing device and the target image, as well as the noise present in the target image due to its deterioration over time.

- The evaluation of the **light intensity** criterion has been executed in special lightning conditions. The darkness criterion has been tested inside a room, in natural light at sunrise. The second test is also performed inside a room, by overexposing the target image to the direct light of a desk lamp. The third simulated event happens by closing and opening the window shades. This experiment is meant to replicate sudden changes in light intensity. The mirroring effect is reproduced by tracking the target images on a computer screen.
- The evaluation of the **viewpoint** criterion has been performed in four different perspectives, from a 45° angle to the left, right, upwards and downwards of the target image as depicted in Fig. 1.
- **Visibility** is tested by uncovering 10% of the visible area at every step. A white sheet of paper is used to cover the target image and uncover it in incremental steps as shown in Fig. 2. When the target is recognized and the view augmented, the step increasing stops.
- The evaluation of the **noise** criterion has been performed by digitally altering the target images and adding different noise levels. The test sequence includes five noisy target images as given in Fig. 3. The noise is incrementally added starting from a 10% noisy target image in steps of 20% up to 90%.
- The evaluation of the **distance** criterion has been performed by starting testing up close, at a 10 cm distance from the target image. The distance is increased at every step by 10 cm until the framework cannot recognize the target anymore.
- The evaluation of the **background** criterion has been performed in two background test sequences, dark versus bright contrasts. This reproduction is meant to simulate the placement of the target image against high-contrast backgrounds.

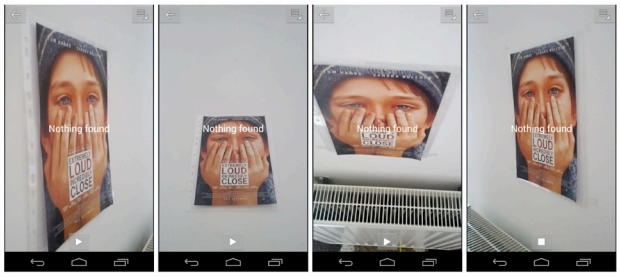

Figure 1. Viewpoint Test Sequence

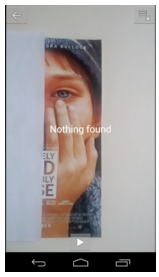

Figure 2. Visibility Test Sequence

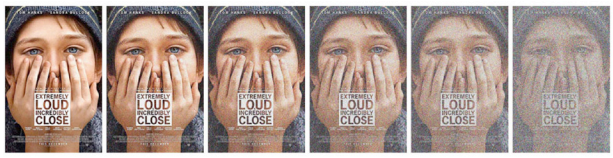

Figure 3. Noise Test Sequence

Target criteria, on the other hand, are those target image attributes that can be configured and directly influence the detection of the target. While an image can be a target image for one framework, it might not be supported by another. For example, some frameworks are robust to aspect ratio changes or various contrast ratios. Moreover, different target images sizes and a number of special printed materials are tested. We also evaluated whether or not a considerable difference in testing time exists between detecting the original target and its grayscale.

- The evaluation of the **grayscale** criterion has been performed on a printed default size target image in grayscale. An example can be seen in Fig. 4.
- The evaluation of the **contrast ratio** is carried out on digitally modified target images as depicted in Fig. 5. Four target images are tested with a contrast value set to -50 and at each step it is incremented by 50.
- The evaluation of the **size** criterion has been executed on four different target images. Each of the six tested target images has been

reduced to 5cm, 10cm, 15cm and 20cm as shown in Fig. 6. The distance between the testing device and the target image has been constant at a default of 30cm.

- The evaluation of the **aspect ratio** requires the target images to be digitally modified. Thus, each of the six original target images is shrunk either vertically or horizontally by reducing the height, respectively the width to one third (1/3), a half (1/2) and two thirds (2/3) as given in Fig. 7.
- The evaluation of the **material** criterion has been performed in three different stages. The first case depicts the target image behind a glass window, as it would be if a poster were placed inside the side panels of a bus shelter. The second case simulates a restaurant menu scenario where the menu is laminated to protect the paper against deterioration and mishaps like spilled drinks. The third target image is printed on a glossy photo printing paper as it might be used in a scenario for displaying promotional ads or as packaging of different promotional products.

