Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2025, 19(2), 283–295
https://doi.org/10.12913/22998624/195718
ISSN 2299-8624, License CC-BY 4.0

Received: 2024.09.05
Accepted: 2024.12.15
Published: 2025.01.01

Analysis of the possibility of hiding decomposed information in the virtual reality environment

Tomasz Szymczyk[1*], Piotr Czajka[2]

[1] Faculty of Electrical Engineering and Computer Science, Lublin University of Technology, Nadbystrzycka 38A, 20-618 Lublin, Poland
* Corresponding author's e-mail: t.szymczyk@pollub.pl

ABSTRACT

Information is a fundamental resource, generated by mankind, it is through it that one can reduce or eliminate the effects of a cataclysm, make or lose a fortune, or even shorten a war. The transmission of hidden information in the form of cryptographic techniques has been known to mankind since the dawn of time. Initially simple, basic encryption has evolved into sophisticated steganographic hiding of information gaining more and more resistance to breakage with the development of technology. It is very difficult to develop a new, effective encryption algorithm. Therefore, it becomes very important to skillfully use already existing cryptographic algorithms and develop a new general encryption algorithm. The idea of where to hide the data also becomes important. The article proposes an author's novel system for decomposing the hidden information. Using steganography, a fragmented password is hidden in 3D objects in the Virtual Reality application world. The data is hidden directly in the VR world (a multiplayer simple game) with the caveat, however, that not everyone knows where to look for it. The time in which the information is sought is also important. Clues as to how, when and where to find the data are encoded in 3D objects. The possibility of hiding information in the metadata, in the texture, in the coordinates of the vertices of the objects and also in the UV map has been thoroughly investigated.

Keywords: virtual reality, information decomposition, information hiding, selective access, steganography, unity, blender.

INTRODUCTION

Steganography is a technique of hiding information in such a way that its presence is not visible to unauthorised persons. Unlike cryptography, which protects the content of a message from unauthorised reading, steganography hides the very fact that the message exists. The purpose of steganography is to ensure that hidden data are not easily detectable or suspicious.

Steganography, often unlike cryptography, allows third parties to interact with a file. Widespread access to files significantly reduces the level of vigilance of people looking for information. Additionally, the data are stored secretly.

Hiding information in three-dimensional objects is a combination of several fields, such as computer graphics, cryptography, steganography and other fields of broadly understood computer science. This method faces many challenges that must ensure resistance to many computer operations. First of all, steganography is intended not to visibly deteriorate the image quality [1].

There are many articles in the literature on information concealment. An interesting approach is presented in article [2]. It concerns the hiding of information in motion, the skeleton of an animated character. Many of the articles deal with hiding information directly in the texture Many of the articles treat hiding information directly in graphics [3–8] in texture [9–12]. It is very common to hide information in vertices [13–15]. The topic of finding and hiding information in the UV map was also touched upon, as in the articles [16–19].

Hiding information in an object of a Blender program graphic file (*.blend) offers surprisingly many possibilities. This file consists of both vectors constituting the essence of the shape, the presented 3D object, as well as the texture with which the object is covered. The way in which texture is applied to the 3D object is saved in a UV map. MapUV contains information about which texture fragments should cover individual surfaces and how they should fit to the vertices of the 3D solid. The UV map is an additional place to hide information.

However, there are few articles combining the above methods with hiding in the virtual reality world. In the article [20], researchers address the topic of adding information to 3D objects. The objects are of various origins: computer-created, acquired with a 3D scanner, or created with photogrammetry. Other researchers have proposed the possibility of hiding keys in GIS video images [21]. In this article, the authors present the results of a system combining information hiding in 3D graphics (Blender file) and virtual reality. This is undoubtedly a unique and innovative approach to the problem.

MATERIALS AND METHODS

Virtual reality

Virtual reality (VR) is a technology that allows the user to enter a virtually created, often three-dimensional, interactive world. Thanks to the phenomenon of immersion, the user can "lose himself" and forget about the real world. In the VR world, often similar to a computer game, he can experience realistic interactions with the environment in real time. With special VR goggles, headsets, motion controllers and other advanced devices, the user feels the illusion of being in a simulated environment. The virtual environment can replicate both real and completely fictional places.

The virtual world studied in the corresponding article (shown in Fig. 1 and Fig. 2) for a multiplayer game. The player can explore the world, race cars, etc. In general, this is a task about responding to the invoice's remarks that there may be secret information hidden somewhere. In such a place, a computer game can easily be tempted to hide information. The information can be a Uniform Resource Locator (URL) to a file, or plain, open, unencrypted text that will appear, for example, on the wall of a house, or after performing some action. The only problem is how to make it so that the hidden information is not revealed to every player. Confidential information is only revealed after meeting a number of conditions described later in the article.

Possibilities of hiding information in virtual reality

Virtual reality is a very suitable field for hiding information. Hiding information in a 3D object on a popular server can be very interesting. The authors propose separating confidential information and hiding it in many places. Moreover, they hide instructions on how to properly assemble and decrypt data. What is new is that some information is hidden in a publicly available virtual reality game server.

