Scientific Visualization, 2021, volume 13, number 1, pages 83 - 103, DOI: 10.26583/sv.13.1.07

Developing of USDZ models for 3D digital analysis results visualization in augmented reality

R. I. Kadyrov[1]

Kazan Federal University (Institute of Geology and Petroleum Technologies)

[1] ORCID: 0000-0002-7566-6312, rail7777@gmail.com

Abstract

The active development of augmented reality (AR) technologies in recent years leads to their increasing application in various fields of sciences. Especially the use of AR can be promising in areas where the results are represented in 3D, have complex shapes and it is often necessary to show them at different angles and zoom, such as digital rock analysis. One of the possible approaches is creating AR-models in the open format of Universal Scene Description Zipped (USDZ), developed jointly by Apple and Pixar and currently supported by a huge number of different mobile devices. This 3D format is distinguished by advanced texturing based on Physically Based Rendering (PBR), the ability to support high-poly models, 3D animation, sounds, creating complex scenes with the assignment of reactions to individual objects and ease sharing using emails, messengers, and websites. This work briefly describes the basics of USDZ models including file structure, meshing, UV mapping, and texturing. It also discusses the methodology for AR-models development based on the results of digital analysis using free software starting from CT-scanning and finishing with exporting. In addition, the information about the limitations of USDZ models and prospects for the development of using USDZ models were provided. The results can be used for native demonstration of 3D research output, quick access and exchange the visual information, for exhibitions in museums and educational purposes. AR models make it easy to share the 3D visual results with people, which can be integrated into websites and used for the support of the scientific publications or native demonstrations on conferences.

Keywords: Augmented reality (AR), X-Ray computed tomography (CT), Universal Scene Description Zipped (USDZ), Digital rock analysis, High-poly mesh, Visualization.

1. Introduction

Augmented reality (AR) is a real-time viewing of information superimposed on the real world. A local processor associated with a data source generates information, which can include visualization, sound, video, or location data. In the contrast, virtual reality (VR) totally simulates real world [1].

The development of AR started in the 1950-s, however, the devices close to augmented visualisation have been created since 1862 [1]. In 1961, Philco Corporation developed the first “Headsight” virtual reality helmets for military use and this was the first application of technology in real life [2]. The term augmented reality firstly was used by Tom Caudell in 1990 [3]. Lewis Rosenberg assembled one of the first working AR systems for the US Air Force in the form of an exoskeleton that allows the user to remotely control machines in 1992 [4].

Despite the gradual development of AR technologies in sciences, some fields remain unaffected. For example, AR in geosciences was used only for demonstration of geological data in the field [5,6], interactive maps [7], geological underground structures [8], 3D LIDAR and radar data from Antarctica [9] and interactive exploration of the mineralogy of rock surfaces [10]. It also has been used to show and interact with well data [6] and for assistance to the geologist during fieldwork by visualization of a 3D terrain and incorporation of geo-tagged virtual elements [11]. However, the use of AR especially can be promising in areas where the results are represented in 3D, have complex shapes and it is often necessary to show them at different angles and zoom, such as digital rock analysis. Since it is based on the method of X-ray computed tomography, the original grayscale image is already three-dimensional. Next digitizing of the porous volume and matrix of the reservoir as well as the results of numerical simulation of different petrophysical processes as permeability, electrical conductivity, and elastic moduli can be also visualized in full 3D [12]. Thus, the data obtained during these works have all the prerequisites for being easily transformed in the required file format and visualized in the AR.

The application of AR to visualize the 3D results of digital analysis can be used for native demonstration of research results, quick access and exchange of the visual information, for exhibitions in museums and in educational purposes. A great advantage of AR is the ability to demonstrate a 3D model on mobile devices without the use of specialized programs that require significant computer resources. AR models make it easy to share the 3D visual results with people, which can be integrated into websites and used for the support of the scientific publications or native demonstrations on conferences.

There are several options for creating AR models, the most obvious of which is the direct writing of the mobile application code, which includes the digital 3D model itself and the targets and can visualize the object over a specific real-world object or on a plane. A similar, but simpler and less laborious way is to use special cross-platform development environments, for example, such as Unity[1] with Vuforia Engine[2]. An example of AR visualization of acid-dissolved part of carbonate sample by a mobile application developed by the author based on this method is shown in Fig. 1. However, this approach has many disadvantages. Including the meshes and targets directly in the program code, which leads either to the need to edit it constantly or to the complexity of the code to be able to import new objects. Supporting such a mobile application for various operating systems, their versions and devices is a developer's nightmare and is not feasible by individual researchers. In addition, a lot of effort requires visualization of textural effects and support of animation.

At the WWDC Conference 2018 Apple company, announced that it has developed a new file format, working together with Pixar, called USDZ Universal. This format allows developers to create 3D models for augmented reality [13]. Universal Scene Description (USD) is the first open-source software under a modified Apache 2.0 license [14] which allows to widely share and create augmented 3D scenes based on many simple assets (USDZ is the zipped version of the USD file) [15].

The main advantages of this AR-format are the lack of need to develop and support specific AR applications since all devices with iOS 12 and macOS computers already support .usdz file by default. Thus, today more than 1 billion devices [16] can demonstrate a 3D digital model created in this format. This 3D format is easy to share by sending .usdz file by email, messenger or publishing on a website. In addition, USDZ format supports advanced texturing of 3D models, 3D animations, sounds and reactions.

It should be noted that the author also considered the possibility of using the Android mobile platform for AR visualization, for example, based on the glTF format, which has many similarities with USDZ [17]. However, due to the huge number of devices on Android and various firmware versions, only a small part of them are capable of supporting AR. However, more and more new models of Android smartphones have this feature [18]. Moreover, at the time of this writing, both individual solutions for converting glTF to USDZ are noted [19], as well as signs of direct support of the USDZ format by Android [20,21].

This article discusses the methodology for creating .usdz files from the results of digital analysis (first of all digital rock analysis, because the author specializes on geosciences), the associated limitations for AR models, features of texturing and additional possibilities for the interaction of AR models with the user. We consider developing the models for two samples starting from CT-scanning to AR-visualization.

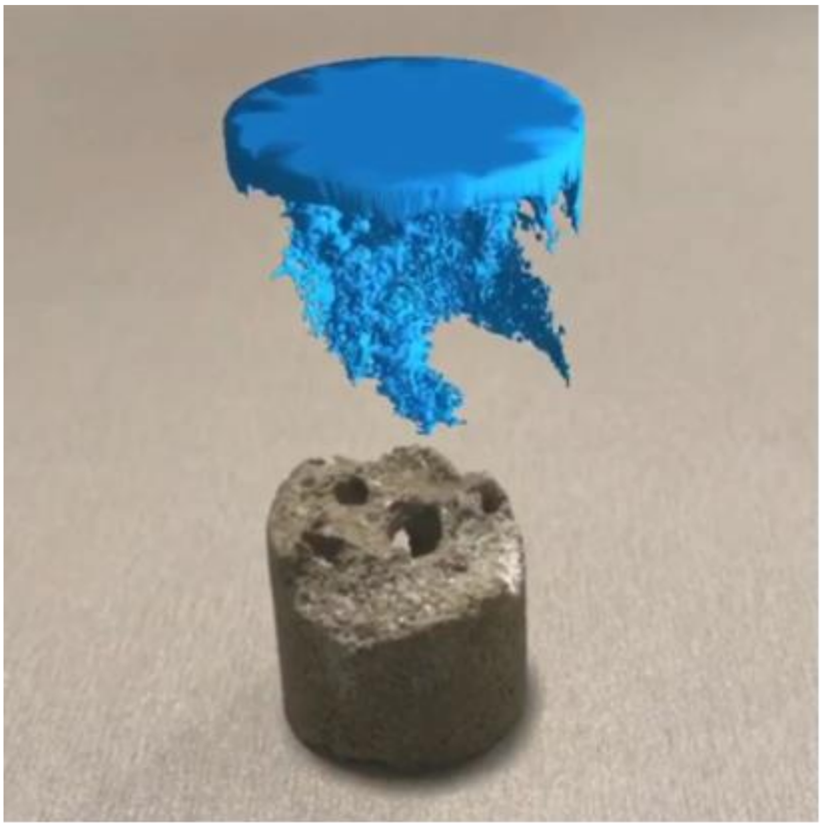

Fig. 1. An example of AR visualization of a part of acid-dissolved part of carbonate sample (blue), implemented using the Unity and Vuforia software. When the camera of a mobile device is aimed at the target (carbonate sample), the application recognizes the object and renders a digital model above it. The target and the model are already included in the application code and strictly fixed in space with each other, so rotating the model or the camera of the device will display the model from the other side. Please view the article on the journal website to see the animated version of the figure

2. Essentials of USDZ model

2.1 USDZ format short characterization

Universal Scene Description is a framework for developing, reading and transferring 3D scenes, which is used for exchange between graphics applications [22]. It is based on the concept of a single object in terms of marshaling and transfer, which can be streamed and used without unzipping. This translates into simplified content delivery and applicability across a wider range of platforms [23].