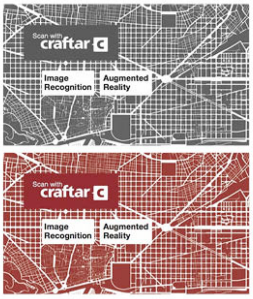

Figure 4. Grayscale Test Sequence

Figure 5. Contrast Test Sequence

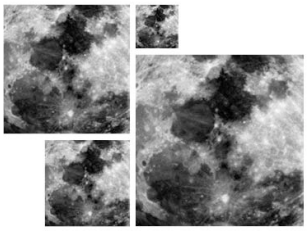

Figure 6. Size Test Sequence

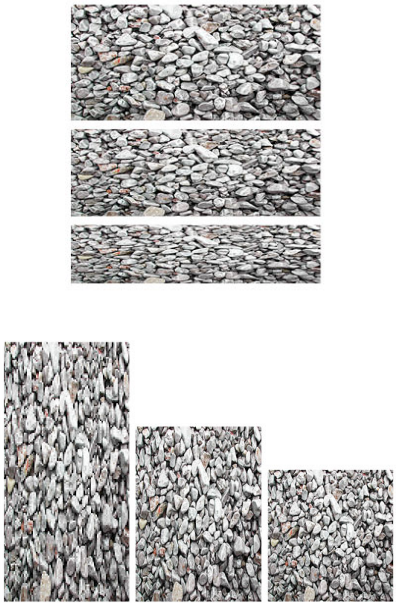

Figure 7. Aspect Ratio Test Sequence

In addition to the two mentioned categories, performance and usability criteria are defined. Each framework description makes a point to state that some basic graphical problem is resolved and the tracker is optimized. These basic issues are collected under performance criteria and it is determined whether or not they were actually solved. Here we mention the constant flicker, a visible motion blur, and the ability to deal with fast moves such that the virtual content is not lost. Registration (the accurate alignment between the real world and the virtual object) is an important factor to overcome, as well as the capability to occlude the virtual object when necessary to create the feeling that the virtual object belongs in the scene, and is part of the scene.

Furthermore, special features are implemented within each SDK for presenting a more comprehensive product. Not all features are supported by every framework. Therefore they are categorized by usability and it is determined which are endorsed. The considered features include face tracking, text detection, flash and usage of the front camera. Moreover, the ability to display the virtual content even when the target image is not in the line of sight anymore, known as extended tracking, is a useful feature in some use cases. Likewise, the possibility to track more than one target image simultaneously comes in handy.

B. Android App

We developed an Android app which integrates the 6 frameworks (ARLab, ARToolKit, D'Fusion, Vuforia, catchoom and metaio) and actively tests the environmental and target criteria. By actively testing, it is meant that a framework is chosen, its camera view opens and a criterion is selected for testing. When the testing environment is set, the user starts the test timer by pressing the Start button. When the virtual content is superimposed into the real world, the testing time is saved as the time needed to detect and recognize the target image.

We determined if the performance and usability criteria are supported by observing the behaviour of the frameworks in a neutral context, on the default target images (no special light conditions). Some feature support, such as face tracking and text detection are set by finding the information in the framework documentation.

Criteria and criteria categories can be added to the list of predefined criteria. Use cases can be defined by adding weighted criteria into a context.

The frameworks can be compared against each other given a constraint or a set of constraints. Furthermore, an overview for each framework is available, displaying the average testing times per framework for each criteria category.

IV. RESULTS

In order to preserve the same testing conditions, such as sunrise light, a criterion is tested by all frameworks before moving on to the next criterion. The testing environment changes between criteria and by testing one criterion at a time for every framework, the environment is minimally altered during the testing.