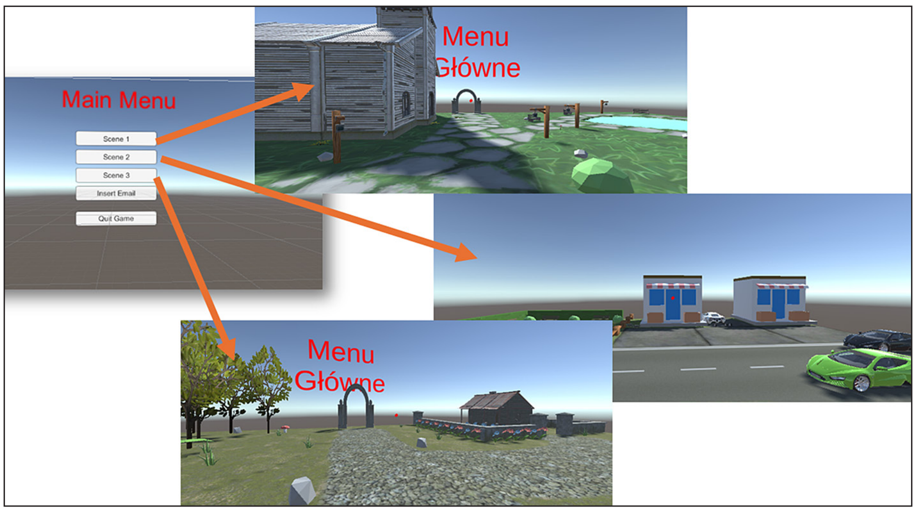


Figure 1. View from a generally available game

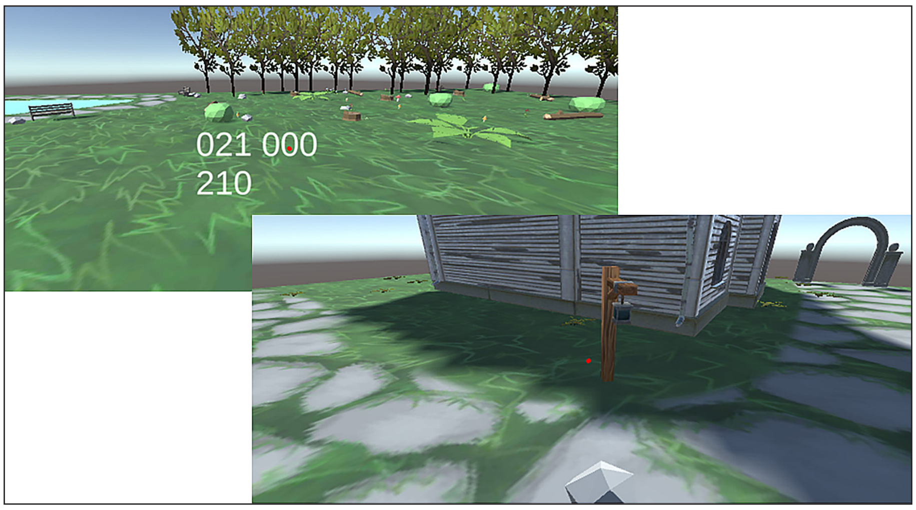


Figure 2. View from the game. The situation of finding the hidden information "021 000 210". Below the discovery of another artifact, by an unauthorized user, which did not cause

Since the graphic elements for the VR world are often modeled 3D solids it was natural to use freeware- Blender.

A Blender file is used as the main carrier of hidden, partially encrypted data. Dividing the password into parts and arranging it in different elements is just one of the additional difficulties. Another is the fact of partial information encryption. A fragment of information is provided in classified open text. The rest of it is encrypted. When analysing complex information constructed in this way, it is very easy to miss the encrypted part and confuse it with the native part of the file. This is a distraction for the decryptor. The explicit part is a logical value in itself and can be misinterpreted as all of the useful information. The Blender file is a binary file. However, using Blender software, the user has access to its integral parts: 3D solid, the texture, the way it is laid out (UV map) and also the metadata. Using the Python programming language, the authors wrote scripts to modify all these parts.

Analyzing the vertices of the spatial solid, one can quickly come to the conclusion that this is the perfect place to hide all the information. But this is not necessarily so, maybe only part of it. Another component of the Blender file is the texture. Colourful graphics is a perfect place to hide information (typical steganography). The texture must be properly cut and projected onto a 3D object. The so-called UV map in shifts can also hide a lot of information. The simplest, and at the same time equally interesting place where information can be hidden are metadata. They are the easiest to spot, so they contain information "with a short validity period" e.g. "... tomorrow between 10:00 and 10:10 the server will present hidden data". Deciphering, obtaining this information after 10:10 is no longer useful. The information from the server can no longer be read – it has been permanently deleted. An illustrative drawing of dividing a password and hiding various types of information is presented in Figure 3.

Hiding information in metadata

It is often said that the darkest place is under the candlestick. This means that information can also be hidden literally in front of the onlookers. One such audacious way is to hide fragments of information in the file's metadata. Metadata are as a rule generally available. However, few people consider that information can be hidden in such a publicly accessible place. The authors examined the possibility of hiding sensitive data in the file's metadata. In view of the fact that this space is practically visible, it is worth considering what information can be hidden in it, especially if their relevance quickly expires, for example the time at which you should log in to a specific server. Discovering such data after their expiry period no longer serves any effect: they are invalid and useless (Figure 4).

The article tested a variant of hiding split data. The beginning may indicate the IP address of the server as well as the number of the level in a publicly available global game that should be played to find additional information. The date

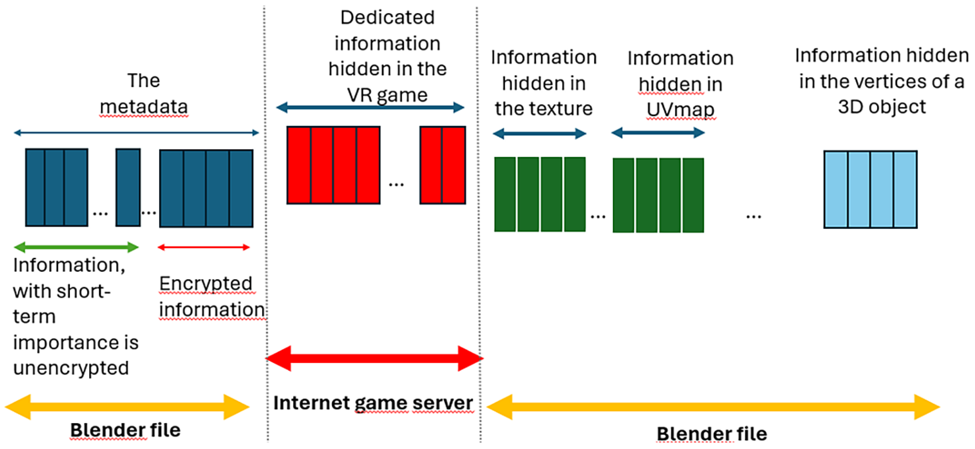
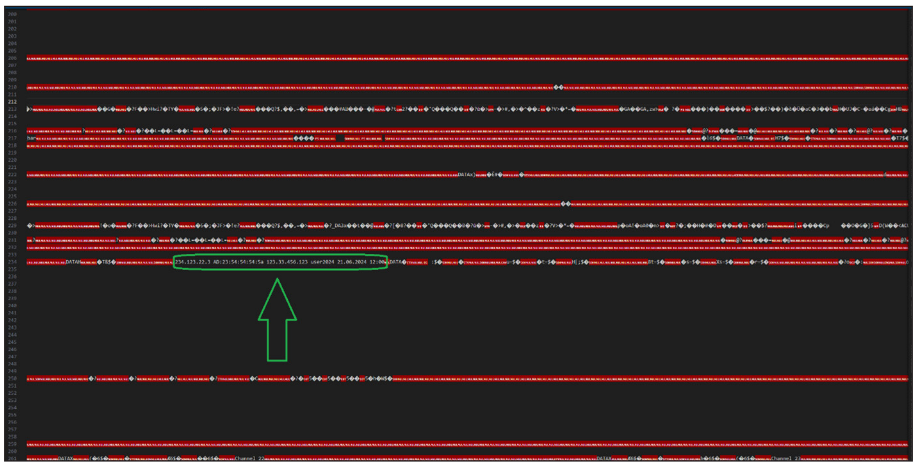

The metadata
Information, with short-term importance is unencrypted
Encrypted information
Blender file
Dedicated information hidden in the VR game
Internet game server
Information hidden in the texture
Information hidden in UVmap
Information hidden in the vertices of a 3D object
Blender file

Figure 3. Conceptual author's separation of the password. In many 3D object elements and in the virtual game world



```
234.123.22.3 AD:23:54:54:54:5a 123.33.456.123 user2024 21.06.2024 12:00
```

Figure 4. Hiding data in the file's metadata. The green arrow marks the area for entering data saved in open text

may also be provided in the open text. As well as information, for example, in what time frame one can find the next information. The huge advantage of not hiding such data is the fact that outsiders will not understand their content and will not know what to do with the IP address. Of course, everyone will immediately track down the server and can only guess that there is something in it.