A .usdz package is a .zip archive (without compression) that can contain following file types [23]:

- .usda, .usdc, .usd (these formats are dedicated to store 3D models, however .usdz archive includes only .usdc file at the moment);
- .png and .jpeg formats for pictures or textures;
- .m4a, .mp3, .wav files for built-in sound.

The .usd file represents a scene with hierarchy of nodes, .usda file contains of .usd scene in ASCII format (human-readable), .usdc file comprises .usd scene in binary representation. The minimum basis for creating an .usdz file is a mesh, reflecting the shape of a 3D object, which is subsequently encoded into a binary .usdc file in the package (in this case, USDZ models will be colored gray by default). However, since the main goal of this work is to visualize 3D structures, in this article we will also consider the creation of 3D textures that are included in the .usdz file in the form of images which are linked to a 3D volume by UV mapping.

2.2 Mesh

A polygonal mesh is a spatial set of vertices, edges, and faces used to describe the morphology of an object. There are certain rules in the construction of a mesh, which it must satisfy: (a) any vertex must be connected to at least 1 edge; (b) any edge must belong to at least 1 face; (c) mutual penetration of two faces is impossible if there is no vertex or edge in the intersection area [24]. The polygonal mesh is the basis of the USDZ model, defining the shape of the object.
The faces can consist of triangles (triangle mesh – Fig. 2A), quadrilaterals (quads – Fig. 2B), hexagons (hexa mesh – Fig. 2C) or other simple convex or concave polygons (n-gons) [24]. First two types are the most common and are supported by many applications, including USDZ format.
Another important concept related to a mesh is decimation, which means reducing the number of polygons to simplify the shape of the object in order to reduce the hardware resources spent on rendering them (Fig. 3) [25]. Various approaches to reduce the number of mesh elements while retaining most of the overall morphology exist. Some of them can create an additional texture for the normals, which allows simulating small details of the shape that are not present in the simplified mesh structure. The latter also allows you to speed up the display of the object [26].

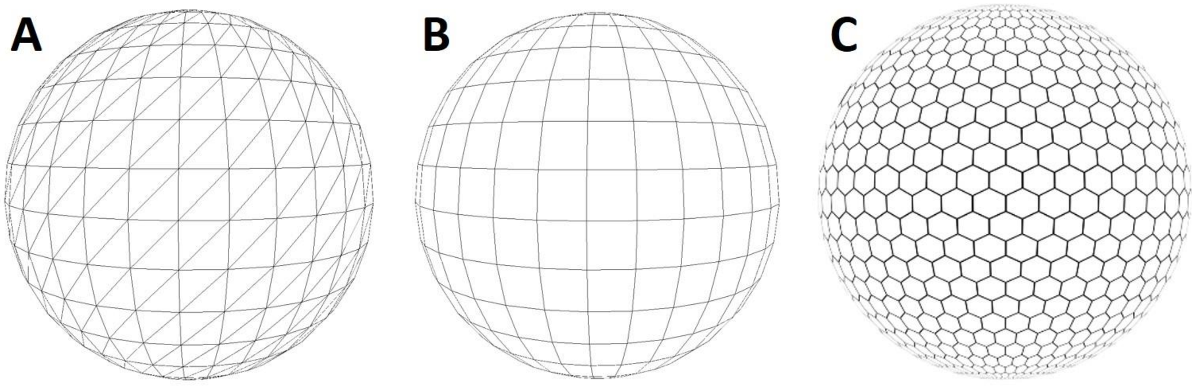

Fig. 2. A spherical mesh consists of different faces types: A – triangles, B – quadrilaterals, C – hexagons

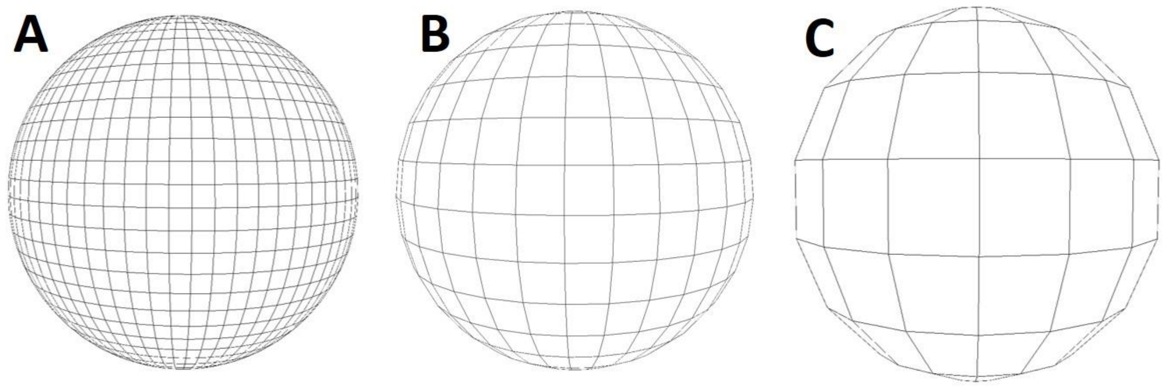

Fig. 3. A decimation of spherical quad mesh from A to C by reducing the number of polygons and increasing their area. The shape of the sphere becomes "coarse" in the process of decimation

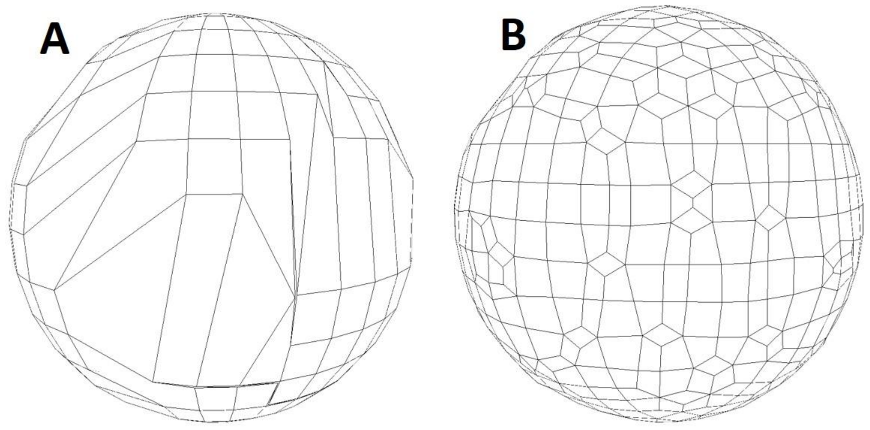

Fig. 4. Remeshing of spherical quad mesh with irregular face sizes (A) to much regular ones (B)

One more useful feature of mesh editing is remeshing, which involves generating new mesh topology while maximal preserving the geometry of the object (Fig. 4). Meshes obtained by scanning devices often have many excess elements. In addition to the task of reducing their complexity, an important goal is to improve the quality of the mesh, which consists of ordering the position and density of vertices, as well as the shapes and sizes of faces, which makes it possible to simplify the process of rendering, transferring and data storage [27].

2.3 UV mapping

UV mapping is the process of unfolding 3D shapes into a 2D plane for the following association of a texture map with a surface of 3D model. The symbols "U" and "V" represent axes on mesh unwrapped to the plane, since the standard XYZ notation is already assigned for the 3D axes. The UV coordinates represent the locations on the UV map to which the corresponding texture regions will be attached. After the UV mapping procedure, the texture is fixed on the 3D surface of the object. When the position of its individual parts changes, the texture will also change with these parts [28].

UV mapping is a complex and time-consuming problem in 3D model development. As an analogy, a cartographic example of translating the shape of the Earth into a flat map can be given. Thus, approach similar to cartography can be used for UV mapping. Normally, they include projecting a 3D shape onto a sweep of one of the correct geometric shapes: cube, cylinder (Fig. 5), sphere, etc. Often, it is required to create a UV map of complex 3D shapes that cannot be approximated by simple geometric shapes (e.g. in game development). In this case, more complex automatic algorithms or manual UV mapping are used in which the sweep of individual 3D elements can be carried out in separate blocks.

