NFT Marker Recognition in Multi-Marker Environment and Media Integration in ARToolKit

Blagoj Nenovski, Igor Nedelkovski

University St Kliment Ohridski, 1 Maj nn, 7000 Bitola, R. North Macedonia

blagoj.nenovski@uklo.edu.mk; igor.nedelkovski@uklo.edu.mk

Abstract:
ARToolKit is an open source augmented reality toolkit that supports the recognition of fiducial markers and NFT (natural feature tracking) markers. Placing fiducial markers for recognition of outdoor objects can result in economic, legal and logistical challenges. NFT markers are software based and in our platform are distributed by filtering of an area around the user's location. Since there are instances for either multiple markers or a single marker that can be recognized from multiple sides we addressed the effect on NFT maker recognition and tracking in multi marker environment. After successful recognition we focus on displaying various multimedia.

Keywords:
ARToolKit, NFT, AR, Augmented reality, AR multimedia

1. Introduction

To address the problem of ARToolKit not supporting cloud recognition we defined a platform [1] that consists of an app executed on a smartphone and a server that filters the markers and their corresponding multimedia based on an area around a user's location. By targeting as many devices as possible we had to rely only on the smartphones camera for optical recognition. We have addressed the comparison of natural features tracking compared to fiducial markers and the benefit of NFT markers in regards to recognizing outdoor objects, making NFT much larger markers. This enables for recognition from further distances while eliminating the need to place fiducial markers which would occlude the objects. We provided guidelines for generating markers, for resizing the photo prior marker creation, discussed the level of extraction features and emphasized the need for camera calibration. We have addressed NFT marker creation [2] with suggestions on separating the objects façade from the photo and resizing the photo prior the marker creation and their effect on the recognition speed and tracking.

Filtering the markers by user location enables for downloading multiple markers for objects that are in the area of the user's location as well as scenarios where a single object can be explored from multiple sides of its façade in which case we need as many NFT markers as the object's available sides for exploration. For such cases we will address the effect on the recognition speed and tracking quality with increasing number of NFT markers.

In this paper we will point out the evident lack on research on the effects on load times, recognition speed and tracking quality in NFT multi-marker environment. We will determine the load times, the recognition speed and display of a photo with increasing number of NFT markers. After that we will address the display of a photo, a video, sound reproduction, a 3D object and display of web link. We will also address changing the position and orientation of the display multimedia.

2. Related work

There is a comparative study of augmented reality SDK's [3] where the ability for multi marker recognition is one of the aspects that are addressed. Real-time camera tracking method using multiple markers with free camera movement is presented in [4]. ARToolKit can combine several co-planar fiducial markers into multi marker set. Single marker and multi marker tracking is tested in [5]. Multi marker tracking in ARToolKit is implemented by tracking all loaded markers separately and then

combining all the tracking results. Tracking multi markers is slower and comes with higher computational cost compared to single independent markers. Multi marker recognition is addressed in [6] with focus on finding the optimal values for multiple marker attributes such as: marker size, marker distance from camera, marker speed, environmental brightness, contrast level and the correlation between the marker size and the distance from the camera. Multi-marker approach for increased tracking robustness is addressed in [7] and research on determining a confidence factor for tracking multiple markers with ARToolKit in [8]. Multi marker tracking is suggested in [9] for ensuring sufficient accuracy by adding single markers to multi-marker setup. An automation method for calibration of multiple fiducial markers in order to obtain stable relations among markers is presented in [10].

All of the before mentioned research is based on fiducial markers and research on the effects such as load times, recognition speed and tracking quality with NFT markers in a multi-marker environment is practically non-existent.