Immediately after the open text in the metadata, an encrypted text containing further information appears. It is encrypted with a strong encryption algorithm such as Advanced Encryption Standard (AES) or Algorytm Rivesta-Shamira-Adlemana (RSA).

Hiding information in the public virtual reality server

The possibility of creating any server (Unity 3D, MySQL database server) with any game has enormous possibilities of transmitting hidden information [22]. By creating a server, a pool of so-called special users can be isolated. They will not be ordinary players, but people looking for further information. This time it is information hidden under the game's graphic elements. It is possible to verify the IP address from which such a user connects, as well as the MAC address of the device that connects to the game server. Hypothetically, let us imagine a situation where the user connects from the old IP address, known at the time of server creation. It is possible for the game to respond differently for this particular user. A virtual reality server containing several boards was created. At this point, the user's story is irrelevant. The user finds out through another channel when a password is going to be available for him and where it can be found. For example, it will be a time interval from one day to another. And only at even minutes. At a specified location, under a particular graphic element, there will be a text file, a fragment of text. This text can be, for example, a password to further encoded information in the Blender file metadata. The virtual reality game server is a very well camouflaged box for transmitting further partial information. Information where the password is located is provided directly to the interested person, for example by e-mail or in the form of a telephone conversation. Due to heavy restrictions and control of game users, it may also be a permanent place on the game board, and information will be hidden for a long period of time. It can be compared to the digital equivalent of a "transfer box".

The original game server created is a typical adventure game where the user has to complete certain missions. From the point of view of hiding data, this is completely irrelevant, because the initiated user is not a traditional player. Completely ignoring all plot threads and the purpose of the game, he focuses on finding hidden information.

In the metadata from the *.blend file, provided in plain text, one can read e.g. rules for access to the server, especially to hidden information. For example: the time period in which information will be waiting for the player as well as additional settings. An additional setting may be the hourly minute interval in which the box will be open or, for example, INFORMATION about the even or odd minutes in which the "box" can be accessed. One can also create a network of servers and then provide the IP address of the specific server where the hidden information is located in the metadata. A combination of all the above criteria is also possible.

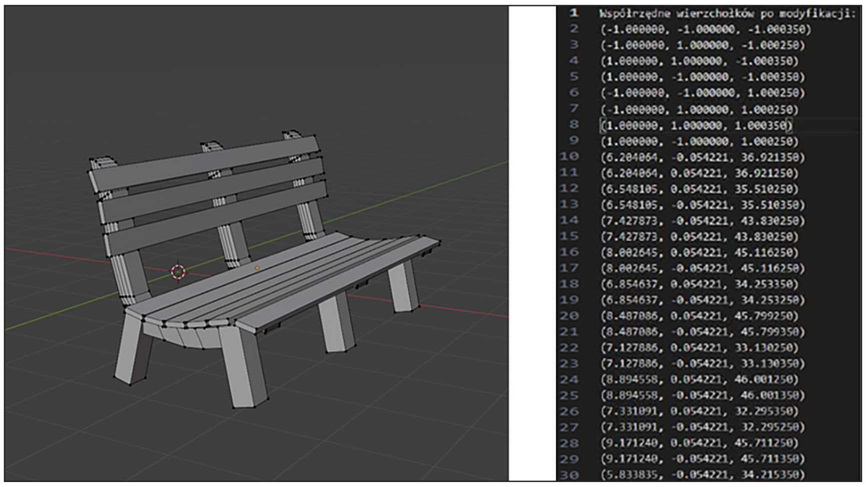
Hiding information in vertex coordinates

Hiding information involves writing it in fractional parts of the point coordinates. Using the properties of Blender, specifically its way of displaying 3D elements in space, it is very difficult to notice changes optically. The position of each vertex is approximated to the nearest mesh value with a precision of one pixel. When enlarging the image, Blender uses the remaining parts of the coordinates and here it would be possible to notice some artefact, inequality or potential change. However, there is a problem of scale and size in general. With graphical zoom and magnification, all shapes and surfaces grow significantly, so again, it is difficult to notice one-pixel deviations. By focusing on the main vertices of a given figure and not adding your own unnecessary ones, you can hide information very well. It is geometrically and optically invisible, but exists as a record in the file, in fractional parts of the coordinates of individual points (vertices) (Figures 5, 6).

Vertex topology was used to hide information. When implementing this scenario, the most important parameter to determine was the vertex offset value to avoid easy detection of information while still ensuring the ability to decode the information. Additionally, the information storage capacity is limited by the number of vertices in the model, because one vertex is responsible for storing three bits of information.

Hiding information in the texture

Data hiding and encoding was done by changing texture properties using the least significant



```
Współrzędne wierzchołków po modyfikacji:
(-1.000000, -1.000000, -1.000350)
(-1.000000, 1.000000, -1.000250)
(1.000000, 1.000000, -1.000350)
(1.000000, -1.000000, -1.000350)
(-1.000000, -1.000000, 1.000250)
(-1.000000, 1.000000, 1.000250)
(1.000000, 1.000000, 1.000350)
(1.000000, -1.000000, 1.000250)
(6.204064, -0.054221, 36.921350)
(6.204064, 0.054221, 36.921250)
(6.548105, 0.054221, 35.510250)
(6.548105, -0.054221, 35.510350)
(7.427873, -0.054221, 43.830250)
(7.427873, 0.054221, 43.830250)
(8.002645, 0.054221, 45.116250)
(8.002645, -0.054221, 45.116250)
(6.854637, 0.054221, 34.253350)
(6.854637, -0.054221, 34.253250)
(8.487086, 0.054221, 45.799250)
(8.487086, -0.054221, 45.799350)
(7.127886, 0.054221, 33.130250)
(7.127886, -0.054221, 33.130350)
(8.894558, 0.054221, 46.001250)
(8.894558, -0.054221, 46.001350)
(7.331091, 0.054221, 32.295350)
(7.331091, -0.054221, 32.295250)
(9.171240, 0.054221, 45.711250)
(9.171240, -0.054221, 45.711350)
(5.833835, -0.054221, 34.215350)
```