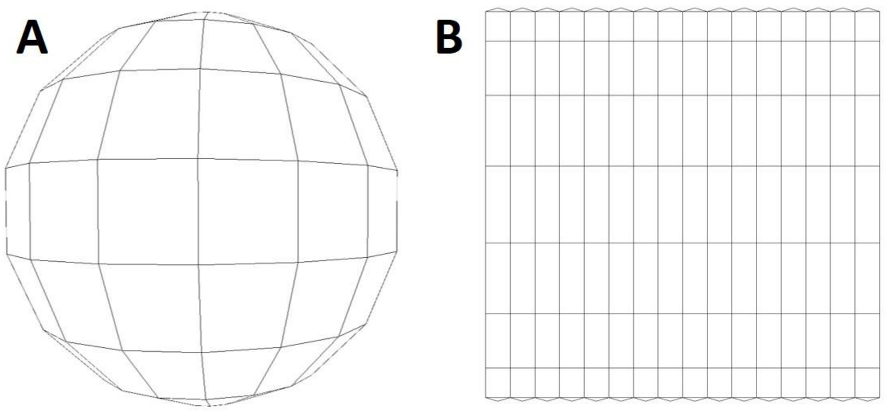

Fig. 5. Example of UV mapping of spherical mesh (A) to flat map (B) based on cylindrical projection

After creating the UV map, textures are imposed on it. The values of the UV coordinates connect the surface of the 3D model with the texture map and are saved in the model file.

2.4 Texture mapping

Texture mapping is a technique of assigning the properties to the surface of a 3D body for displaying on it the details that are actually absent in the object's morphology. Usually textures include at least color, opacity and normal maps, but all of them depend on the spectrum, intensity and direction of the light reflected from the surface by changing which, you can change the perception of the object’s geometry [29].

Table 1. USDZ textures and their brief descriptions

Texture	Brief description
Diffuse color (albeldo)	represents the proportion of incident light that is reflected diffusely from a surface [30]
Normal	used to simulate lighting irregularities and indentations without using additional polygons [31]
Emissive color	represents colour of a glowing object that is intrinsic to a surface, rather than coming from light from other sources that is reflected by the surface [30]
Metallic	represents areas on a texture set or material that behaves like a metal [32]
Roughness	describes asperity of the surface, which induce light diffusion [32]
Occlusion	represents areas of the model which should receive high or low indirect lighting coming from ambient lighting and reflections [33]
Opacity	represents areas of the model with different values of transparency [32]
Clearcoat	changes the perceived thickness of a coating of clear lacquer [34]
Clearcoat roughness	changes the roughness of clearcoat specular [34]

The USDZ format uses Physically Based Rendering (PBR) textures, which help to make 3D models look more realistic when AR-model is placed in the real world [33,35]. The texturing based on PBR-technique demonstrates a more correct visualisation of the interaction between light and object surface [32]. PBR uses a set of maps, each of which is responsible for its own aspect of the display. Each texture map is saved in the .usdz archive in the raster format of .jpeg or .png files and linked to the 3D model via a UV map. The list of texture maps used in USDZ and their brief description is presented in Table 1. Thus, the importance of PBR texturing of USDZ models relates not only with improving the demonstration quality (especially in AR mode) but also with adding some visual information that is initially absent in mesh, which allows decreasing the file size.

3. Methods

The proposed procedure of AR model development involves the model processing steps of using third-party free software. The schematic representation of the process with the designation of the used programs is shown in Fig. 6 and described in details below.

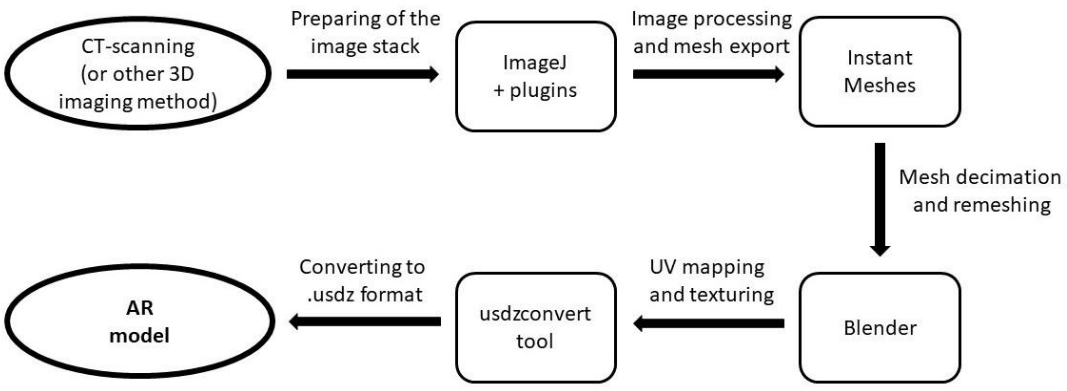

Fig. 6. Schematic pipeline of the AR model development process with the proposed software

The method for obtaining the USDZ model will be demonstrated on the basis of two samples. The first is a carbonate reservoir sample after filtration of hydrochloric acid through it with the diameter of 3 cm and height 2.5-2.7 cm (the sample presented in Fig. 1 and described in more detail in [36]). This sample is well suited for AR-visualisation of its virtual copy since it has a complex shape with depressions and an internal channel (wormhole), as well as a carbonate texture. The second is a small cylindrical carbonate sample (5 mm in diameter and 5 mm in height) with fracture inside. The main idea for it is a demonstration of colourized crack inside the hollow shape of the sample.

3.1 CT-imaging and digital analysis

The first step included μCT scanning of the sample and 3D image reconstruction. X-ray computed tomography was performed using the General Electric v|tome|x S240 micro- and nanofocus X-ray research system for computed tomography. The following analytical parameters for X-ray scanning were maintained for the first sample: current of 170 μA, voltage of 150 kV, number of projections of 1200, average of 3, timing for 1 projection of 200 ms. The obtained voxel resolution was 34.33 μm. Next parameters for the second sample were set: current of 130 μA, voltage of 100 kV, number of projections of 1200, average of 3, timing for 1 projection of 200 ms. The voxel resolution was 7.88 μm. The reconstruction for both samples was proceeded by FDK algorithm from circular cone-beam projections [37]. The output files were 16-bit image sequences in .tif format.
Digital analysis was performed in Fiji/ImageJ[3] free software [38] using "3D ImageJ Suite" [39], "MorphoLibJ" [40] and "3D Viewer" [41] plugins. For the first sample the procedure includes conversion of the μCT-image sequence from 16-bit to 8-bit, applying the Gaussian blur filter with sigma 2.00 and 3D hysteresis thresholding (with ticked "show Multi-threshold" function) of rock matrix in 3D ImageJ Suite plugin. Since the main task is to build a complex surface of a carbonate sample with large channels and pits, the internal disconnected pores can be removed. However, in this case, they were left in the model in order to see how they will be displayed and affect the final model. Segmented rock matrix was displayed as a surface in 3D Viewer plugin (Fig. 7) and then saved as .obj file. OBJ is an open-source file format which simply represents 3D geometry by saving the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make each polygon defined as a list of vertices, and texture vertices [42]. A file in .obj format can be opened by any text editor.
The second image sequence (Fig. 8) was also converted to 8-bit, filtered by Gaussian blur filter with sigma 2.00 and then all pores and the fracture (Fig. 8B) were segmented using 3D hysteresis thresholding (with ticked "show Multi-threshold" function) of 3D ImageJ Suite plugin. Then for binary images "Connected components labelling" MorphoLibJ plugin tool was used. After that, grey map was set for label image and fracture label was selected in the same plugin. Obtained image sequence of fracture was displayed as a surface in 3D Viewer plugin and saved as .obj. Next step included 3D hysteresis thresholding of rock matrix from the previously fil-

tered image sequence and its joining with segmented fracture stack in image calculator. Finally, obtained shape was also saved as .obj file using 3D Viewer plugin (Fig. 8C).
It is worth noting that the stage of digital analysis can include any action leading to segmentation or modelling of a certain volume, which needs to be visualized. This step can be proceeded in any software appropriate for working with CT models.

