applied sciences

MDPI

*Article*

# Image Analysis and Processing for Generating Camouflages from Digital Earth Photographs

**Aneta Poniszewska-Marańda *, Michał Suszek and Krzysztof Stepień**

Institute of Information Technology, Lodz University of Technology, Al. Politechniki 8, 93-590 Lodz, Poland
* Correspondence: aneta.poniszewska-maranda@p.lodz.pl

**Abstract:** Camouflage is present both on the civilian market and in military. It is used by hunters and military enthusiasts, but also by clothing designers and game developers. They are forced to design their own or choose between already developed camouflages, both military and civilian, so they cannot adapt it to their needs in a fast and easy way. Currently, there are not many software solutions that allow for easy generation of digital camouflages and support the user in selecting the colors for final camouflage result. The approach presented in the paper proposes to solve these problems by analyzing the colors from digital Earth images of the target areas and using the developed image processing algorithm for generating digital camouflages. Based on the proposed approach and its designed algorithm, the application was created to allow the user to easily generate the digital camouflages. The paper also presents the results of analysis of camouflage quality, comparing the camouflages generated with the developed algorithms and their application and the selected market generators together with the selected military digital camouflages. By using the proposed algorithm to generate the camouflage and implementing the centroid algorithm for color extraction, it was possible to create better quality camouflages compared to those created by existing software solutions. This was supported by an analysis of the camouflage quality in chosen terrain variants, in which developed application achieved the best results.

**Keywords:** image analysis; generating images; digital camouflage; camouflage generation; modeling



**Citation:** Poniszewska-Marańda, A.; Suszek, M.; Stepień, K. Image Analysis and Processing for Generating Camouflages from Digital Earth Photographs. *Appl. Sci.* **2023**, *13*, 403. https://doi.org/10.3390/app13010403

Academic Editor: Athanasios Nikolaidis

Received: 31 October 2022
Revised: 15 December 2022
Accepted: 22 December 2022
Published: 28 December 2022



## 1. Introduction

Image processing is a broad topic. Many types of objects can be represented as an image, for example, printed texts, digital Earth images, medical images, or handwriting, so the term "pattern processing" is more appropriate. The simplest definition of this concept boils down to the qualification or grouping of patterns based on their structure or content. Image processing involves image analysis and classification. Image analysis is the extraction of features based on the image, and image recognition (classification) is the assignment of the image to a class based on the extracted image features [1–3].

Camouflage in nature has been around for a long time; animals do not want to be detected by predators, and predators do not want to be detected by their prey. In humans, the art of camouflage was developed when hunters wanted to move as close as close as possible to their victim without being detected in order to increase their chances of a successful hunt. The history of military camouflage, or camouflage used by the military, began during World War I when, along with the development of technology on the battlefield, the military noticed that colored uniforms did not help soldiers to survive during skirmishes that took place at distances of 300 m [4–6].

The lessons learned from experience from battles of that period showed the effectiveness and necessity of camouflage, and outlined further developments in the field of camouflage. Subsequent conflicts and advances in technology led to research on the most effective camouflages. With the development technologies that effectively supplement human vision, such as thermal imaging, the need for camouflage continues and is very

important to the military. Camouflage not only helps to hide the target, its role is also to build an image of the army as a professional, well-prepared, but also elite, group.

Camouflage is also present on the civilian market, for which the camouflages are used by hunters, military enthusiasts, organizers of various types of survival events, clothing designers, and computer game developers. They can design their own camouflages or choose from already developed and available camouflages—military or civilian. In this way, they can quickly and easily achieve what they need by adapting the available camouflage to specific needs. There are few companies on the market that offer the possibility of designing one's own camouflage, and their offer is addressed mainly to hunters. There are some private camouflage projects on the Internet, but they have their disadvantages, which will be described later in this paper. Therefore, there is a lack of solutions on the market, which gives us the possibility to, in an easy and fast way, analyze the digital Earth images given by the user to allow him generate the appropriate camouflage.