Figure 5. Example of recording hidden values in vertex coordinates: (a) 3D object model (benches) (b) recording of the bench vertices

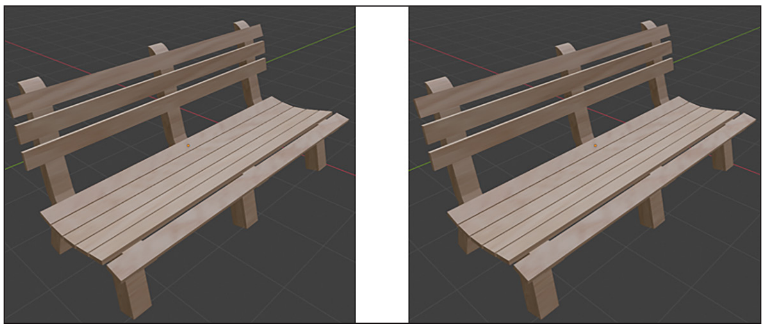


Figure 6. An example of hiding information in the vertices of a 3D object (a) a record of the coordinates of the original object (b) the same object with encoded information (91 bytes)

bit (LSB) algorithm [23, 24]. The effectiveness of this method for two-dimensional images has been proven by numerous studies. Due to the specific nature of the information to be hidden (separated, normal and "short-term"), it was decided neither to hide information in the frequencies of the FFT (Fast Fourier Transform) transtormat [25–27], nor, even less, to Spectral Texture Steganography using the wavelet transform (Figures 7, 10) [28].

Hiding information in UV map

Texturing is an important process when modelling. UV mapping is used for this purpose, thanks to which the creator has control over where a given fragment of the texture appears. The fundamental problem of this scenario is to have as many vertices as possible to manipulate. This is because each character is converted to bit form, requiring 1 to 4 bytes per character using UTF-8 encoding. Depending on the needs, one can manipulate the shift length of a given vertex, and it is also necessary to encode 0 in a different way than 1. It was assumed that one vertex of the model will be able to hide two bits of information. It is assumed that when 0 occurs in the binary string, the vertex is moved so that the tip is 0.000250, and when 1 occurs, it is 0.000350 (as shown in Figure 8), and these operations are performed for both axes. Thanks to such a small coefficient, the changes are so small that people will not notice the difference. This was determined during several variants of research. The algorithm is identical to the one in Figure 9 with the difference that in the 3D object the information is hidden in the Z coordinate and here in the V coordinate.

RESULTS OBTAINED

The research scenarios were created to analyse and examine each of the aspects described above. In the case of hiding data in the metadata of the *.blend file, the size of the file after adding the data was examined, as shown in Figure 11.

The next scenario was an in-depth analysis of hiding data in a computer game on a public server. 20 people aged 15–49, who were not informed about the purpose of the game, were examined. The subjects first moved around the board doing specified tasks and scoring game levels. After

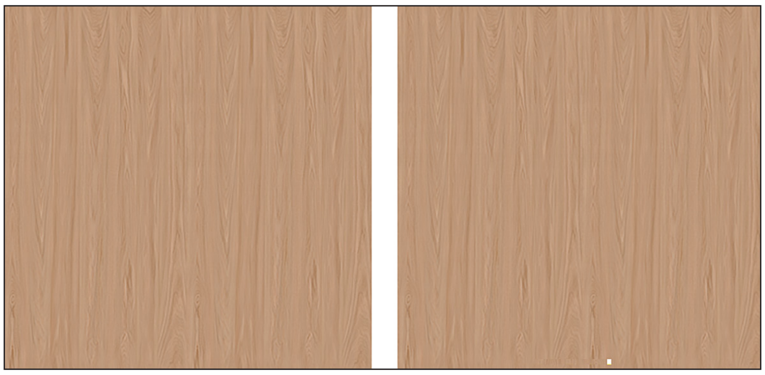


Figure 7. View: (a) original texture (b) texture with encoded information of 123 bytes

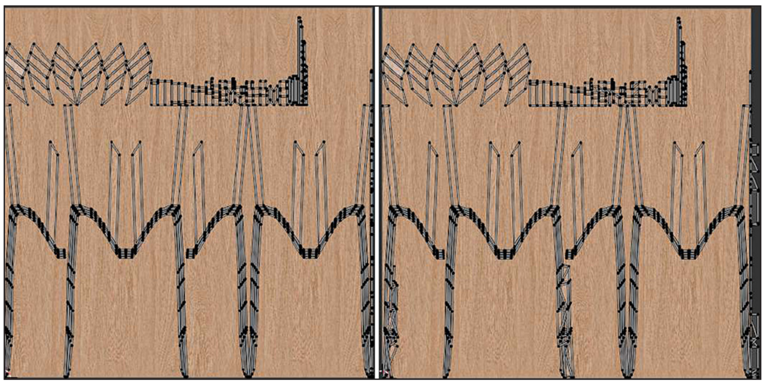


Figure 8. View of UV map (a) original (b) with coded information (91 bytes)

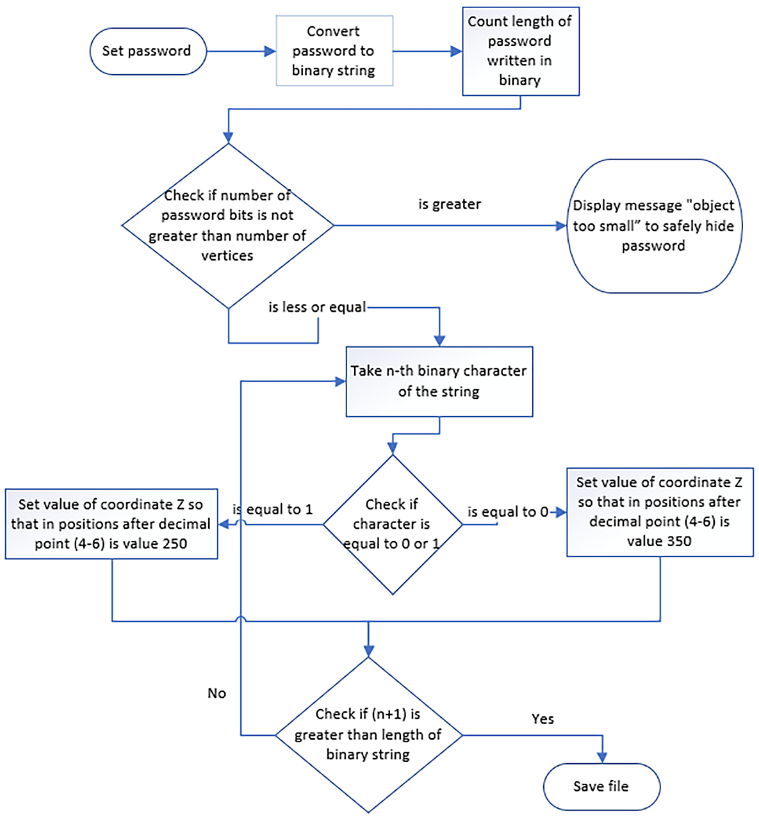

Set password
Convert password to binary string
Count length of password written in binary
Check if number of password bits is not greater than number of vertices
is greater
Display message "object too small" to safely hide password
is less or equal
Take n-th binary character of the string
Check if character is equal to 0 or 1
is equal to 1
Set value of coordinate Z so that in positions after decimal point (4-6) is value 250
is equal to 0
Set value of coordinate Z so that in positions after decimal point (4-6) is value 350
Check if (n+1) is greater than length of binary string
No
Yes
Save file