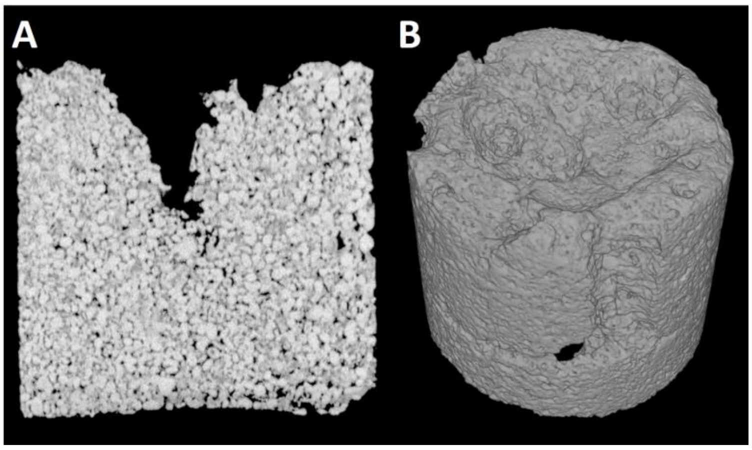

Fig. 7. The orthoslice (A) and 3D digital model (B) of the 1-st sample obtained by µCT and segmentation of rock matrix by thresholding in Fiji/ImageJ

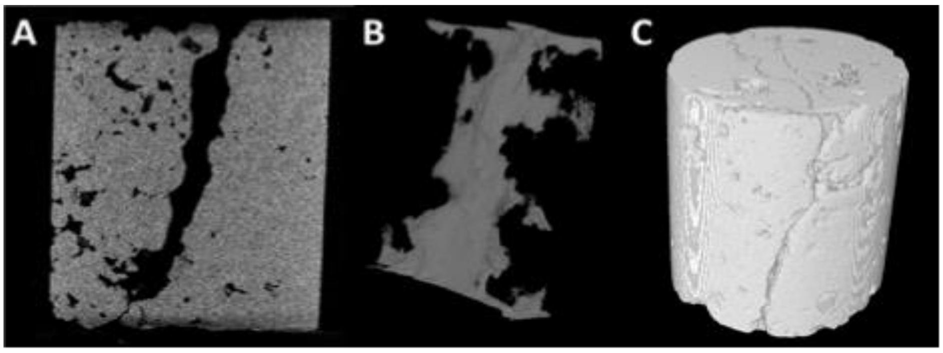

Fig. 8. The orthoslice (A) and 3D digital model of the fracture (B) and the full volume (C) of the 2-nd sample obtained by µCT and digital analysis in Fiji/ImageJ

3.2 Mesh workflow

The created surface of the first sample was presented as a complex triangle mesh and consisted of more than 4.3 million faces. The following step included decimation and remeshing, which were proceeded in Instant Meshes[4] free software [43]. Next options were chosen for the model: remesh as quads, configuration details ticked – extrinsic, target vertex count – 190.77 K. Converting from a triangular mesh to a quad mesh is recommended due to the slight reduction in mesh file size. Then the parameters of orientation field and position field were solved and new mesh was extracted and saved as new .obj file (the extension ".obj" must be added to the name of the saved file). Decreasing the number of faces from 4.3 M to 168.5 K with keeping the shape of the sample was a result of mesh transformation (Fig. 9).
A similar procedure was implemented for the second sample. The only difference was that full volume model was pre-loaded into the free software Blender[5], where the model was remeshed using the modifier with the following parameters: octree depth – 8, mode – smooth. This manipulation was managed in order to get rid of the inner surfaces of pores and cracks, which are not of particular interest in visualizing the transparent surface of the sample and only complicate the model. The number of faces was decreased from 4.7 M to 355.7 K in the result of this process. After that, full volume and fracture models were decimated by the same method in Instant Meshes. The number of faces was reduced from 355.7 K to 47.6 K (set target vertex 49.5

K) (Fig. 10 A-B) and from 640.5 K to 49.3 K (set target vertex 53.45 K) (Fig. 10 C-D) for full volume and the fracture respectively.

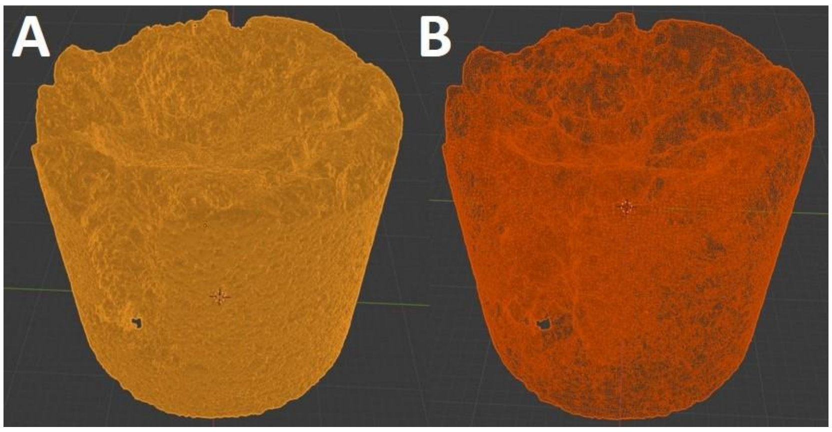

Fig. 9. The result of decimation and remeshing with decreasing of faces number from 4.3 M (A) to 168.5 K (B) and transformation from triangle to quad mesh

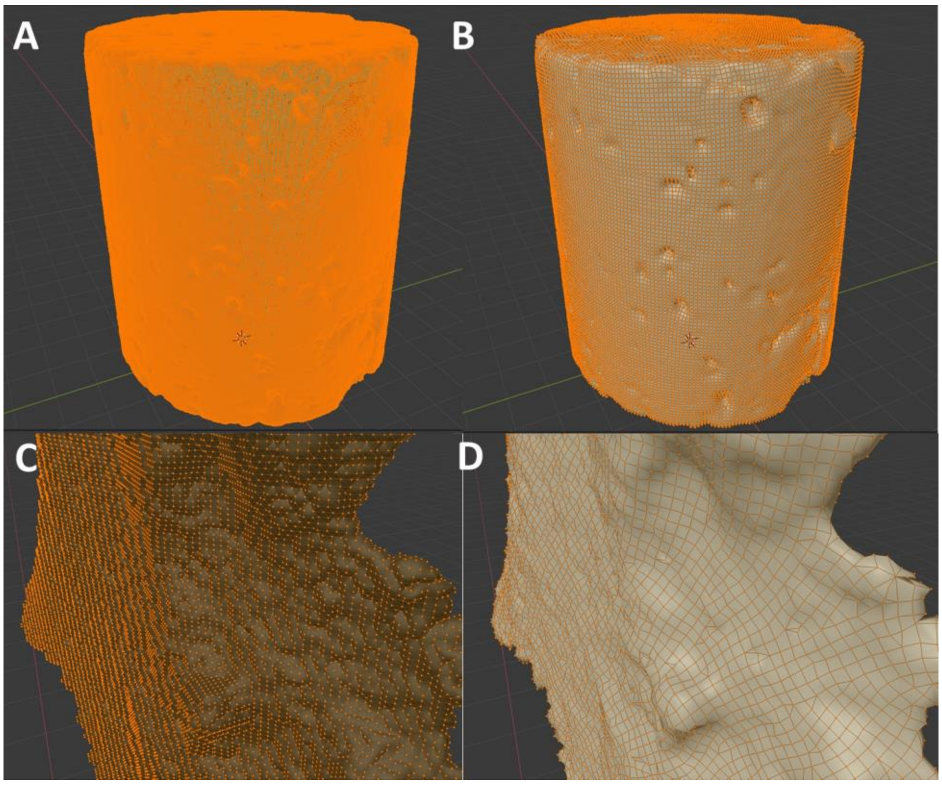

Fig. 10. The result of decimation and remeshing with decreasing of faces number from 355.7 K (A) to 47.6 K (B) faces for the full volume of sample 2 and from 640.5 K (C) to 49.3 K (D) faces for its fracture

3.3 UV mapping and texturing

UV mapping and texturing have proceeded in Blender free software. Imported as the .obj file meshes were displayed as a lying object because axes *Z* and *Y* is usually swapped in 3D mesh software. The rotation to 90° through *X*-axis was applied for the correct view. After that, the model was centred in the coordinate grid. Next step included switching to edit mode, selecting all elements and creating UV map. Since manual or automatic unwrapping is inapplicable for such high-polygonal meshes, a simple cube-projection UV mapping was used.
The resulting projection for sample 1, presented in UV edit mode, was a set of individual faces, which often overlap each other, but formed close to a square shape as a whole (Fig. 11). The

projection had been shifted and expanded to fill the 2048×2048 map as much as possible. Then the .obj file was resaved to write new information about UV coordinates.
Since the first sample has a natural lime texture and the accuracy of the three-dimensional texture of the model is not important here, a free set of PBR textures[6] for limestone was used for the model. It included 5 types of maps which were more than enough for our task: albeldo (diffuse colour), ambient occlusion, metallic, normal and roughness. Every texture map had the size of 2048×2048 pixels and perfectly fits UV map (Fig. 11). Each face of UV map was textured by a limited area of the underlying pixels.