3. Methodology

To determine the recognition speed we used ARToolKit's feedback on the state of a marker being loaded and a marker being recognized. We achieved this by subtracting the marker load time from the marker recognition time. Our focus when creating markers was to create better markers that could easily be recognized and tracked by entry level devices. That is why we used: Samsung J3(2017) and Samsung J4+ (2018) as test devices. We also added an additional flagship device: Samsung S9+ (2018) so we could compare the load times and the recognition speed. To eliminate the advantage in a scenario where the marker creation device is the same as the smartphone used in our tests we used a different smartphone to take the marker photos. To remove the parts that are not needed for object recognition and to enable quality display of multimedia over the recognized object, prior the marker creation we extracted the objects façades from the photos. Based on the conclusions of our previous work the images were resized to 1000 pixels before creating the markers.

When testing we put the smartphones in a fixed position and run the tests simultaneously. In the real world test we simulated the use of the app in a perspective of a user that is using the app for the first time: holding the smartphone in a natural position and then pointing it at the object until the marker is recognized. After that we simulated various intensity phone movements. All of the tests were done with sampleRate set to 30 and cutoffFreq parameter set to 15.

Visualized results of the recognition speed are a representation of subtracting the marker recognition time form a baseline of 5000ms, a limit we defined for a good user experience.

4. Time required for: loading the markers, recognizing the markers and displaying a photo when using additional number of markers

Our defined platform is initially expected to contain a small number of markers at a specific location. By enriching the content on the server itself, there will be markers for objects that are at a geographical distance smaller than the values set for filtering the markers. The impact of a different number of markers that can be used to recognize the same object on different mobile phones can determine the degree of a quality user experience.

Our goal was to measure the time required for loading the markers and their recognition speed.

We also provide an overview of the time it took to load different number of markers to determine whether it can affect the overall user experience. For the purposes of this validation, in addition to the marker that we used for recognition (1), additional 5 markers (2, 3, 4, 5, and 6) were created. All markers used a resized photo to 1000 pixels, a DPI value of 48, 2 for tracking features, and 1 for initialization features. We created scenarios for each of the set markers in a separate directory. The first test loaded only the marker that was used for recognition. After the first test was run, the recognition marker and an additional marker were loaded from the directory in numerical order. In this test, the same marker was recognized again, but two markers are loaded. The tests continued by

adding one additional marker for each of the scenarios. Table 1 shows the file size and number of features for each of the markers.

Table 1:

File sizes and number of features for each of the markers

	File sizes in KB				features	
Marker	**iset**	**fset**	**fset3**	**Sum**	**fset**	**fset3**
1	28	1	67	96	44	517
2	35	2	62	99	39	450
3	36	1	71	108	75	563
4	34	2	55	91	75	423
5	17	2	73	92	28	543
6	32	1	59	92	56	478

From the test results we noticed that loading an additional marker takes up to a maximum of 10 milliseconds. The difference between the load times of one marker and the additional 5 markers does not exceed 50 milliseconds for low-end phones. These values are low and do not play a big role in the overall user experience.