A. Environmental Criteria

Light intensities are supported overall as observed in Fig. 8. On the left side are the average times for the desk lamp and screen glare light conditions. The graph on the right illustrates the testing times from the semidarkness and sudden change tests. Metaio has the lowest average testing time, followed by Vuforia with an average testing time under a second. The worst timing is registered by ARToolKit while ARLab and catchoom are steady under a second and a half.

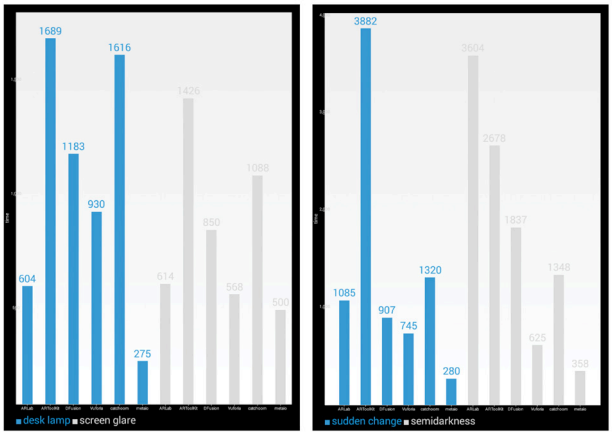

Figure 8. Light Intensities

Most of the frameworks detect 45° angles. The **viewpoint** testing times are shown in Fig. 9, the left graph provides the average testing times for the left and right perspectives, while on the right side are given the up and down viewpoints results. ARToolKit provides good testing times, nevertheless it should be considered that it recognizes only two out of the four perspectives. ARLab has difficulties detecting the target image from a 45° upward angle. The other viewpoints are easier to detect. Three out of the six frameworks detect the target image from a 45° right angle under half a second.

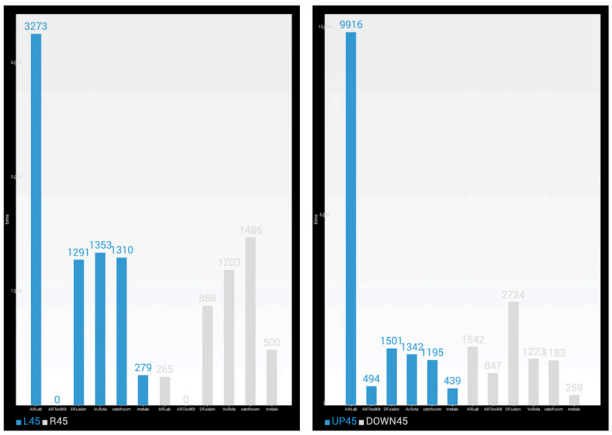

Figure 9. Viewpoint

Detection at 10% and 20% **visibility** requires a longer processing time. Vuforia detects the target at 10% visibility somewhere between one and two seconds, being the only framework that scores this performance. For more than 20% visibility Fig. 10 illustrates that Vuforia's average time is less than a second. Catchoom detects the target image, having only 20% uncovered, in little over two seconds and faster given more visibility. ARLab and metaio need at least 40% of visible area before starting detection, while ARToolKit cannot detect anything under 80% visibility.