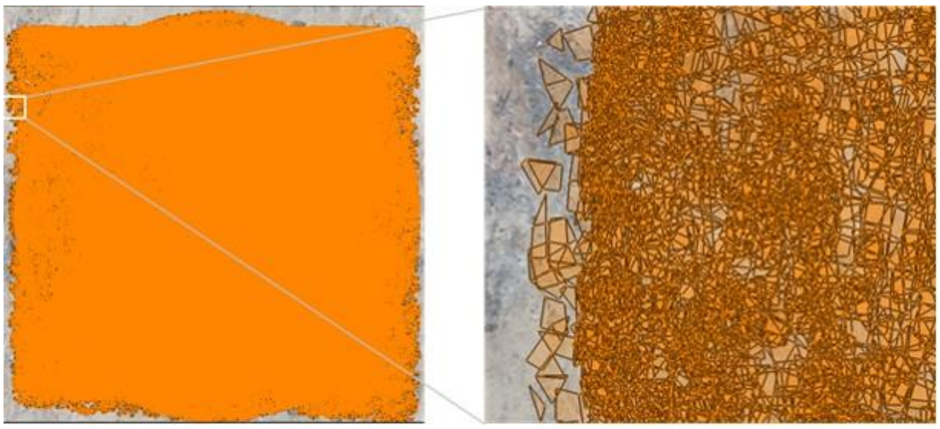
Fig. 11. The UV map for the mesh based on cube projection and underlying diffuse color texture map

The model of the second sample included two meshes, which were separately imported to the Blender software. After that, they were scaled down to 1% of the current view, because their sizes were too large for visualization. Moreover, sample surface mesh was additionally decreased to 0.998 of its scale and raised up to 0.001 relatively to *Z*-axis for the absence of intercrossing of upper and lower surfaces of two meshes (Fig. 12 B). Then UV mapping based on cubic projection has proceeded for each mesh. It was important to place obtained projections on one 2048 × 2048 UV map without crossing (Fig. 12 A). Unlike the previous one, their own PBR textures were created for this model. They included all necessary types of maps: diffusion, opacity, normal and roughness. However, adhering to the task to make the sample's surface transparent, it was important not to overload rendering with texturing. For this reason, in contrast to the fracture, the sample's surface was textured by the uniform colour on all maps (Fig. 13).

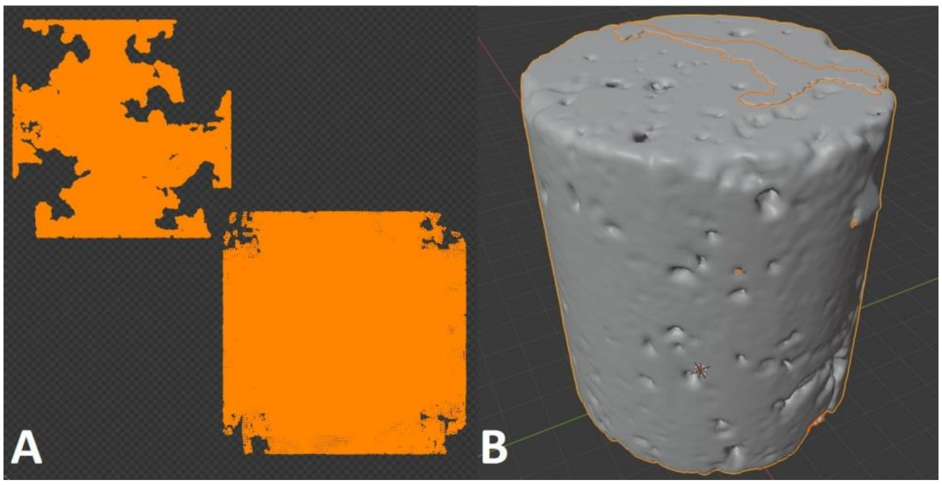

Fig. 12. The UV map of the second sample with cube projections of the fracture (top) and sample's surface (bottom) meshes (A) and their mutual arrangement in 3D visualization (B)

The diffuse colour map was made in texture mode after selecting fracture volume. Texture map 2048×2048 with grey base map (RGB – 204, 204, 204) was added to texture slot in tool settings. Then the fracture surface was filled by green colour (RGB – 3, 255, 0). Thus, green colour was automatically added to the fracture's UV map (fig. 13 A). The UV projection of the sample's surface was spatially related with uniform grey base colour, which was further corresponded to its diffuse colour. After all manipulations, diffuse map was saved as .png image.
Opacity map was created by the same method. However, due to opacity depends to one of RGB channels value, the texture of the fracture was filled by white colour which was meant totally opaque, where each of R, G and B channels were equal to 1 (255, 255, 255). The base colour was chosen as grey, where each of R, G and B channels were equal to 0.788 (201, 201, 201) and related to transparency of the sample's surface. Obtained opacity map was saved as .png image (Fig. 13 B).

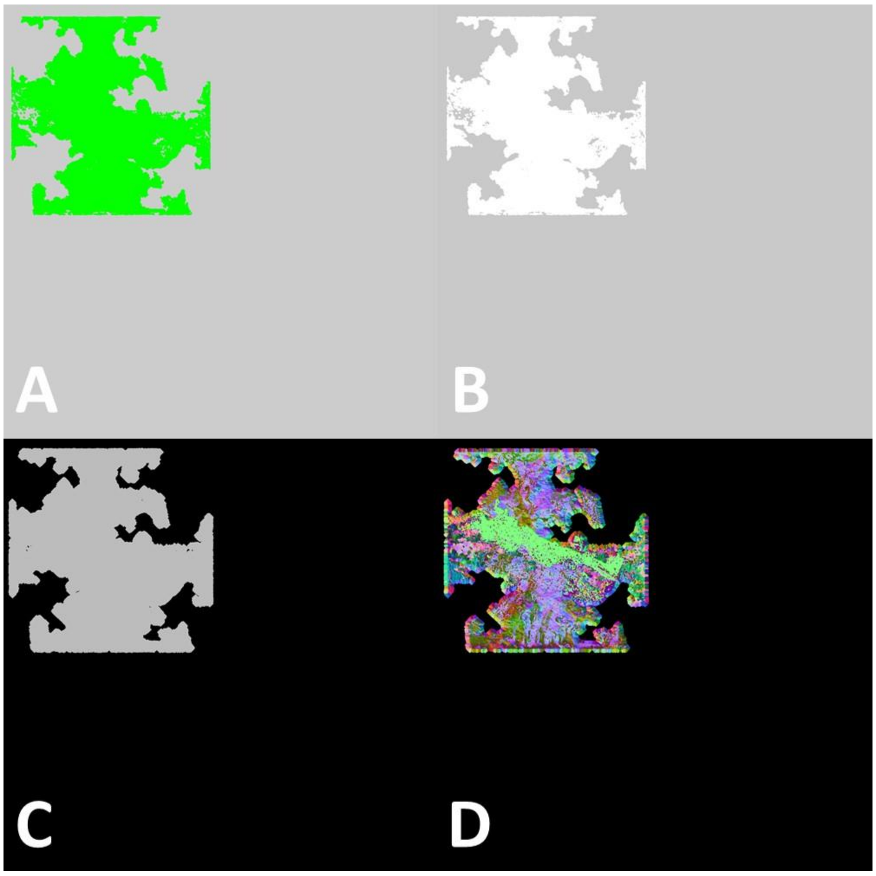

Fig. 13. Generated texture maps for the second sample: A – diffuse colour map, B – opacity map, C – roughness map, D – normal map

A similar approach was used to create the roughness map. The base map colour was set as 0 (0, 0, 0) which means that the sample's surface was not rough. The value of 0.737 (188, 188, 188) was assigned for each RGB channel of the fracture. Then the roughness map was exported as .png image (Fig. 13 C).
The application of the described approach depends on the visualization tasks and can be used for all types of USDZ maps except normal, which should be baked. The fracture was selected in object mode and new image texture for base colour was created in the surface section of the material tab. Then cycles render engine was selected in the render tab and normal bake type with object space in bake section was set. After baking, obtained normal map was saved as .png image (Fig. 13 D). In the end, two meshes were joined to one model and saved as one .obj file.