The generation of camouflages from the digital Earth images is an example of image analysis and processing [3,7]. This paper analyzes the current state-of-the-art in the aspects of existing camouflage generators along with determining the advantages and disadvantages of their solutions, and then the development of camouflage generation algorithm and creation of an application that allows the user to generate the camouflages while offering support in color selection. Moreover, the results of the generator were compared with other camouflages available on the market as well as those generated by online generators.

The main contributions of the paper are as follows:

1. Analysis of currently existing camouflage generators with their advantages and disadvantages.
2. Development of algorithms for generating the digital camouflage and the creation of an application that will allow the user to generate camouflage based on digital Earth images while offering support in the selection of colors.
3. Comparison of results in camouflages generation of available software tools and created solution by:
4. Assessing the efficiency of various approaches of machine learning when applied in image analysis and processing field on the example of camouflages generation;
5. Considering the possibility of image analysis and processing for generating camouflages.

The paper is structured as follows: Section 2 presents the overview of selected issues regarding the camouflage and analysis of selected online camouflage generators. Section 3 describes the technical aspects of the camouflage generator approach—the camouflage generator algorithm and developed "CammoGenerator" application, together with its architecture. Section 4 deals with analysis and verification of the camouflage generation algorithm and developed application.

## 2. Selected Practical Aspects of Camouflage

The general idea of a camouflage is to hide an object (building, vehicle, person) or change visible features (e.g., speed or direction). Properly selected colors, shapes, materials, and lighting are used for this [8].

### *2.1. Principles of Camouflage Operation*

Depending on the used technique, the following types of camouflage can be distinguished [4,5,9]:

- Decorative —it is based on the use of artificial masking structures, which are used to show a different state of the object than its true state. It can be, for example, an imitation of a destroyed building above a combat post, which makes it uninteresting to the enemy, while maintaining its defensive potential.
- Imitative—it consists of making a given stationary object similar to another harmless object of the surroundings, e.g., a camouflage showing the position of a soldier hidden under a trunk, which is a natural object that does not arouse suspicion found in forests.

- Color or classic camouflage—by changing the colors of the object, as well as the appropriate deforming patterns, it blends the object into its surroundings, e.g., the need for at least a minimal similarity of the object's color to its surroundings. The example in Figure 1 shows the difference between no camouflage and color masking.
- Natural and vegetal—it consists of using camouflage to mask the natural properties of the terrain, e.g., fog, as well as live or cut vegetation. Live vegetation is used to mask stationary positions, which makes it difficult to see objects from the air and land. The cut vegetation allows masking moving and smaller objects. Such a camouflage is effective in the area from which the vegetation was obtained, and its durability, which translates into effectiveness, decreases with time, e.g., the use of grass to disguise the soldier and his weapons. The example in Figure 2 shows the use of grass to camouflage a soldier and his weapon.

**Figure 1.** Example of color camouflage—a tank without camouflage on the **left**, camouflage on the **right** [6].

**Figure 2.** Example of vegetal camouflage—grass camouflage [6].

There are also other methods of camouflage, but their purposes do not coincide with the purpose of the camouflage presented in this paper, so they were omitted. The main technique used in the work was color camouflage, which, due to the ease of application on objects and the possibility of staying in motion of camouflaged objects, is the best masking technique, adapted to the needs of the recipients.

### *2.2. Design Classification*

Color camouflages use various phenomena to achieve the desired effect. Some of them break the silhouette of the object well, and others make the object blend in with the surroundings. They can also use different techniques, e.g., some camouflages use geometric shapes, and others use overlaid photos of vegetation or the ground. Due to these differences, the camouflages can be classified as follows [4,5,8]:

- Mimetic—usually made up of small spots to make the object resemble its surroundings, which naturally consists of a noise of various colors. These spots are used to make

the object similar to a given color noise occurring in a given area. There is a micro- and macro-pattern in mimetic patterns. The camouflages presented in Figure 3 are an example of German mimetic camouflages from the times of the Second World War.

- Deformational—composed of overlapping geometries of soft, natural, or sharp geometric shapes. Camouflages of this type are very effective at longer distances, because large shapes do not merge into one color.
- Pixel/Digital—currently the most popular type of camouflage. Its pattern is built by pixels filled with appropriate colors. The principle of operation is similar to the mimetic camouflage; it also has a micro- and macro-pattern. The camouflage shown in Figure 4 is one of the first digital camouflages.