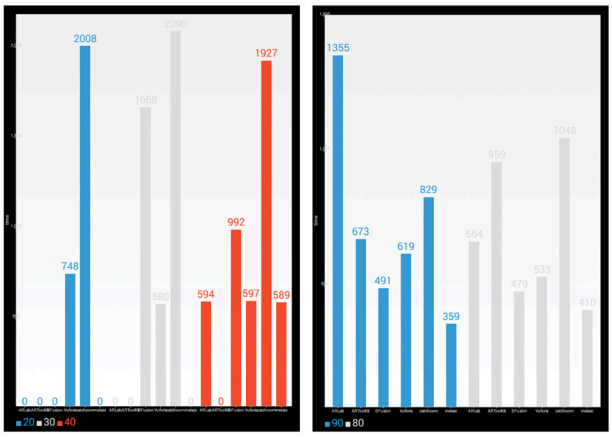

Figure 10. Visibility

Vuforia and metaio have very similar times when dealing with **noise**. However, Vuforia detects a 200% level noise target image while metaio goes as high as 70%. Fig. 11 displays the average testing result for the 10 and 30% noisy targets on the left and 50, 70 and 90% noise levels on the right. The fastest times are still recorded by metaio while ARLab is the slowest detector. Four of the six frameworks have a detection time over one second. Nevertheless, Vuforia is still the most impressive while metaio outperforms D'Fusion and catchoom.

Table II provides the minimum and maximum **distances** supported by each framework. Because there are too many distances to be displayed in a readable graph, Fig. 12 depicts the average over all supported distances per framework. Vuforia is portrayed as the fastest framework when it comes to the distance between the testing device and the target image. Metaio should be on the second place, if not on first place, considering that its average testing time increased because of the extra

effort required to detect the target from 240cm away. The worst timing is registered by the ARLab framework.

TABLE II.
SUPPORTED DISTANCES PER FRAMEWORK

Distance	ARLab	ARToolKit	D'Fusion
10cm	yes	yes	
20cm	yes	yes	
30cm	yes	yes	yes
40cm	yes	yes	yes
50cm	yes	yes	yes
60cm	yes		yes
70cm	yes		yes
80cm	yes		yes
90cm			yes

Distance	Vuforia	catchoom	metaio
10cm			
20cm	yes	yes	yes
30cm	yes	yes	yes
40cm	yes	yes	yes
50cm	yes	yes	yes
60cm	yes	yes	yes
70cm	yes	yes	yes
80cm	yes	yes	yes
90cm	yes	yes	yes
150cm	yes	yes	yes
210cm	yes		yes
240cm			yes

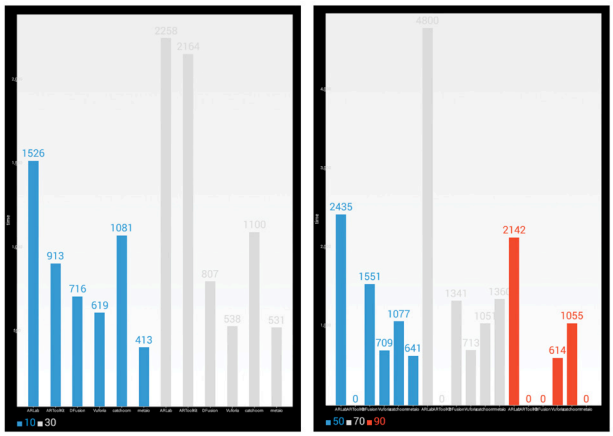

Figure 11. Noise

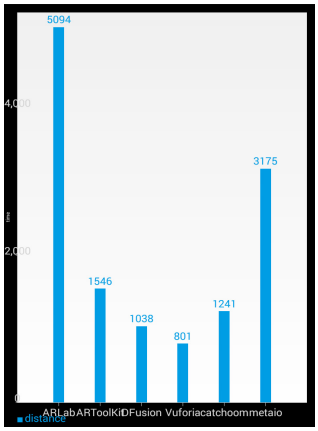

Figure 12. Distance

Fig. 13 demonstrates the average time needed by a framework to detect the target on bright colours versus a target placed against a dark coloured **background**. The left graph illustrates the average over all testing times per framework, and the right one details the two possible cases. D'Fusion supports both bright and dark backgrounds in a constant manner. D'Fusion and Vuforia have similar times dealing with dark contrasts, while Vuforia performs slightly better in the case of brighter backgrounds. Vuforia is the only framework that has stored a better time on bright backgrounds than a darker one. As both graphs can attest to it, metaio is the leader of the background criterion.