3.4 Converting to USDZ

The last step included creating USDZ model from obtained .obj files and texture maps using free usdzconvert tool[7]. It is an archive with a set of scripts, one of which 'usdzconvert' is a Python-based tool to convert from various file formats (.obj, .gltf, .fbx, abc) to .usdz. The tool is presented in the form of command line in the shell. A conversion of the model into a .usdz file was started by moving to the folder with the prepared mesh and texture files. Then the command 'run_uszdconvert' (or 'usdzconvert' on macOS) with specifying in one line the name of the mesh file (e.g. 'mesh.obj') and indicating the type of texture map and the name of the corresponding texture file in .png or .jpeg format (e.g. '-diffuseColor albeldo.png') were run. Indication of one of RGB or alpha channel after the type of map and before the name of the image file (e.g. '-opacity r opacity.png') is necessary for all maps except the diffuse colour, normal and emissive colour. In this case, the parameter values are taken from the marked channel values in the range from 0 to 1. The names of each file or the commands should be separated by space. In case of the absence of PBR maps, textures for the entire model can be set by numerically specifying the values for each texture parameters (from 0 to 1) in the USDZ convert tool. For example, the parameter '-metallic 0' means the absence of metallic surface in the whole texture of the model, '-metallic 1' indicates that the whole surface of the object is metallic. Maps that were not specified in the converter are textured based on the default settings. More information about USDZ converter capabilities can be obtained by the command 'run_usdzconvert -h'. In the result, two .usdz files for 2 models with the same name as for mesh were created.

4. Results and discussion

4.1 USDZ model overview

The obtained USDZ models were tested on iPhone 7 (iOS 13.6.1). The model can be opened in Object and AR modes (Fig. 14). In the first mode, it is possible to see all the details of the model in diffused lighting; it can be rotated and enlarged. The model in AR mode is installed on a flat well-illuminated surface and can also be rotated, enlarged and shifted along the horizontal plane. The texture in AR mode is visualized realistically in response to ambient lighting.

The USDZ model of the first sample reflects the entire complex structure of the sample surface, including large internal channels, into which one can penetrate in the AR mode. It should be noted separately that all the faces of the original mesh, including the inner surfaces of the channels, have textures.

However, some mistakes related to meshing and texturing were detected (Fig. 15). Small internal isolated pores that were not removed during digital analysis turned into numerous small fragments within the model, which only complicate the model and take up more memory (Fig. 15 A). For this reason, deleting of such internal elements is recommended, if they are not of interest for rendering. Branches of these small pores from the main large channels in the result of the described meshing procedure turned into holes on the surface of mentioned channels (Fig. 15 B). This problem can be solved by using more detailed meshing to describe such small shapes or by closing holes and smoothing them [24,25]. In addition, areas of a sharp transition in the tone of textures were identified in parts of the model with complex geometry (for example, inflexions and internal channels) (Fig. 15 C). They look like separate patches in sharp contrast to the surrounding background and are formed as a result the random arrangement of some faces on the UV map.

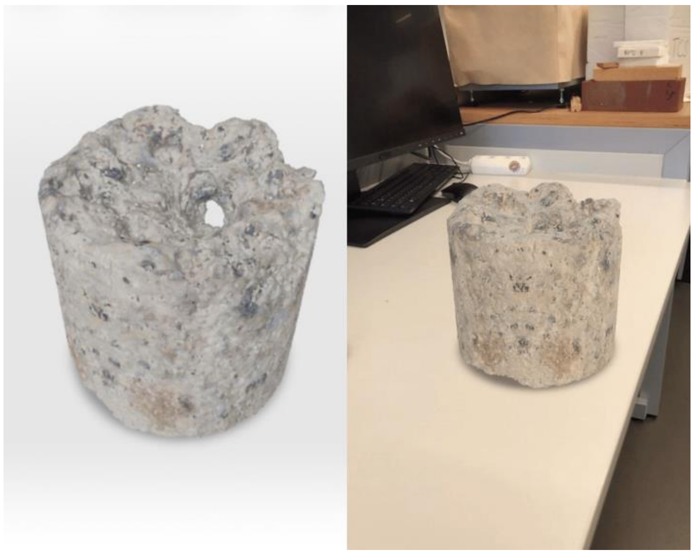
Fig. 14. The USDZ model of the first carbonate sample in Object mode (left) and AR mode (right). This USDZ model can be downloaded from https://github.com/Rail7777/USDZ-for-Scientific-Visualization/raw/main/Acidized_sample_(Fig_14).usdz

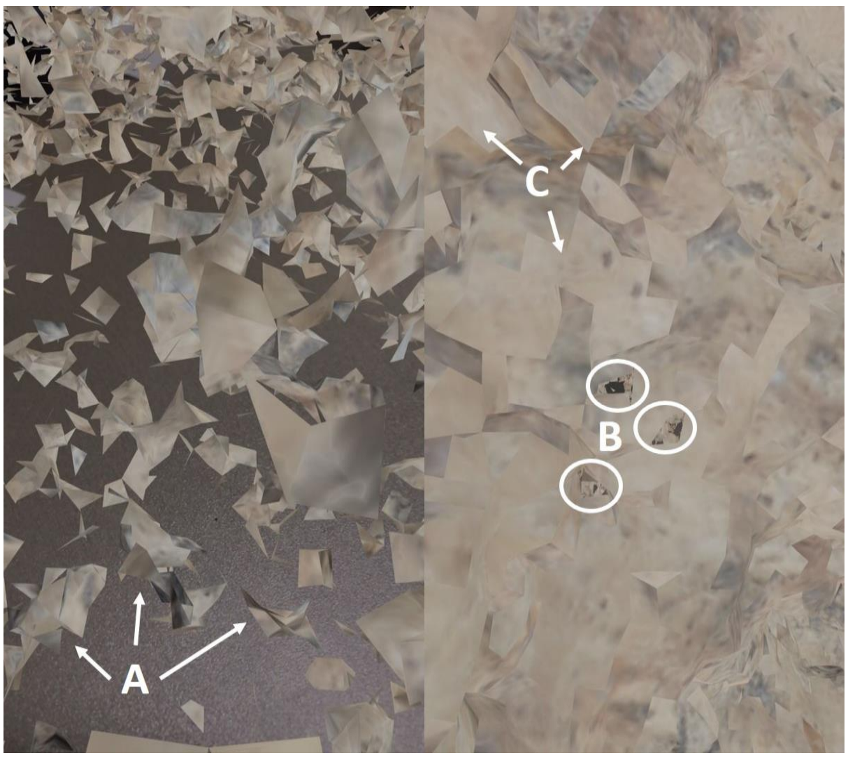

Fig. 15. The mistakes detected in the first USDZ model: A – fragments formed by mesh faces of small internal pores, B – holes in a solid mesh surface, formed in branches of thin pore channels, C – areas of abrupt tone transition of textures formed by the random arrangement of some faces on the UV map

The model of the second sample shows high transparency of the sample surface by virtue of zero values of roughness and normal map (Fig. 16). On the one hand, it has a glossy sheen indicating a high degree of light reflection. On the other hand, the main irregularities and cavities of the surface are clearly visible. The surface of the fracture demonstrates a bright matte colour with realistic contrasting shadows that accentuate its relief.

However, this model also has bugs in visualization. First of them is the disappearance of the fracture during model rotation around the *Z*-axis (Fig. 17A). It is an internal USDZ bug connected with total outer transparency of the model and can be also observed in low polygonal models. The second is the disappearance of a part of the outer fracture surface in a side view which allows seeing its inner surface. This mistake is probably connected with the complexity of the mesh.

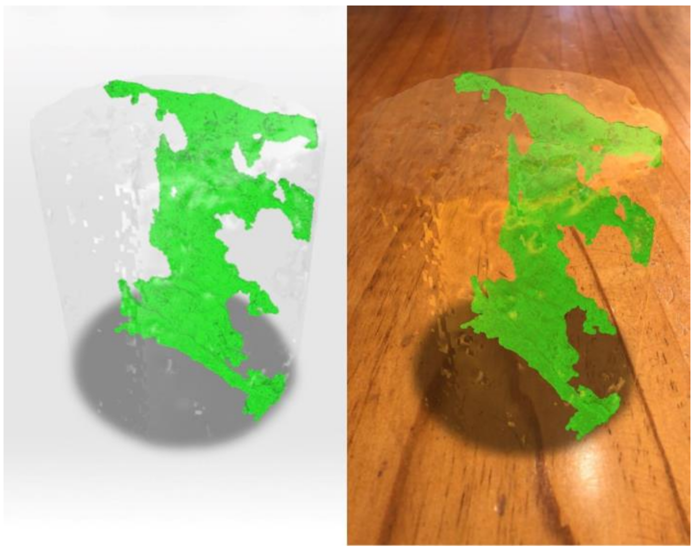
Fig. 16. The USDZ model of the second sample with the fracture in Object mode (left) and AR mode (right). This USDZ model can be downloaded from https://github.com/Rail7777/USDZ-for-Scientific-Visualization/raw/main/Cracked_sample_(Fig_16).usdz