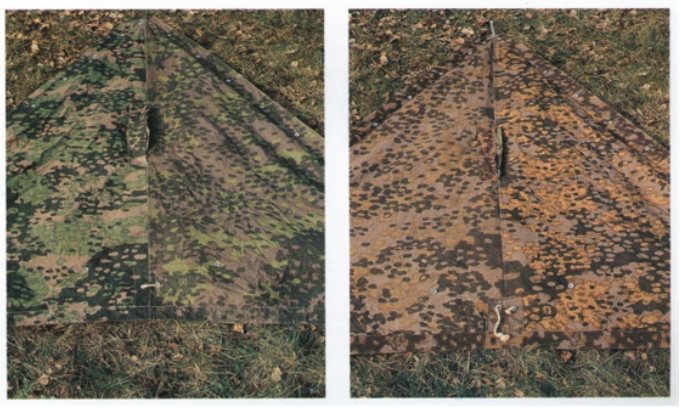

**Figure 3.** Example of Zeltbahn mimetic camouflage in spring (**left**) and autumn (**right**) versions [8].

The *micro-pattern* is a composition of small spots that can be distinguished only when viewed from close range [4–6]. The role of this type of pattern is to hide the object from spatial vision, i.e., the area where the observer does not focus the pattern and thoughts. The micro-pattern makes the camouflaged object blend into its environment through carefully selected small shapes and colors that are similar to the natural noise of colors found in nature. It is ideally suited to hide from spatial vision. Figure 5 presents an example scale of micro-pattern. An example of a micro-pattern is shown in Figure 3, where the camouflage is made up of a large number of small spots, i.e., a micro-pattern. In addition, it can be observed in Figure 4, where it appears as small spots that merge into larger lumps.

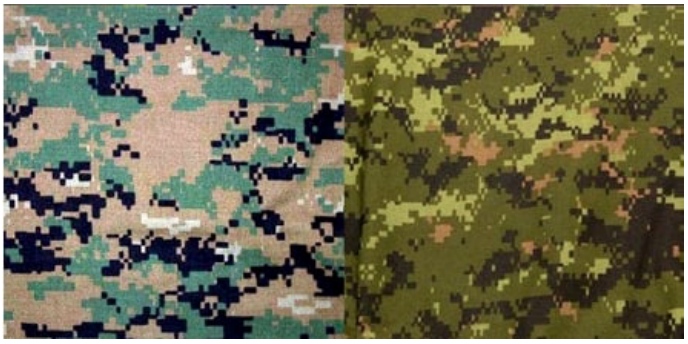

**Figure 4.** Examples of pixel camouflages: MARPAT on the **left**, CADPAT on the **right** [6].

**Figure 5.** Example of determining the micro scale of a pattern on human silhouette [4].

The *macro-pattern* is a large stain designed to break the shapes of camouflaged objects. Its role is mainly to influence central vision, focused on finding shapes. When an object is spotted through spatial vision, the observer will focus on it and try to identify the object by recognizing the shape of the shadows as well as the lines of symmetry. The right macro-pattern is able to confuse the observer's eyesight, who will not be able to match it with the appropriate pattern, and may inadvertently consider it as unimportant, which will lead to the target not being detected. Figure 6 shows an example of macro-pattern deforming a human silhouette.