Figure 9. Algorithm for hiding information in the coordinates of the vertices of the figure. The variable used is Z

collecting a certain number of points, i.e. passing some points in the game, they unknowingly finished the entire game. Despite picking up various items in the game, no hidden information was displayed to anyone.

Texture similarity analysis of objects is only possible using image differentiation: original (without information) and modified (with hidden information). The bitwise similarity of the binary images of the differences observed as a function of the number of encoded bits is also examined, as shown in Figure 12. A measure of Sum of Absolute Differences – SAD -between two image of size N×M was used.

$$SAD = \sum_{i=1}^{N} \sum_{j=1}^{M} (|\mathrm{A(i,j)} - \mathrm{B(i,j)}|) \quad (1)$$

where: $A(i,j)$ is the pixel value at position (i,j) in the first block, $B(i,j)$ is the pixel value at position (i,j) in the second block, N and M represent the horizontal and vertical dimensions of the blocks.

Figure 10. Final view of 3D object with encoded data (in texture, UV map, vertices and metadata)

The application of this formula involves summing the absolute difference values for each pixel in the two images. The SAD value gives an overall measurement of the difference between the two images, where a larger value indicates a larger difference. The texture can be treated as a complete image, the change of individual bits is virtually imperceptible to the human eye. This is shown in Figure 7. Using formula (2), the MSE between the original texture (containing no information) and each subsequent texture with encoded information was calculated (Figure 13). The mean squared error (MSE) coefficient between these textures was counted from equation (2):

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=i}^{N}(I(i,j) - U(i,j))^2 \quad (2)$$

where: M – image height, N – image width, $I(i, j)$ – pixel value at position (i, j) in the original image, $U(i, j)$ – pixel value at position (i, j) in the modified image.

Two more measures given by formulae (2) and (3) respectively were then calculated for the texture under study. Peak signal to noise ratio (PSNR) is a measure of distortions calculated according to the formula [29]

$$PSNR = 10 * log_{10}\frac{N*\vec{V}_{max}}{\sum_{i=1}^{N}\vec{V}_i^{*}-\vec{V}_i^{2}} \quad (3)$$

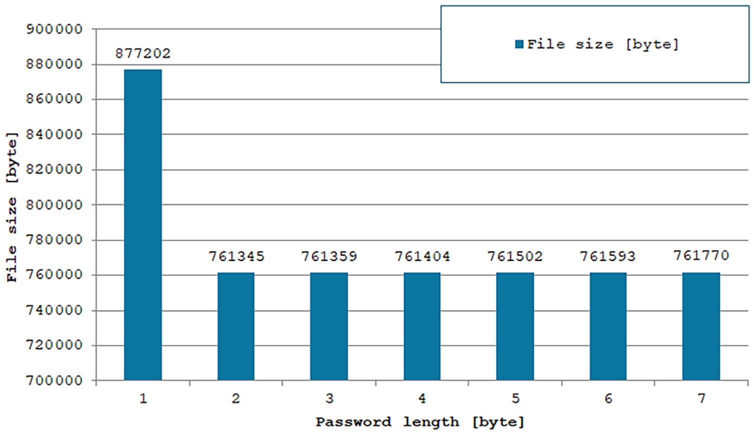


Figure 11. Changing the size of the file with encoded information

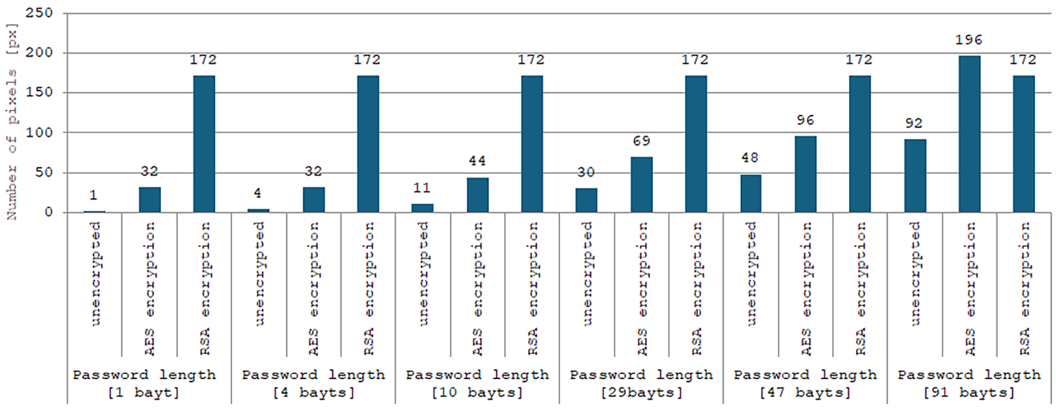
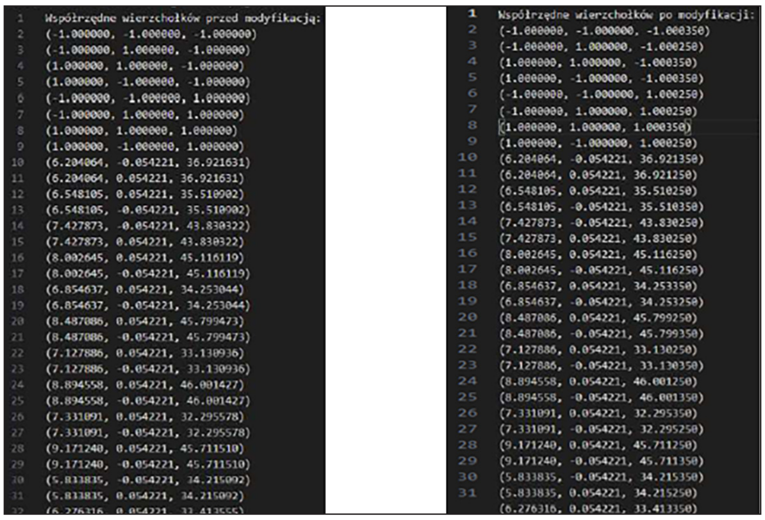