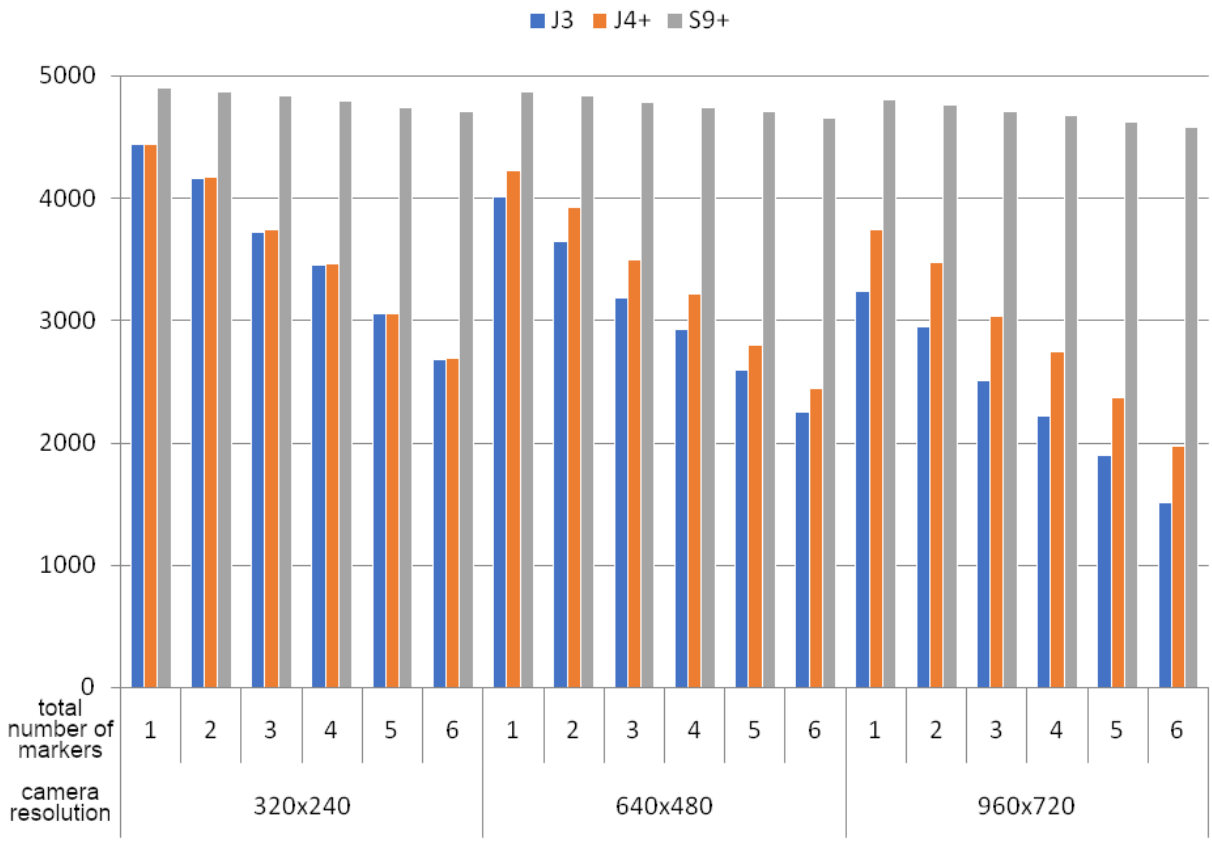

Figure 1. Recognition speed with different number of total markers

From figure 1 we can see the time required for marker recognition when multiple markers are included. We can see that the total number of loaded markers is directly proportional to the recognition time. This trend is repeated for the three camera resolution values tested, i.e. adding markers further reduces the recognition speed by an almost identical time for each additional marker.

Analyzing the data from the high-end phone, we can see that regardless of the resolution used, even in the case with the recognition marker plus additional 5 markers, we obtained high recognition speed. The customization in the process of creating the markers, as well as the selection of the camera

resolution, makes our platform suitable for low-end devices, but also allows for increasingly better results when using high-end phones.

In addition to the recognition speed we did additional field analysis on the tracking quality. We noticed that once a marker is recognized, in each of the cases regardless on the number of additional number of markers, the tracking was stable. From this we can confirm that the working principle of ARToolKit is to focus on the recognized marker as long as it is visible in the camera view.

With real word use of the app by the end users, phone movement is expected with changes in the camera view in order to explore the object from different perspectives. Various factors, such as rapid phone movement or a recognized marker leaving the camera view can lead to the marker not being recognized. In such a case, it should be taken into account that the speed of its re-recognition is in the range of the recognition speed obtained from the tests. This time can be reduced by setting a buffer space of several seconds. In such a case, upon stopping the recognition of a particular marker within the given buffer period, the application would only concentrate on recognizing the last recognized marker. The version of ARToolKit we were working with didn't allow dynamic loading of markers while the camera view is displayed, which prevented us from implementing that idea.

5. Multimedia content display in ARToolKit

Augmented reality applications can be rich in multimedia content. Since the available demos of the ARToolKit Android app only include a pre-generated cube, next we worked on the display of a photo, a video, sound reproduction, display of a web link and a 3D object.