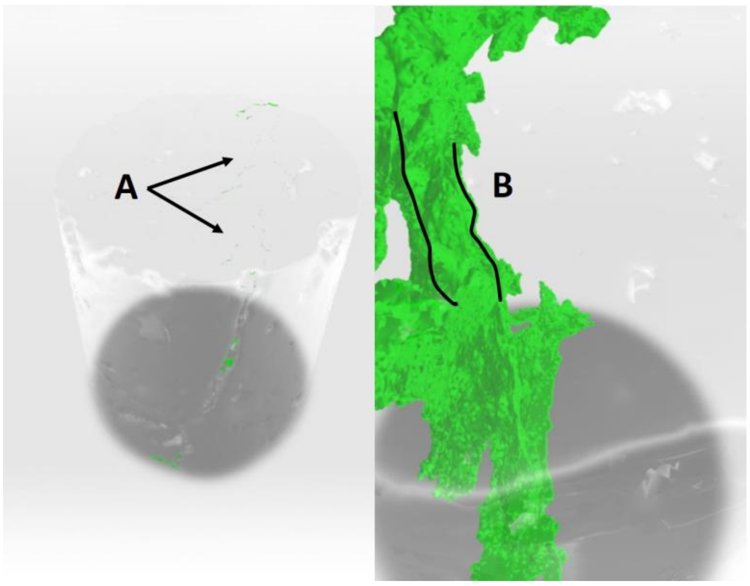

Fig. 17. The mistakes detected in the second USDZ model: A – the disappearance of the fracture in a 180° sector during model rotation, B – the disappearance of the outer surface of the crack and the display of its inner surface (the area between the black lines) when viewed from the side

4.2 USDZ limitations

Since USDZ was originally intended for AR visualization of low-poly models, understanding the limits of its applicability for working with high-poly models of digital rock analysis is extremely important. The latter can have a much more complex surface geometry that requires a huge number of faces (from hundreds of thousands to millions) to describe the shape. The tests carried out show that the maximum number of faces for a successful launch of the AR model is about 800 K, however, the model should completely lack textures (except for those that are set by default). In the case of adding complex textures, the number of faces should be significantly reduced to 150 – 250 K. Ultimately, for the successful work of the AR, the size of

the USDZ model should not greatly exceed 30 MB. The use of complex textures in the model, especially the transparency, further reduces the size of the .usdz file (maximum about 20 MB) necessary for its successful work.
Another feature of digital rock analysis models is a significant size distribution of their parts. Along with the large fragments, much smaller objects may be present. All this requires either their enlargement or smoothing, or detailing the mesh in this area. The latter, however, can lead to errors in rendering the USDZ model due to the irregularity of the mesh.
The next challenge when working with high-poly models is UV mapping and subsequent texturing. In this case, due to the huge number of faces, manual UV mapping by separate blocks is practically impossible. Most automatic UV mapping algorithms are also unable to do this . The results they produce are no better than simple geometrical projection methods (e.g. cube, cylinder, and sphere) that are much more time-efficient.
The resulting UV maps are also related to the quality of texturing. The scatter of individual faces on the UV map, which are adjacent to the 3D model and texturing by overlaying an image, leads to the appearance of contrasting spots in the visualization. Manual texturing methods used for low-poly models are labour-intensive and ineffective since it is very difficult to paint geometrically complex interior areas, such as pore channels. Nevertheless, the texturing method demonstrated for the first model is sufficient for rendering at this stage. In addition, based on digital rock analysis practice, it can be assumed that most segmented volumes will have a single colour as shown in the case of the crack. The latter is easily accomplished by filling the entire diffuse map with one colour.

4.3 Advanced prospects for USDZ based visualization

As it was demonstrated in this work, the USDZ models can be easily obtained using free software, successfully applied for AR-visualization and transmission of 3D models. They have realistic rendering characteristics through the use of PBR-texturing and allow working with high-poly models. Due to its recent appearance and an initial inadaptability for application in the field of scientific imaging, certain bugs are noted in the USDZ models that can be resolved in the future.
The described methodology can be used to develop AR models not only based on X-ray CT, but also magnetic resonance imaging (MRI), confocal microscopy, FIB-SEM and other types of 3D imaging. That means the possibility of AR models application in many different fields of science, including medicine, biology, palaeontology, archaeology, etc. Potentially, USDZ models can be developed for AR visualization of any 3D scientific data. However, the process of USDZ model developing is not easy for an ordinary researcher. Further development of this direction is possible on the basis of creating a unified open-source software for simple semi-automatic / automatic decimation and remeshing, PBR-texturing and conversion to a .usdz file.
One of the main concepts of USDZ is the ability to create a scene including several models that can interact with each other and with the user. This allows creating new complex models based on combining several USDZ and opens up new possibilities for advanced interactive visualization. It allows combining several USDZ models into a single scene while setting its own properties for each model (for example, position, sizes, material properties, etc.) and reactions, which include triggers and actions. This approach is currently implemented in a proprietary application, so this issue is beyond the scope of this article. However, the development of such open-source software could be beneficial for the scientific community.
Another interesting possibility of using the described models is 3D animation. USDZ can contain simple animation of an object in the form of moving, rotating, and resizing [22]. The current USDZ converter is also capable of converting simple animation from .abc or .gltf formats. However, there is currently no possibility for visualization of much complex animation, which can include the simultaneous change in volume and spatial position (e.g. the fluid flow inside porous space). Nevertheless, an approach of 3D animation in AR could be promising for volumetric demonstration of many physical processes.

5. Conclusion

Thus, the possibility of using the USDZ format for AR-visualization of geometrically complex models obtained on the basis of digital analysis of rocks was considered. A brief information on the structure of the .usdz file, a methodology for developing an AR-model based on free software, limitations for models of this format and advanced prospects for USDZ based visualization were presented. The main advantages of this format are realistic textures based on Physically Based Rendering, a huge number of mobile devices that can support this format, and ease of transfer between users. The results can be used for the building of AR models of any 3D scientific output even with complex geometry for quick access and exchange the visual information and native demonstration in conferences, exhibitions and to support scientific publications or educational process.

Acknowledgements

This work was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities (Project № 0671-2020-0048 of State Assignment № 075-00216-20-05 of 04.06.2020 (Part II Section 1)).
The author is grateful to the “Algarysh” internship program, International Geothermal Center, Bochum University of Applied Sciences and personally thanks Prof. Erik Saenger and Prof. Danis Nurgaliev for supporting the work.

Computer Code Availability

1. Name of code – Fiji/ImageJ
Developer(s) and contact address – Fiji contributors (contacts, tel. numbers and e-mails available on https://imagej.net/Contributors)
Year first available – 2007
Hardware required – PC on Windows XP, Vista, 7, 8 and 10; Mac OS X 10.8 "Mountain Lion" or later; Linux on amd64 and x86 architectures.
Software required – no
Program language – Java
Program size – 320 Mb (Windows x 64 version)
Details on how to access the source code – download from https://imagej.net/Fiji/Downloads
License – GPL

ImageJ plugins

1.1. Name of code - 3D ImageJ Suite
Developer(s) and contact address –Thomas Boudier and J. Ollion; Academia Sinica, Taipei, Taiwan.
Telephone number and e-mail – n/a; thomas.boudier@upmc.fr
Year first available - 2007
Hardware required – PC (64 bit) Windows, MacOS, Linux.
Software required - Fiji/ImageJ
Program language - Java
Program size – 854 KB
Details on how to access the source code – download from https://imagej.nih.gov/ij/plugins/3d-viewer/
License – GPL

1.2. Name of code - MorphoLibJ
Developer(s) and contact address – David Legland, Ignacio Arganda-Carreras; French National Institute for Agriculture, Food, and Environment (INRAE), Paris, France.
Telephone number and e-mail – +33 (0) 240675243; david.legland@inrae.fr

Year first available - 2014
Hardware required – PC (64-bit) Windows, MacOS, Linux.
Software required - Fiji/ImageJ
Program language - Java
Program size – 845 KB
Details on how to access the source code – download from https://github.com/ijpb/MorphoLibJ/
License – GPL

1.3. Name of code - 3D Viewer
Developer(s) and contact address – Benjamin Schmid, Johannes Schindelin; MPI-CBG Dresden, Germany.
Telephone number and e-mail – +33 (0) 240675243; david.legland@inrae.fr
Year first available - 2017
Hardware required – PC (64-bit) Windows, MacOS, Linux.
Software required - Fiji/ImageJ
Program language - Java
Program size – 491 KB
Details on how to access the source code – download from https://github.com/mcib3d/mcib3d-core/
License – GPL (V3)