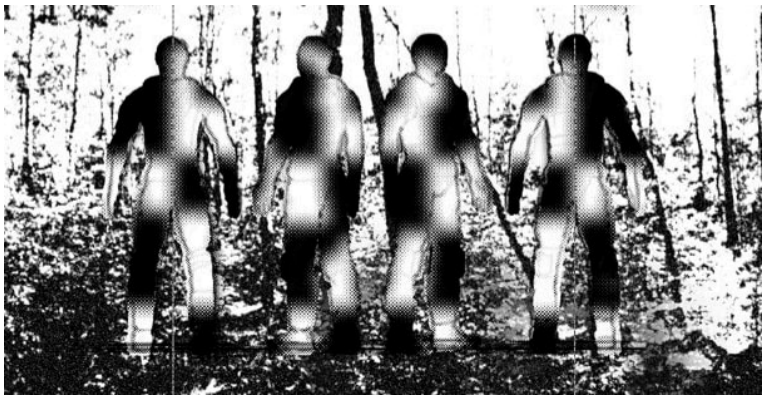

**Figure 6.** Example of determining the micro scale of a pattern on human silhouette [4].

### *2.3. Colors and Their Role in Camouflages*

Camouflage colors are a very important element in its creation [10–12]. During the stage of creating the camouflage, we make sure that the colors are as close as possible to those in the environment. Such colors can be obtained from the analysis of photos of areas at different times of the day and year, as well as weather conditions. When creating universal camouflages, we analyze the environments of a given area and choose the most common colors. However, this approach may lead to a situation where we choose colors that are very similar to each other. It will lead to a situation where our pattern will be practically invisible. In order to prevent such a situation, the adjacent colors should be chosen so that they are contrasting with each other, which strengthens the macro-pattern. Such a selection will make the pattern visible, but we cannot choose too contrasting colors, because the micro-pattern will suffer. The colors combined with the appropriate pattern create a counter-shade [4], which consists of painting the less illuminated elements brighter and the most illuminated ones darker. The use of such a technique allows to hide the solids of an object by hiding shadows that should appear naturally in a given place, presented in Figure 7 [13,14].

**Figure 7.** Example of the counter-shadow phenomenon on Gazelle [4].

## 3. Overview of Available Camouflage Generators

Currently, there are no military camouflage generators available on the market. People interested in using camouflage must use already designed commercial or military solutions. However, there are a few shared private camouflage generators that can be found on the Internet. Two of the most popular generators were selected for analysis: Camouflage Generator [15] and Digital Camouflage Generator [16]. During the analysis, only pixel patterns were selected in order to obtain a reliable and equal comparison.

*Camouflage Generator* [15], created by Ulf Åstrom, is one of the most extensive camouflage generators available. It offers a large selection of types of generated patterns; these are: two types of pixel and four types of deformation with already defined color palettes. It also provides a limited ability to edit the appearance of the camouflage in terms of color, degree of pixelation, and size of spots. The camouflage generator is available as a website, a preview of which is shown in Figure 8.

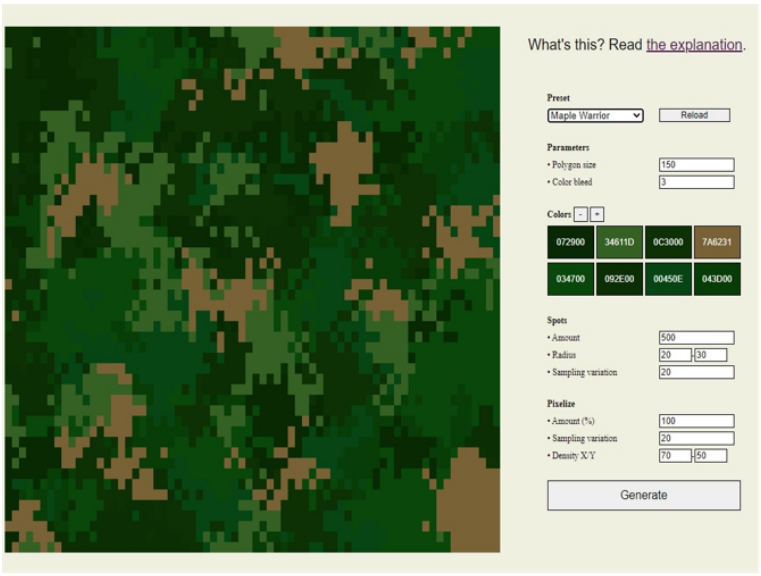


**Figure 8.** General view of Camouflage Generator [15].

The application consists of a place where the camouflage preview is displayed, a drop-down list that allows to select the type of camouflage, a panel with a palette of colors used in the camouflage, places to enter the parameters of generated camouflage, and a button to generate. The generated pixel camouflage has an outlined micro-pattern in the form of small spots on the larger shapes of macro-pattern. Due to the density of the spots and their small size (in the context of macro-pattern), the phenomenon of deformation does not occur here, and the silhouette of an object will not be properly deformed.