B. Target Criteria

We tested both the original target image and its **grayscale** version in order to compare the testing times as shown in Fig. 14. Metaio recorded the lowest testing time, while three other frameworks tested in under a second. Catchoom and ARLab have recorded times a little over a second. As it can be seen, coloured or gray, the detection time is roughly the same. A slight difference can be observed at ARLab.

All frameworks passed the **contrast ratio** test, nevertheless not without difficulties as Fig. 15 concludes. The left graph illustrates the default target testing times (contrast level 0) and the -50 contrast level. The right side shows the average times for the 50 and 100 contrast level target images. ARLab tested poorly and metaio holds the

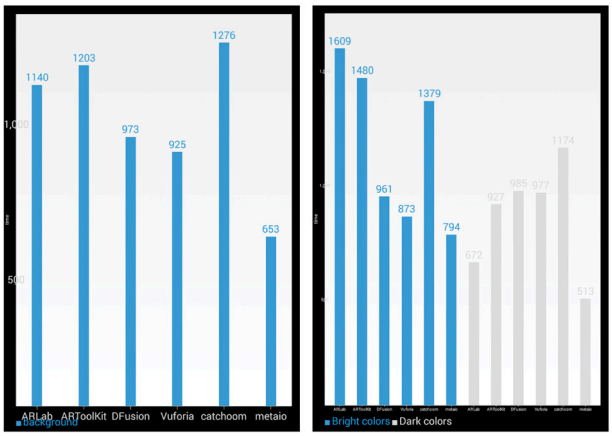

Figure 13. Background

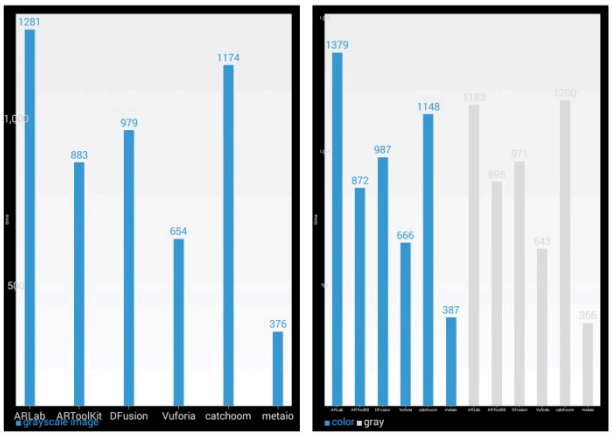

Figure 14. Grayscale

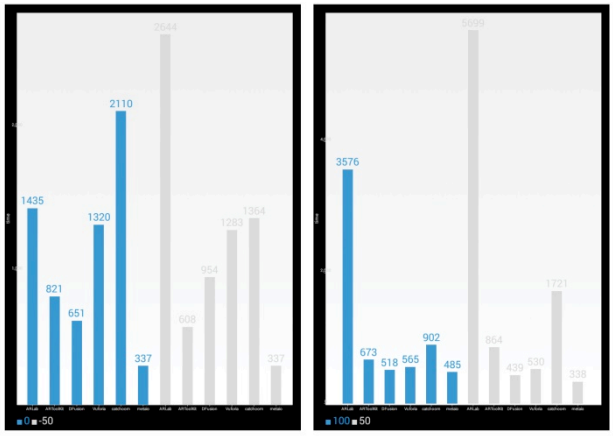

Figure 15. Contrast Ratio

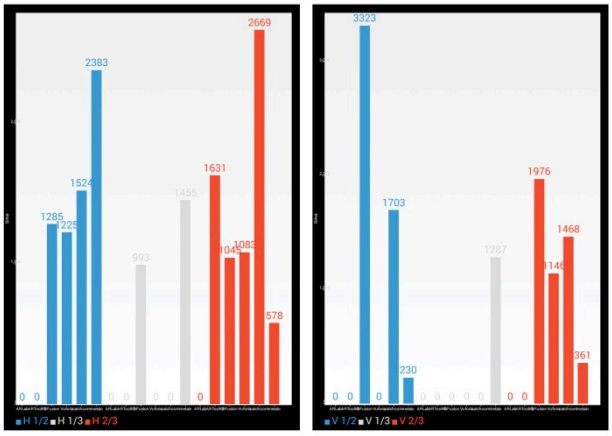

Figure 17. Aspect Ratio

fastest testing times. ARToolKit, D'Fusion and Vuforia also stored average testing times under one second. Catchoom needs on average one and a half seconds while ARLab takes double the time, close to 3 seconds and a half to detect target images of different contrast levels.