Figure 12. Changing the length of information depending on the method of encryption (hiding in metadata-encrypted part)



```
Współrzędne wierzchołków przed modyfikacją:
(-1.000000, -1.000000, -1.000000)
(-1.000000, 1.000000, -1.000000)
(1.000000, 1.000000, -1.000000)
(1.000000, -1.000000, -1.000000)
(-1.000000, -1.000000, 1.000000)
(-1.000000, 1.000000, 1.000000)
(1.000000, 1.000000, 1.000000)
(1.000000, -1.000000, 1.000000)
(6.204064, -0.054221, 36.921631)
(6.204064, 0.054221, 36.921631)
(6.548105, 0.054221, 35.510902)
(6.548105, -0.054221, 35.510902)
(7.427873, -0.054221, 43.830322)
(7.427873, 0.054221, 43.830322)
(8.002645, 0.054221, 45.116119)
(8.002645, -0.054221, 45.116119)
(6.854637, 0.054221, 34.253044)
(6.854637, -0.054221, 34.253044)
(8.487086, 0.054221, 45.799473)
(8.487086, -0.054221, 45.799473)
(7.127886, 0.054221, 33.130936)
(7.127886, -0.054221, 33.130936)
(8.894558, 0.054221, 46.001427)
(8.894558, -0.054221, 46.001427)
(7.331091, 0.054221, 32.295578)
(7.331091, -0.054221, 32.295578)
(9.171240, 0.054221, 45.711510)
(9.171240, -0.054221, 45.711510)
(5.833835, -0.054221, 34.215092)
(5.833835, 0.054221, 34.215092)
(6.276316, 0.054221, 33.413555)
```

```
Współrzędne wierzchołków po modyfikacji:
(-1.000000, -1.000000, -1.000350)
(-1.000000, 1.000000, -1.000250)
(1.000000, 1.000000, -1.000350)
(1.000000, -1.000000, -1.000350)
(-1.000000, -1.000000, 1.000250)
(-1.000000, 1.000000, 1.000250)
(1.000000, 1.000000, 1.000350)
(1.000000, -1.000000, 1.000250)
(6.204064, -0.054221, 36.921350)
(6.204064, 0.054221, 36.921250)
(6.548105, 0.054221, 35.510250)
(6.548105, -0.054221, 35.510350)
(7.427873, -0.054221, 43.830250)
(7.427873, 0.054221, 43.830250)
(8.002645, 0.054221, 45.116250)
(8.002645, -0.054221, 45.116250)
(6.854637, 0.054221, 34.253350)
(6.854637, -0.054221, 34.253250)
(8.487086, 0.054221, 45.799250)
(8.487086, -0.054221, 45.799350)
(7.127886, 0.054221, 33.130250)
(7.127886, -0.054221, 33.130350)
(8.894558, 0.054221, 46.001250)
(8.894558, -0.054221, 46.001350)
(7.331091, 0.054221, 32.295350)
(7.331091, -0.054221, 32.295250)
(9.171240, 0.054221, 45.711250)
(9.171240, -0.054221, 45.711350)
(5.833835, -0.054221, 34.215350)
(5.833835, 0.054221, 34.215250)
(6.276316, 0.054221, 33.413350)
```

Figure 13. Demonstration of a fragment of vertex coordinate record in 3D object. (a) without coded information (b) with coded information

$\vec{V}_i$

where: N is the overall number of vertices, $\vec{V}_{max}$ is the most remote point from the centre of the model, $\vec{V}_i^*$ is the current vector in the fingerprinted model, $\vec{V}_i$ is a corresponding vector in the original model. The bigger the PSNR, the less distorted is the model.

The results of the dependence of the MSE, SSIM and PSMR coefficient on the number of encoded bits are shown in Table 1. Structural Similarity Index Measurement (SSIM) is a measure originally used to compare the structural similarity of images [30]. It can also be utilised in 3D objects comparison, for example, operation on orthographic projections of the model. It is then calculated according to the formula:

$$SSIM(I_1, I_2) = \frac{(2*\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (4)$$

where: μ_x is the mean over a window in the original 3D image, μ_y is the mean over a window in the fingerprinted 3D image, σ_x is standard deviation (square root of variance) over a window in the original 3D image, σ_y is standard deviation (square root of variance) over a window in the fingerprinted 3D image, σ_{xy} is co-variance over a window between analysed images, x and y refer to a local window in the analysed images, C1 and C2 are small constants as it is in Equation 4 [29].

The next scenario was an in-depth analysis of hiding data in a computer game on a public server. 20 people aged 15–49, who were not informed about the purpose of the game, were examined. The subjects first moved around the board doing specified tasks and scoring game levels. After collecting a certain number of points, i.e. passing some points in the game, they unknowingly finished the entire game. Despite picking up various items in the game, no hidden information was displayed to anyone.

Table 1. Size of files (in bytes) containing the texture and the calculated difference

Password length [byte]	File weight [bytes]	A measure SAD	MSE coefficient value	Number of changed pixels	SSIM [%]	PSNR [dB]
0	877 202	0	0	0	100	--
8	899 764	3	0.369568	8	99.9	221.45
32	899 830	19	0.369572	32	99.9	214.26
80	899 866	46	0.369577	80	99.9	198.17
232	900 019	132	0.369587	232	99.9	184.77
376	900 119	232	0.369596	376	99.8	175.49
736	900 428	427	0.369623	736	99.8	143.12

The next step in testing the usability and security of storing information on a public, publicly accessible VR game server was to define scenarios for an informed person with knowledge of the hidden information. The scenario was repeated 20 times for each variant. The types of variants are given in Table 2 a score of 0 points was introduced for failure 'N' and 1 point for each success 'Y'. The arithmetic mean was then calculated and expressed as a percentage. This is shown in the last column of Table 2. Another research scenario dealt with the possibility of hiding information of vertices deformation of vertices as a function of the number of bits encoded is shown in Figure 14. The last element tested was the hiding of information in UV maps. The results are demonstrated in Figure 15 below.