5.1. Photo

When displaying content from the past, most of the time it would be presented in a form of a photo. To display a photo we drew a plane with 2 triangles in OpenGL [11] and applied the selected photo as a plane texture. The default width of the plane is set to 300 pixels, and the height is obtained from the aspect ratio of the photo. ARToolKit yields a transformation matrix that contains the marker's position, orientation, and skew relative to the camera. This transformation is given to OpenGL and even though the photo is in 2D, it is displayed in a 3D environment. This gives the effect that the image is part of the physical world. A more realistic experience can be achieved when using transparent photos in png format. We strongly suggest erasing parts that are not related to the object that we want to display, such as the background, trees, obstacles or side objects. To contribute to a more realistic experience we suggest applying a gradient between the transparent part and the object being displayed.

Figure 2. Representation of an object’s past with a photo

5.2. Video

For displaying a video we use the same method of drawing a plane as in the case of displaying a photo. We initially tried continuous extraction of a single frame from a video and plotting the result onto the plane [12]. Such a method proved to be inefficient due to the high use of computational resources. We managed to optimize the app to a degree but even in cases where we had a reduction in the reproduction interval the results didn't provide realistic user experience. By giving the plane an external texture [13], we got an effective rendering of the videos.

5.3. Sound

In addition to a picture and a 3D object, a sound or a song can also be played. This option is removed when a video is selected because the sound is part of the video itself. Only a sound can be played without coupling it with a photo or a 3D object. We use the MediaPlayer [14] component from Android to play sounds. Sound can be paused when an object is not being recognized and resumed on re-recognition, or can be paused manually.

5.4. 3D object

To display a 3D object we needed a 3D engine. To display 3D objects we initially included the ArToolKitJpctBaseLib [15] library which includes concepts such as: camera, 3D-object and textures and supports the following formats: 3DS, OBJ, MD2, ASC and XML. The library is under the same license as ARToolKit.

Our initial display approach was the same as with a photo display, i.e. the 3D object was loaded once the marker was recognized. Usually the 3D objects are larger compared to photos and might need to be built before the marker is recognized. Real world tests with this library showed us that building and rendering the objects after the marker is recognized resulted in a display delay. We made an effort to optimize the application to load and build the 3D objects the moment it receives the configuration file. When a marker is recognized the 3D object would be ready for display after recognition, reducing the delay time between the markers recognition and the 3D object display. Real testing with this approach showed us that the time required to build the model is long and does not provide a quality user experience.

Figure 3. Display of a 3D object

Next we tried 3D Model Viewer [16], a 3D object display engine available as open source software. Android 3D Model Viewer is a demo of OpenGL ES 2.0 and has the ability to load

Wavefront OBJ, STL and DAE files. The application does not use additional libraries and the nature of its code being open source allows for robust application. It is available on the Play Store, making it suitable for testing and previewing a model before it is added to the platform.

By testing with several different models, we realized that the building process of the models and their display are significantly faster, and the possibility of transformations, such as changing the size, position and orientation, gave us a good synergy with our platform. The built model is given the pose of the marker in which the model is displayed. As with other visible multimedia contents, we have the possibility to change the size, position and orientation with 6 degrees of freedom.

5.5. Web Link

To receive additional information about the recognized objects we added the ability to display web links from the platform itself or other web pages. Web links are opened as a WebView component that is part of the Android operating system.