The frameworks that detect different **sizes**, keep the average testing times throughout the tests constant as proved by Fig. 16. ARLab only recognizes 10cm high target images and D'Fusion does not detect 20cm high images while ARToolKit only recognizes 20cm high targets. Metaio has the fastest time and close behind are ARLab and Vuforia, both just over half a second. Catchoom tests the highest average time and most of the various target sizes are detected under a second. Catchoom needs over a second when dealing with bigger sizes.

Metaio is the only framework that supports all six **aspect ratio** changes. ARLab, on the other hand, detects nothing. Vuforia is the closest framework to metaio. Nevertheless, metaio still has the best times and covers all constraints. Fig. 17 shows the average results for the horizontal aspect ratio on the left and the considered vertical ratios on the right.

All **materials** are supported by every framework, with the exception of ARToolKit, which was not tested on glass (technical difficulties). The left graph in Fig. 18 denotes the overall average for each framework and the right one breaks down this average by the material types: glass, glossy paper and plastic. ARLab has the best time

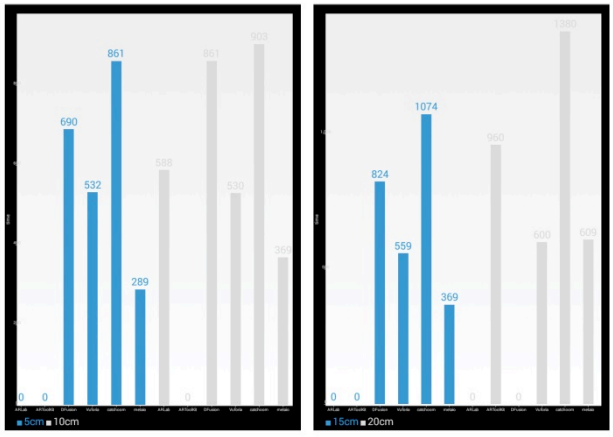

Figure 16. Size

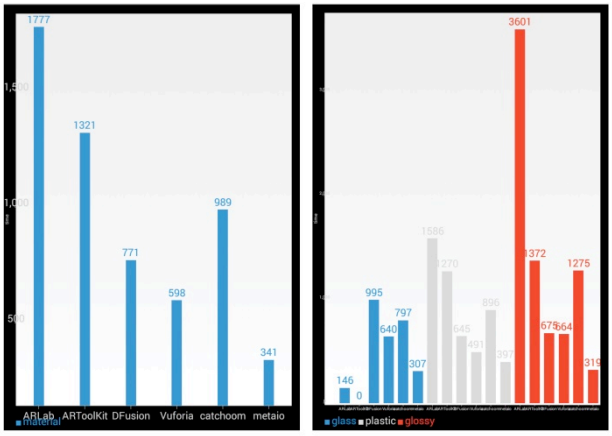

Figure 18. Material

on glass detection and the worst on plastic. Metaio has the lead on glossy and plastic while D'Fusion and Vuforia tolerate them, showing a worse performance on glass. ARToolKit keeps itself constant around the second marker.

V. DISCUSSION

A number of different constraints have been tested on the evaluated frameworks. These constraints raise difficulties for some frameworks while others overcome them easily.

A. Scenarios

Finally, we defined a number of use cases called scenarios. A scenario is a collection of criteria considered for a specific context by acknowledging their importance through weights.