Comparison to other methods

The proposed steganography method was compared to other solutions available in the literature. However it have to be emphasised that steganography in virtual reality is not explored yet and only few papers discuss the subject. Moreover it is impossible to create exhaustive comparison because of lack of full assessment data in the mentioned papers. Each author uses various measures and assesses his method in the other way. In addition other authors focus on

Table 2. Data access success rate

Condition	User logged in as: "guest"	User correct: (login, password)	IP/MAC address of the player's computer: invalid	IP/MAC address of the player's computer: correct	Start the correct level in the VR game	Correct places in the VR scene	Correct time	Data access success rate
1	Yes	No	No	Yes	Yes	Yes	Yes	0%
2	Yes	No	Yes	No	Yes	Yes	Yes	0%
3	Yes	No	No	Yes	No	No	Yes	0%
4	Yes	No	No	Yes	Yes	Yes	No	0%
5	No	Yes	Yes	No	Yes	Yes	No	0%
6	No	Yes	Yes	No	Yes	No	No	0%
7	No	Yes	Yes	No	Yes	No	Yes	0%
8	No	Yes	Yes	No	Yes	Yes	No	0%
9	No	Yes	Yes	No	Yes	No	No	0%
10	No	Yes	No	Yes	Yes	Yes	Yes	100%

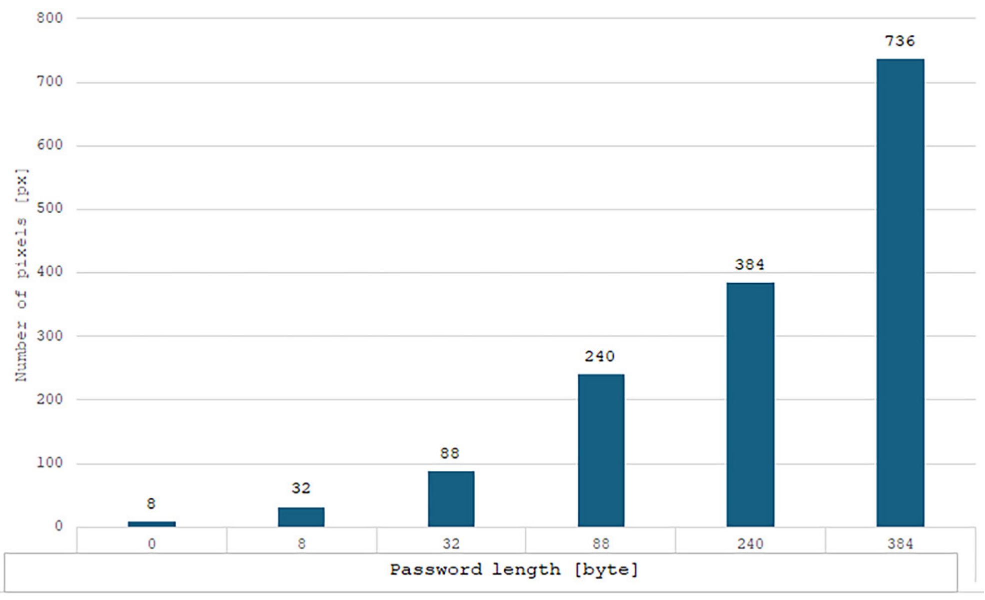


Figure 14. The effect of changing the password length on the number of changed pixels in the image

Table 3. Comparative analysis of information hiding methods in VR

Proposed method	Dividing secret content into independent parts	Sending various parts of secret data via various channels	Possibility of secret content data read if attacker find one part of hidden data	SSIM [%]	PSNR [dB]	Hiding information in:
	Yes	Yes	No	99.8–99.9%	221.45–143.12	Virtual reality, 3D models
Method 1 [19]	No	No	Yes	99.7–99.6%	158.6–148.3	3D models
Method 2 [31]	No	No	Yes	Lack of data	Lack of data	Virtual reality
Method 3 [32]	No	No	No	Lack of data	Lack of data	Augmented reality
Method 4 [33]	No	Yes	Yes	Lack of data	Lack of data	Augmented reality

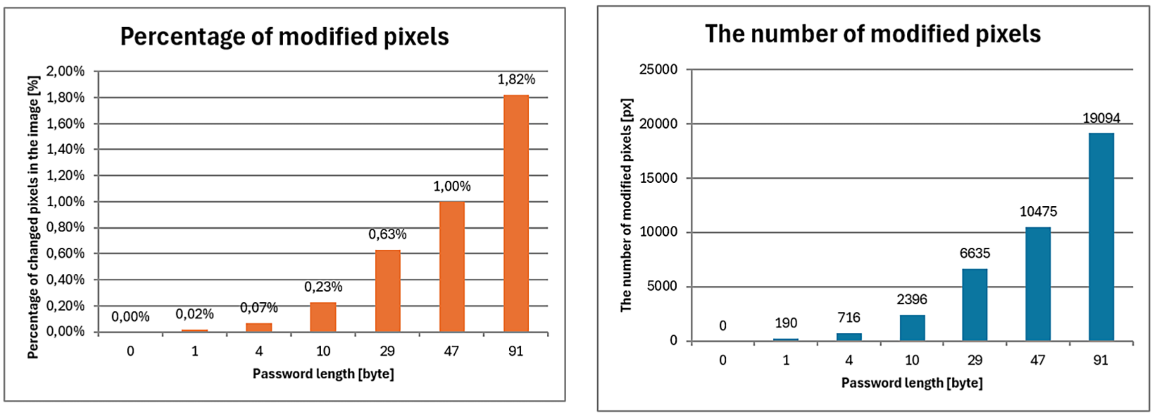


Figure 15. Dependence between the length of the hidden information and changes in the image. a) percentage dependence of changes b) quantitative dependence of changes expressed in pixels

hiding data in one channel, without dispersing it among various channels what is the main goal of the presented paper (Table 3).

CONCLUSIONS

3D objects combined with virtual reality are an excellent carrier of information, the number of elements where information can be hidden and its appropriate division into parts corresponding to the main part of the information as well as parts that are important only in a certain period of time and then simply outdated seems to be an interesting approach to the process of data hiding. Generally, all project assumptions were met.

Using Blender binary files, the Python programming language has managed to hide fragmented information correctly many times. The space on the VR server with Unity software and a MySQL database will hold over 65 kB of text the problem will be displaying it in VR. The metadata of the *.blend file has a much smaller capacity than MySql (BLOB, TEXT) but it will also hold a long descriptive text. Hiding in the UV Map and in the vertices of a 3D object Strictly depends on the number of vertices available there. An attempt to hide there (the simplest method, LSB type) too much information immediately changes the optical appearance of the figure and reveals the presence of information. The same applies to hiding data in the texture.

However, it should be noted that if the blend file format is changed, reformatting to another format, e.g. obj, causes not only changes in the texture and very often not converting it to another file, i.e. loss, which automatically involves the loss of data saved both in the texture and the UV map. The hidden information is also lost when converting to another *.obj or *.str data structure.

It is worth mentioning, however, that once a Blend model is well-built, it is not, by definition, compressed or converted to another format.

Hiding information in the VR application proposed by the authors brings other problems. Changing the computer, even due to damage and failure to save the MAC address of the original machine, will also prevent access to data stored in virtual reality on the server. The proposed very complex method of hiding data using various aspects, however, is not destined to fail. By design, the data are not converted to any other format and the user connects to the computer from a specific IP/MAC computer and both of these parameters are improved. In the event of damage to the machine, the user can use specialised software to change both the IP address to the correct one and also the MAC address of the network card to the correct one.