2. Name of code – Instant Meshes
Developer(s) and contact address – Wenzel Jakob, Marco Tarini, Daniele Panozzo, Olga Sorkine-Hornung; BC 345, Station 14, CH-1015 Lausanne, Switzerland
Telephone number and e-mail – +41 21 69 31329; wenzel.jacob@epfl.ch
Year first available - 2015
Hardware required – PC (Intel, 64 bit) Windows, MacOS, Linux.
Software required - Compiling from scratch requires CMake and a recent version of XCode on Mac, Visual Studio 2015 on Windows, and GCC on Linux.
Program language - Python
Program size – 1.33 Mb (Windows x 64 version)
Details on how to access the source code – download from https://github.com/wjakob/instant-meshes
License – BSD

3. Name of code – Blender
Developer(s) and contact address – Blender Foundation; Stichting Blender Foundation
Buikslotermeerplein 161, 1025 ET Amsterdam, the Netherlands
Telephone number and e-mail – n/a; foundation@blender.org
Year first available - 1994
Hardware required – CPU x32- x64, RAM 4-16GB, OpenGL 2.1-3.2 graphics card with 1-4GB RAM.
Software required - no
Program language - Python
Program size – 161 Mb (Windows x 64 version)
Details on how to access the source code – download from https://www.blender.org/
License – GPL (V3)

4. Name of code – usdzconvert
Developer(s) and contact address – Apple Inc.; Apple, One Apple Park Way, Cupertino, CA 95014
Telephone number and e-mail – +1 (408) 996–10–10; n/a

Year first available - 2019
Hardware required – PC on Windows or Unix platform.
Software required - no
Program language - Python
Program size – 37.6 Mb (Windows x 64 version)
Details on how to access the source code – download from https://github.com/tappi287/usdzconvert_windows and follow the instructions
License – MIT

References

1. Leal Filho W. Augmented Reality // Encyclopedia of Sustainability in Higher Education / ed. Leal Filho W. Cham: Springer International Publishing, 2019. P. 76–76.
2. Comeau C.P., Bryan J.S. Headsight television system provides remote surveillance // Electronics. 1961. Vol. 34. P. 86–90.
3. Caudell T.P., Mizell D.W. Augmented reality: an application of heads-up display technology to manual manufacturing processes // Proc. Twenty-Fifth Hawaii Int. Conf. Syst. Sci. 1992. Vol. ii. P. 659–669 vol.2.
4. Rosenberg L.B. The use of virtual fxtures as perceptual overlays to enhance operator performance in remote environments (Technical Report AL-TR-0089), USAF Armstrong Laboratory, Wright-Patterson AFB OH. 1992.
5. Pierdicca R. et al. Smart maintenance of riverbanks using a standard data layer and Augmented Reality // Comput. Geosci. 2016.
6. Lee S., Suh J., Park H.D. BoreholeAR: A mobile tablet application for effective borehole database visualization using an augmented reality technology // Comput. Geosci. 2015.
7. Westhead R.K. et al. Mapping the geological space beneath your: Feet the journey from 2D paper to 3D digital spatial data // International Conference on Information Society, i-Society 2012. 2012.
8. Mathiesen D. et al. Geological visualisation with augmented reality // Proceedings of the 2012 15th International Conference on Network-Based Information Systems, NBIS 2012. 2012.
9. Boghosian A.L. et al. Inside the ice shelf: using augmented reality to visualise 3D lidar and radar data of Antarctica // Photogramm. Rec. 2019.
10. Engelke U. et al. HypAR: Situated mineralogy exploration in augmented reality // Proceedings - VRCAI 2019: 17th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry. 2019.
11. Gazcón N.F. et al. Fieldwork in Geosciences assisted by ARGeo: A mobile Augmented Reality system // Comput. Geosci. 2018.
12. Andrä H. et al. Digital rock physics benchmarks—Part I: Imaging and segmentation // Comput. Geosci. 2013. Vol. 50. P. 25–32.
13. Konathala S. All about Apple's new USDZ File Format — Simplified [Electronic resource]. 2018. URL: https://medium.com/techinpieces/all-about-apples-new-usdz-file-format-simplified-12dff29f3fc0 (accessed: 17.08.2020).
14. Modified Apache 2.0 License [Electronic resource]. 2016. URL: https://github.com/PixarAnimationStudios/USD/blob/release/LICENSE.txt (accessed: 17.08.2020).
15. Introduction to USD [Electronic resource]. URL: https://graphics.pixar.com/usd/docs/index.html (accessed: 17.08.2020).
16. Apple Reports First Quarter Results [Electronic resource]. 2018. URL: https://www.apple.com/newsroom/2018/02/apple-reports-first-quarter-results/ (accessed: 17.08.2020).
17. glTF Overview [Electronic resource]. 2013. URL: https://www.khronos.org/gltf/ (accessed: 19.08.2020).

18. ARCore supported devices [Electronic resource]. 2020. URL: https://developers.google.com/ar/discover/supported-devices (accessed: 19.08.2020).
19. USD from glTF [Electronic resource]. 2020. URL: https://github.com/google/usd_from_gltf (accessed: 19.08.2020).
20. Augmented reality with model-viewer [Electronic resource]. 2020. URL: https://developers.google.com/web/updates/2019/05/model-viewer-ar (accessed: 19.08.2020).
21. 3D and AR Early Adopters Program [Electronic resource]. 2020. URL: https://developers.google.com/search/docs/guides/3d-ar (accessed: 19.08.2020).
22. Universal Scene Description [Electronic resource]. 2016. URL: https://github.com/PixarAnimationStudios/USD (accessed: 17.08.2020).
23. Usdz File Format Specification [Electronic resource]. 2016. URL: https://graphics.pixar.com/usd/docs/Usdz-File-Format-Specification.html#UsdzFileFormatSpecification-Purpose (accessed: 21.08.2020).
24. Schneider P.J., Eberly D.H. Geometric Tools for Computer Graphics // Geometric Tools for Computer Graphics. Elsevier, 2003. 1056 p.
25. Hansen C.D., Johnson C.R. Visualization Handbook // Visualization Handbook. 2005. 984 p.
26. Sherman W.R., Craig A.B. Understanding Virtual Reality: Interface, Application, and Design // Understanding Virtual Reality: Interface, Application, and Design. 2nd-nd ed. 2018. 938 p.
27. de Floriani, Leila, Spagnuolo M. Shape Analysis and Structuring / ed. De Floriani L., Spagnuolo M. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. 296 p.
28. Watkins A. Creating games with Unity and Maya: how to develop fun and marketable 3D games // Choice Reviews Online. 2011. 528 p.
29. Ebert D. Texturing and modeling: a procedural approach. Academic Press, 1994. 332 p.
30. Eck D. Introduction to Computer Graphics. 2015. 440 p.
31. Buss S.R. 3-D Computer Graphics. Cambridge: Cambridge University Press, 2003. 396 p.
32. McDermott W. THE PBR GUIDE. 3rd-rd ed. Allegorithmic, 2018. 104 p.
33. Pharr M., Humphreys G. Physically Based Rendering: From Theory to Implementation. Elsevier, 2004. 1056 p.
34. Principled BSDF [Electronic resource]. 2020. URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html (accessed: 28.08.2020).
35. Greenberg D.P. et al. A framework for realistic image synthesis // Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97. New York, New York, USA: ACM Press, 1997. P. 477–494.
36. Kadyrov R.I. et al. Structural Transformation of the Void-Pore Space of a Lime Reservoir During HCl Treatment // Chem. Technol. Fuels Oils. 2018. Vol. 54, № 3. P. 307–318.
37. Feldkamp L.A., Davis L.C., Kress J.W. Practical cone-beam algorithm // J. Opt. Soc. Am. A. 1984.
38. Perez J.M.M., Pascau J. Image Processing with ImageJ. Packt Publishing, 2013. 140 p.
39. Ollion J. et al. TANGO: A generic tool for high-throughput 3D image analysis for studying nuclear organization // Bioinformatics. 2013.
40. Legland D., Arganda-Carreras I., Andrey P. MorphoLibJ: Integrated library and plugins for mathematical morphology with ImageJ // Bioinformatics. 2016.
41. Schmid B. et al. A high-level 3D visualization API for Java and ImageJ // BMC Bioinformatics. 2010.
42. Wavefront OBJ File Format Summary [Electronic resource]. 2005. URL: https://www.fileformat.info/format/wavefrontobj/egff.htm (accessed: 31.08.2020).
43. Jakob W. et al. Instant field-aligned meshes // ACM Trans. Graph. 2015.

[1] https://unity.com
[2] https://developer.vuforia.com/
[3] https://imagej.net/Fiji/Downloads
[4] https://github.com/wjakob/instant-meshes
[5] https://www.blender.org/
[6] https://freepbr.com/materials/markeed-limestone-rock/
[7] https://github.com/tappi287/usdzconvert_windows