Scenario 1: Consider an indoor app for visualizing large objects like furniture, home furnishings, and appliances. A target image would be placed in the spot where the real object would be, and the virtual object is displayed on top of the target. A couple of constraints that must be considered include:

- How near or far the user can be from the target.
- Having different viewpoints for a better outlook of the virtual object.

- Occluding the virtual object by real objects if necessary.
- A correct alignment between the real scene and the virtual object.
- Using more than one target for a better placement of the virtual object, at the right place and at the right scale.
- Seeing the virtual object even when the target is not visible anymore.

For this scenario, metaio and Vuforia score the highest points. Catchoom is left behind because it does not support extended tracking.

Scenario 2: All target criteria are considered for ranking the frameworks when dealing with a magazine app scenario. An image in the magazine represents a target image for the magazine app. By hovering over with a smartphone the virtual content is revealed to the user. More constraints are considered together with:

- Gray coloured images and strong contrast images.
- Small size images.
- Images printed on different paper types.
- Recognize images given that they are not completely visible.
- Text detection to support the images.

Catchoom and ARLab have a better grasp of the environmental criteria than metaio and D'Fusion. However, metaio supports text detection and catchoom does not.

Scenario 3: The two use cases presented above are indoors scenarios. Now we also consider outdoors scenarios and suppose a company comes up with a new marketing idea and uses AR for posters creatively located on the side panels of bus shelters. A user can either be at the bus stop looking at the poster or notice it from inside a moving bus and try to decipher its message. An outdoors scenario focuses on the environmental criteria along with:

- Sudden light changes, such as the sun hiding behind the clouds.
- How far away the user can be from the target.
- Deterioration of the poster in time due to rain, wind, sun exposure and so on.
- A dynamic background such as people walking behind it or cars driving by.
- How much of the poster is visible from where the user sits.
- Fast moves as the user passes by in the bus.
- Once the poster is recognized, the virtual content is persistent even though the user is in a moving bus.

Vuforia has the lead in this scenario, followed by catchoom. Vuforia provides better support than catchoom for performance and usability constraints.

Scenario 4: Some supermarkets today use AR for promoting their products. Passing by a shelf with various products, the customer can point the camera to the boxes, select a product and initiate a game. For example bursting some bubbles, such that when a predefined number of busted bubbles is reached, the customer wins a discount for the chosen product. A couple of features must be supported and some requirements must be met, such as:

- Support for more than one target image per frame, such that the customer can choose which product he wants to play for.
- A blurred image would make it hard for the customer to play the game.
- The target image can be on a cereal box or a label on a wine bottle.
- Flash might be needed if there is not enough light in the store for the app to recognize the target.
- The label can be torn or the image on the cereal box can be ripped.

Vuforia gains the highest score, closely followed by metaio. The most noticeable difference is given by the support for fast moves; from the tests performed it has been noticed that metaio does not support fast moves.

Scenario 5: The next two use cases require special features, such as text recognition and face detection. An app exists today for translating text from English into Spanish and several other languages. Imagine you are visiting a foreign country in which you do not know the language and need help getting around. This app can be used to translate signs, window ads, and menus. An important feature that the Augmented Reality framework must support is text detection, as well as:

- The contrast between the text and the background where it is written.
- Different materials on which the target is printed, glossy paper, metal plates, glass and others.
- Handle flickering as a constant flicker of the translated word would make it difficult to read.
- How far away the user can be from the target.
- Contamination when dealing with outside signs.

Metaio and Vuforia register the best results for the relevant environmental criteria while no framework distinguishes itself when dealing with target criteria.

Scenario 6: A number of applications already exist for virtually trying on products such as glasses, hats and order them online. Take sunglasses, for example. By using such an app, the customer can browse through the catalogue, choose a pair of glasses, switch between available colours and the sunglasses are projected on the image of the face looking back from the phone. The most important feature that must be supported is face tracking, together with:

- A static background for easier face detection.
- A constant light source or the colour of the skin keeps changing.
- Fast moves lead to losing the tracking.
- The possibility of switching to the front camera.