Acknowledgements

The authors declare equal contribution to the paper. The authors are grateful to the Lublin University of Technology for the financial support granted to cover the publication fees of this research article.

REFERENCES

1. Farrag S, Alexan W. Secure 3D data hiding technique based on a mesh traversal algorithm. Multimedia Tools Appl. 2019. doi:10.1007/s11042-019-07888-1.
2. Liu X, Lee H, Jiang Y. Adversary-guided motion retargeting for skeleton anonymization. arXiv. 2024. doi:10.48550/arXiv.2405.05428.
3. Feng X, Lu Z, Yang W. A robust image steganography based on edge detection and LSB matching revisited. Multimedia Tools Appl. 2017; 76(20): 21401–17. doi:10.1007/s11042-017-4866-2.
4. Jain N, Chaudhary S. Secure and high capacity image steganography technique using curvelet transform and LSB insertion. J Inf Secur Appl. 2018; 41: 41–51. doi:10.1016/j.jisa.2018.05.003.
5. Kaur A, Aujla GS. A hybrid approach for image steganography using LSB, DCT and DWT. Multimedia Tools Appl. 2019; 78(1): 425–37. doi:10.1007/s11042-018-6281-7.
6. Wang H, Zhang H, Liu X. Improved adaptive image steganography based on deep convolutional neural networks. IEEE Access. 2020; 8: 19083–93. doi:10.1109/ACCESS.2020.2968736.
7. Chan C-K, Cheng L-M. Image steganography based on LSB matching revisited. IEEE Trans Inf Forensics Secur. 2011; 2(2): 190–7. doi:10.1109/TIFS.2007.898004.
8. Kozieł G. Steganography usage to control multimedia stream. Adv Sci Technol Res J. 2014; 8(21): 5–8.
9. Li J, Zhang Y. A novel steganography technique using texture mapping function and convolutional neural networks. Appl Sci. 2020; 10(12): 4211. doi:10.3390/app10124211.
10. Song H, Huang Y. Steganography based on PCA and texture synthesis. Signal Process. 2016; 128: 59–66. doi:10.1016/j.sigpro.2016.03.003.
11. Wang Z, Wu S. Deep learning-based steganography for texture images. IEEE Access. 2019; 7: 167614–23. doi:10.1109/ACCESS.2019.2954112.
12. Cheng Y, Wang C. A high-capacity steganographic approach for 3D polygonal meshes. Vis Comput. 2006; 22(9–11): 845–55. doi:10.1007/s00371-006-0069-4.
13. Chuang H, Cheng C, Yen Z. Reversible data hiding with affine invariance for 3D model. In: IET International Conference on Frontier Computing, Theory, Technologies and Applications; 2010. 77–81. doi:10.1049/cp.2010.0541.
14. Alkhamese A, ElGhawalby H, Eid A, Hanafy I, Awad W. Highly secured 3D object steganography technique for hiding compressed data. Alfarama J Basic Appl Sci. 2024; 5(2): 208–20. doi:10.21608/ajbas.2024.228974.1171.
15. Li N, Hu J, Sun R, Wang S, Luo Z. A high-capacity 3D steganography algorithm with adjustable distortion. IEEE Access. 2017; 5: 24457–66. doi:10.1109/access.2017.2767072.
16. Song H, Kim J. Efficient data hiding in 3D geometric models using UV mapping. J Comput Sci. 2017; 35(4): 123–34. doi:10.1007/s00530-017-0134-8.
17. Lee S, Park J. Robust steganographic techniques for 3D models based on UV coordinates. Comput Graph. 2019; 83: 123–38. doi:10.1016/j.cag.2019.07.002.
18. Wang X, Zhang Y. High-capacity data embedding in 3D models using UV map distortion. IEEE Trans Inf Forensics Secur. 2020; 15: 1025–34. doi:10.1109/TIFS.2019.2933456.
19. Kozieł G, Malomuzh L. 3D model fragile watermarking scheme for authenticity verification. Adv Sci Technol Res J. 2024; 18(8): 351–65. doi:10.12913/22998624/194146.
20. Klubsuwan K, Mungsing S. Digital data security and hiding on virtual reality video 3D GIS. Int J Adv Comput Sci Appl. 2020; 11(9): 572–9.
21. Zhao Q, Liu H. Secure information hiding in 3D meshes with UV mapping. Comput Secur. 2023; 113: 102484. doi:10.1016/j.cose.2022.102484.
22. Sadurski P, Szymczyk T. Application for hiding information in virtual reality [Engineering thesis]. Lublin University of Technology; 2023.
23. Kumar A, Bala A. A novel image steganographic method based on LSB substitution technique. J Inf Assur Cybersecurity. 2017; 1(2): 65–72.
24. Jain A, Singh V, Agarwal R. Adaptive image steganography based on LSB matching revisited. J Comput Sci Technol. 2019; 3(4): 112–9.
25. Kozieł G. Simplified steganographic algorithm

based on Fourier transform. Adv Sci Lett. 2014; 20(2): 505–9. doi:10.1166/asl.2014.5322.

26. Sharma A, Kumar R. A novel approach for image steganography based on wavelet transform and fast Fourier transform. Int J Comput Appl. 2018; 180(29): 12–8.
27. Zhang Y, Li L. Image steganography based on combined wavelet-FFT transformation. J Vis Commun Image Represent. 2016; 38: 472–80.
28. Gupta S, Sharma V. A novel approach for image steganography based on discrete wavelet transform. J Inf Hiding Multimed Signal Process. 2018; 9(3): 512–21.
29. Structural Similarity Index. Available from: https://www.ni.com/docs/en-US/bundle/ni-vision-concepts-help/page/structural_similarity_index.html. Accessed 2025 Oct 21.
30. Kim H, Yang K, Kim Y. A study of the perceptually weighted peak signal-to-noise ratio (WPSNR) for image compression. IEEE Xplore. 2019. doi:10.1109/ICIP.2019.8803307.
31. Klubsuwan K, Mungsing S. Digital Data Security and Hiding on Virtual Reality VDO 3DGIS-Map. Int J Manag Sci Eng Manag. 2009; 4(3): 163–76. Available from: doi:10.1080/17509653.2009.10684575.
32. Patrão B, Cruz L, Gonçalves N. Large Scale Information Marker Coding for Augmented Reality Using Graphic Code. In: 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR); 2018 Dec 10-12; Taichung, Taiwan. IEEE; 2018, 132–5. doi:10.1109/AIVR.2018.00034
33. Li C, Sun X, Li Y. Information hiding based on Augmented Reality. Math Biosci Eng. 2024; 16(5): 4777–4787. doi:10.3934/mbe.2024.1.4777.