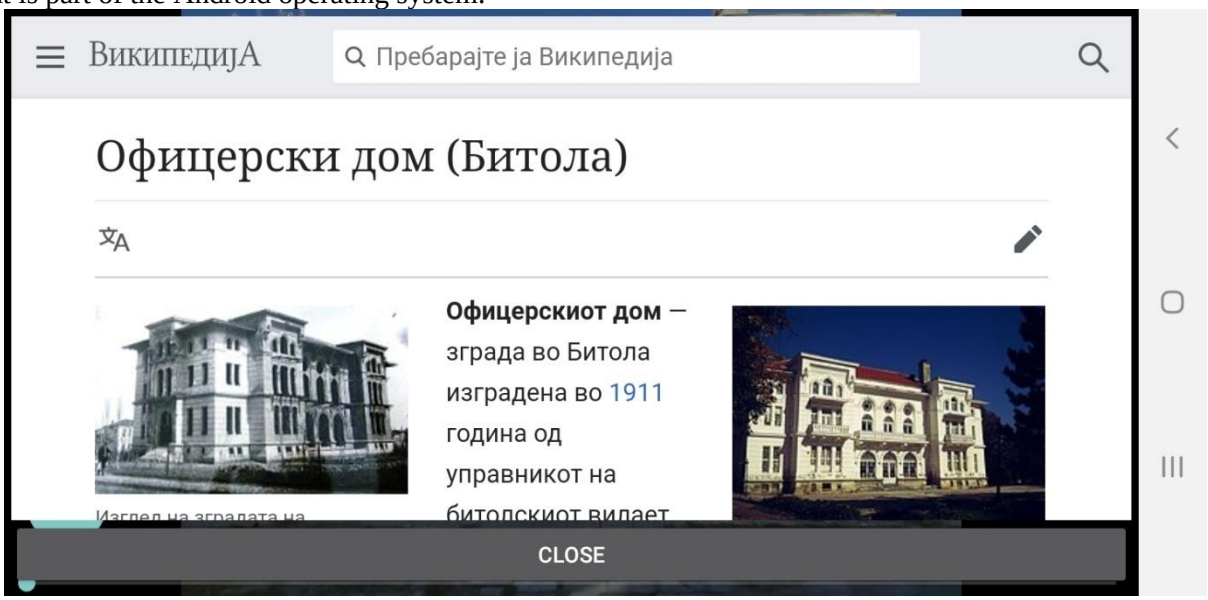

Figure 4. Web link

5.6. Changing the position and orientation of multimedia

Real world tests showed us that after adding a marker and its display objects, when displaying the multimedia over the recognized marker, there is usually the need to change the size, position and orientation of the displayed multimedia. The default setting for the size of the plane on which photos and videos are displayed is set to 300 pixels in width and a variable height in proportion to the content being displayed. By using 640 x 480 pixels as the default camera resolution, choosing a value of 300 pixels is an adequate size for a display that does not take up majority of the screen. However, the display content may not be the correct size relative to the marker. By increasing or decreasing the size of the plane, we can adjust the size to be equal as the size of the object in the camera view.

Since we have 3D registration, to address the need of 6 degrees of freedom, we can move the plane along the X, Y, and Z axes, or rotate it along any of the axes. In that way, by repositioning the objects, we also allow for robustness in cases where instead of the facade itself, we use a door or another/auxiliary object for recognition.