D'Fusion is the framework that fits best the constraints imposed by such a use case, followed by metaio and Vuforia. Vuforia does not support face tracking, thus loosing points, while D'Fusion beats metaio by supporting fast moves.

Table III shows the recommended framework for each of the described use cases.

TABLE III.
RECOMMENDED FRAMEWORKS

Scenario	Framework
Interior Design App	metaio
Magazine App	metaio
Bus Shelter App	Vuforia
Supermarket Promotions App	Vuforia
Tourist Translator App	metaio
mCommerce App	D'Fusion

B. *Findings*

Testing the frameworks on the default size target from various distances brought some interesting findings. Up close, 10cm away from the target image, the target is in most cases not completely visible. However, ARToolKit and ARLab recognized the targets. This is surprising because ARToolKit needs an 80% visible area of the target, while Vuforia only requires 10%, and still could not recognize the target up close. From these findings, it is concluded that the size of the target image and the distance to it are not strongly correlated.

The light intensity tests revealed a weak ARToolKit tracker when dealing with sudden change in light conditions or a semi dark environment. Each of the considered four sequences for testing light intensities instigates environmental issues. In each case four out of the six frameworks show signs of extra work.

Another surprising result was registered by Vuforia, which can detect a target image with a 200% level of noise. Most of the frameworks can still overcome a 70%, some even 90%, but 200% is a worth mentionable achievement.

VI. CONCLUSION

This paper presented a methodic technique for evaluating open AR frameworks for Android development. A number of evaluation criteria are defined and the results published. As a prototype we developed a test app which is implemented using Eclipse with Android Development Tools (ADT). The prototype can actively test the integrated frameworks and save the testing times in a local database for further comparison.

We would like to state that no AR framework is better than another, each having its advantages and disadvantages. In some circumstances, given a set of constraints, a framework can outperform others.

In future work, we will concentrate our attention on optional features support such as multi-targets, face tracking and text detection. The app itself can be further developed to become a powerful tool of evaluation.

For a more detailed documentation of the presented methodology and results can be found in [7].

ACKNOWLEDGMENT

This work was supported by the Institute of Information Systems and Computer Media of the Graz University of Technology and financed by the Evolaris Next Level GmbH software company.

REFERENCES

[1] TechNavio - Infiniti Research Ltd. *Global augmented reality market 2014-2018*. A market research report.

[2] Juniper Research. *Mobile augmented reality: smartphones, tablets and smart glasses 2013-2018*. A market research report.

[3] Juniper Research. *Over 2.5 Billion Mobile Augmented Reality Apps to Be Installed Per Annum by 2017*. A market research report.

[4] Dominik Rockenschaub. "Entwicklung und anwendung einer systematischen vorgehensweise zur analyse marker basierter augmented reality frameworks für mobile endgeräte.", *Master's thesis*, Johannes Kepler Universität Linz, Linz, Austria, 2012.

[5] Jennifer Lynn. *iOS vs. OS: Current Job Market Is Growing Faster For Android Developers Than Apple*.

[6] Jose Dolz. *Markerless augmented reality*.

[7] Marneanu Iulia. "Evaluation of Augmented Reality Frameworks for Android Development", *Master Thesis*, Graz University of Technology, Graz, Austria, 2014.

AUTHORS

I. Marneanu is a master student at Graz University of Technology, Rechbauerstraße 12, 8010 Graz, Austria (e-mail: iulia.marneanu@ student.tugraz.at).

M. Ebner, Assoc. Prof. is with the Institute of Information Systems and Computer Media of Graz University of Technology, Inffeldgasse 16c, 8010 Graz, Austria (e-mail: martin.ebner@tugraz.at).

T. Rößler is working at Evolaris Next Level GmbH, Hugo-Wolf-Gasse 8, 8010 Graz, Austria (e-mail: Thomas.Roessler@evolaris.net).

This work was supported in part by the Institute of Information Systems and Computer Media of Graz University of Technology Submitted 20 June 2014. Published as resubmitted by the author 14 October 2014.