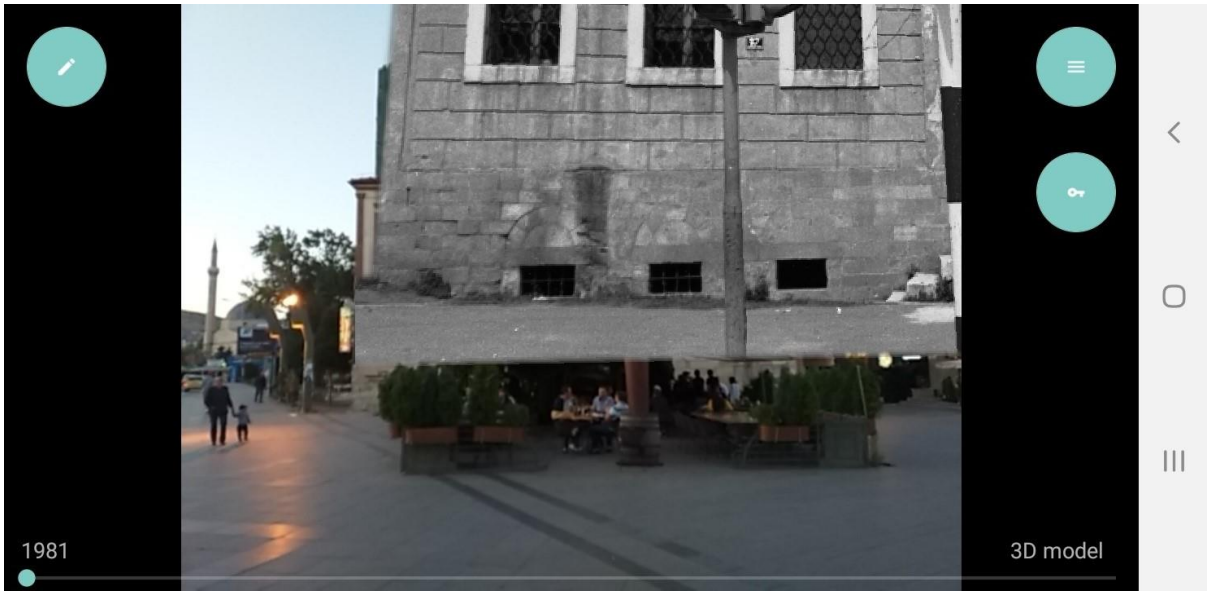

Figure 5. Initial photo display before resizing and positioning

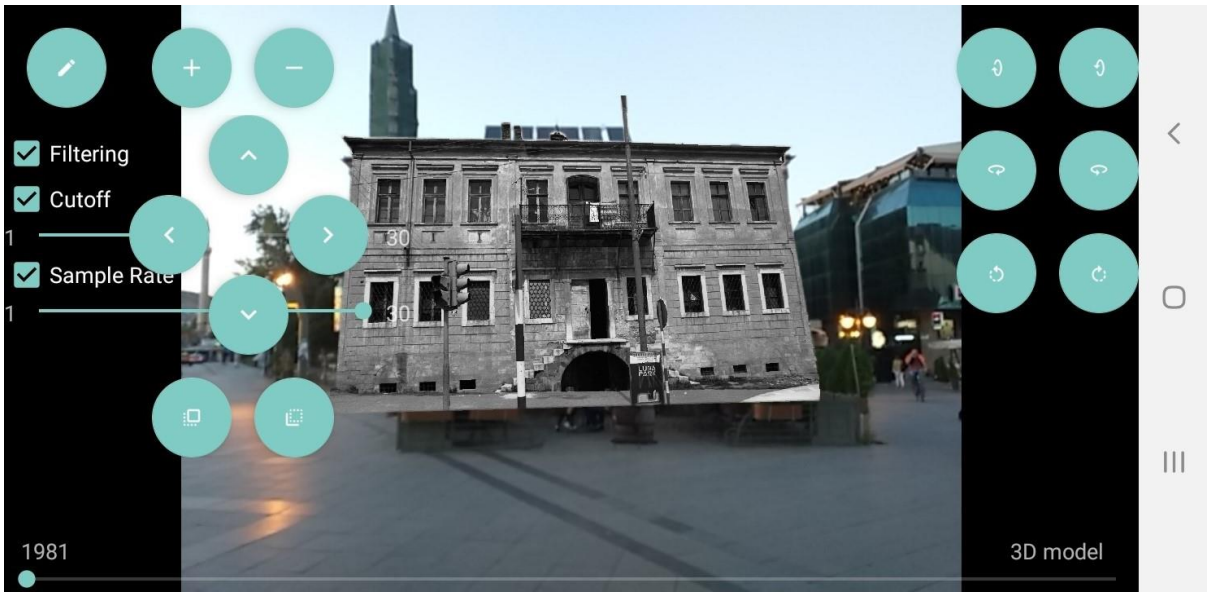

Figure 6. Adjusted size and position of the photo

6. Conclusion

In this paper we addressed the impact of different (total) number of markers when recognizing NFT markers on the load times and marker recognition speed and tracking. Load times of an additional marker take up to 10 milliseconds on low end smartphones. In a scenario with 5 additional markers, 50 milliseconds do not play a big role in the overall user experience. The total number of loaded markers is directly proportional to the time required for recognition for each of the tested camera resolutions. When loading additional markers with similar file sizes and similar number of natural features the recognition speed is reduced by an almost identical time. When using a high end smartphones, high recognition speed is obtained even in cases with 5 additional markers. On site testing proved us that tracking is not affected with additional markers, but we have to emphasize the effect that additional markers have on the recognition speed. In real world use, once a marker is not recognized, re-recognition time is dependent on the total number of additionally loaded markers. ARToolKit's inability for dynamic marker loading does not allow us to implement a buffer technique for certain a time interval for marker re-recognition.

For display of a photo we drew a plane and applied the photo as a texture. For more realistic experience we suggest using transparent photos of only the display objects and a gradient between the object and the background. For video play we gave external texture to the drawn plane and for sound we used MediaPlayer from Android. For 3D objects we incorporated 3D Model Viewer which provided much better results compared to the "default" ArToolKitJpctBaseLib library. Display of photos, videos, and 3D objects usually need resizing, repositioning or changing the orientation. We implemented an in app functionality to address this issue.

References:

[1] B. Nenovski and I. Nedelkovski, "Defining a feature-rich end-to-end augmented reality platform for spatial exploration," in Proceedings/8th International Conference on Applied Internet and Information Technologies, vol. 8, no. 1, pp. 103-108, "St Kliment Ohridski" University-Bitola, Faculty of Information and Communication Technologies-Bitola, Republic of Macedonia, 2018.
[2] B. Nenovski and I. Nedelkovski, "Recognizing and tracking outdoor objects by using ARToolKit markers," International Journal of Computer Science & Information Technology (IJCSIT), vol. 11, no. 6, pp. 21-28, 2019.
[3] D. Amin and S. Govilkar, "Comparative study of augmented reality SDKs," International Journal on Computational Science & Applications, vol. 5, no. 1, pp. 11-26, 2015.
[4] J.-H. Yoon, J.-S. Park, and C. Kim, "Increasing camera pose estimation accuracy using multiple markers," in International Conference on Artificial Reality and Telexistence, pp. 239-248, Springer Berlin Heidelberg, 2006.
[5] D. Wagner and D. Schmalstieg, "ARToolKitPlus for pose tracking on mobile devices," in Conference: Proceedings of 12th Computer Vision Winter Workshop CVWW07, 2007.
[6] I. Rabbi, S. Ullah, M. Javed, and K. Zen, "Analysis of ARToolKit fiducial markers attributes for robust tracking," in International Conference of Recent Trends in Information and Communication Technologies (IRICT'14), University Technology Malaysia, pp. 281-290, 2014.
[7] K. Anagnostou and P. Vlamos, "Square AR: Using augmented reality for urban planning," in 2011 Third International Conference on Games and Virtual Worlds for Serious Applications, pp. 128-131, IEEE, 2011.
[8] I. V. Vista, P. Felipe, D. J. Lee, and K. T. Chong, "Remote activation and confidence factor setting of ARToolKit with data association for tracking multiple markers," International Journal of Control and Automation, vol. 6, no. 6, pp. 243-252, 2013.
[9] D. F. Abawi, R. Dörner, and S. Reinhold, "Interactive specification of virtual objects' poses in an AR system using multi-marker tracking."
[10] D. C. B. de Oliveira, F. A. Caetano, and R. L. D. S. Da Silva, "A method to automate the calibration of a multiple fiducial marker setup," in 2014 XVI Symposium on Virtual and Augmented Reality, pp. 89-95, IEEE, 2014.
[11] The graphics pipeline, URL: https://open.gl/drawing.
[12] MediaMetadataRetriever, URL: https://developer.android.com/reference/android/media/MediaMetadataRetriever.html#getFrameAtTime(long).
[13] GLES11Ext, URL: https://developer.android.com/reference/android/opengl/GLES11Ext.html.
[14] MediaPlayer, URL: https://developer.android.com/reference/android/media/MediaPlayer.html.
[15] ArToolKitJpctBaseLib, URL: https://github.com/plattysoft/ArToolKitJpctBaseLib
[16] Android 3D Model Viewer, URL: https://github.com/andresoviedo/android-3D-model-viewer.