Thanks to predefined patterns with colors and simple editing, the application allows for intuitive generation of various camouflages. The main disadvantage is the lack of the ability to edit the scale and size of the camouflage, which is a considerable difficulty, especially when there is a need to apply a pattern to, e.g., part of an outfit. Additionally,

the user has no support in choosing the colors, so he has to analyze the terrain where the generated camouflage will be used. The generated pattern does not have a macro-pattern, so the pattern loses its effectiveness.

*Digital Cammo Generator* [16] is a simple browser-based digital camouflage generator. There is only one type of pattern available in two color versions: forest and desert. Both color versions already have a prepared color palette, without the possibility of changing them. The size range of rectangles from which the generated camouflage will be built and the density of their occurrence are editable.

The generated camouflage has a micro-pattern in the form of color noise, made of rectangles, but it does not have a macro-pattern, which makes it a mimetic type, without deforming properties, which can be a significant difficulty in masking a given object. Likewise, it is not possible to change the scale and size. The application has a place for the generated camouflage, a panel with two color palettes to choose from, three parameters, and a button to refresh the preview.

Digital Cammo Generator is a simple application that allows to generate very simple and not advanced patterns. Its simplicity facilitates the generation process, but it does not allow to adjust the generated pattern to one's own needs. A predefined color palette speeds up the process of creating a camouflage, but the inability to change colors means that the given generator creates camouflages that are limited to a very small number of areas.

In the case of both generators, the user does not have the tools that would allow him to choose the right colors for his own camouflage. Such a function would allow for a faster and more effective color selection process, which would translate into the effectiveness of a given camouflage. The generated patterns do not contain the macro-pattern. It reduces the camouflage properties due to the lack of stains deforming the solids of the camouflaged object. Moreover, the size of the pattern is always constant, which is a problem when the user wants to apply it to a large area. Such a pattern, due to too-large scale used when combining several generated patterns, will become ineffective due to too-sharp and regular joining. Both generators have predefined color palettes so that the user can see a sample pattern, but not all of them allow to change these colors. It means that the created patterns cannot be adapted to the conditions in the areas of interest to the user [17,18].

Based on the analysis presented above, it can be seen that current camouflage generators do not offer assistance to the user in the color selection process. These applications also have a problem with creating an effective macro-pattern, which is an important element of effective camouflage. The collected advantages and disadvantages make it possible to define the functions that the proposed application should have.

## 4. Selected Technical Aspects of the Camouflage Generator Approach

The *CammoGenerator* application allows to generate digital camouflages based on digital Earth images. The user gives the digital Earth images of his interest and then creates a color palette based on them, which he then uses to compose his own camouflages. Based on the application operation, the following functional requirements were formulated:

- The user can add colors to the color palette in two ways: (1) enter a picture and generate the given number of colors, then add the selected colors to the palette; (2) choose the color we want from the dialog box. When the user selects at least two colors, the CammoGenerator application will allow access to the camouflage edit window.
- The user has the option to remove a color from the palette.
- The user can change the size of the generated image and its background color.
- The user has the option to add or remove a color from the camouflage.
- The user has the option to generate a camouflage and save it in a file.
- The user has the option to enter the name and format of the file.
- The application checks if the colors in the color palette are not repeated.
- The application will store and display the colors that were generated from the photo by the user.

The *CammoGenerator* application was created in a multitier architecture that separates the user interface, processing, and data storage from each other into separate layers. Such architecture allows for independent modification of each layer separately, without adversely affecting other layers.

The implementation of the three-tier architecture was carried out by separating three layers, presented in Figure 9. The presentation layer includes four classes responsible for interaction with the user and presenting data obtained from the logic layer. The logic layer consists of two components: *ColorExtractor*, responsible for providing the function of extracting colors from the image, and *CammoGenerator*, which provides a GUI interface for communication with the data layer and a simple interface for generating a camouflage. The individual layers are described in detail in the following subsections.

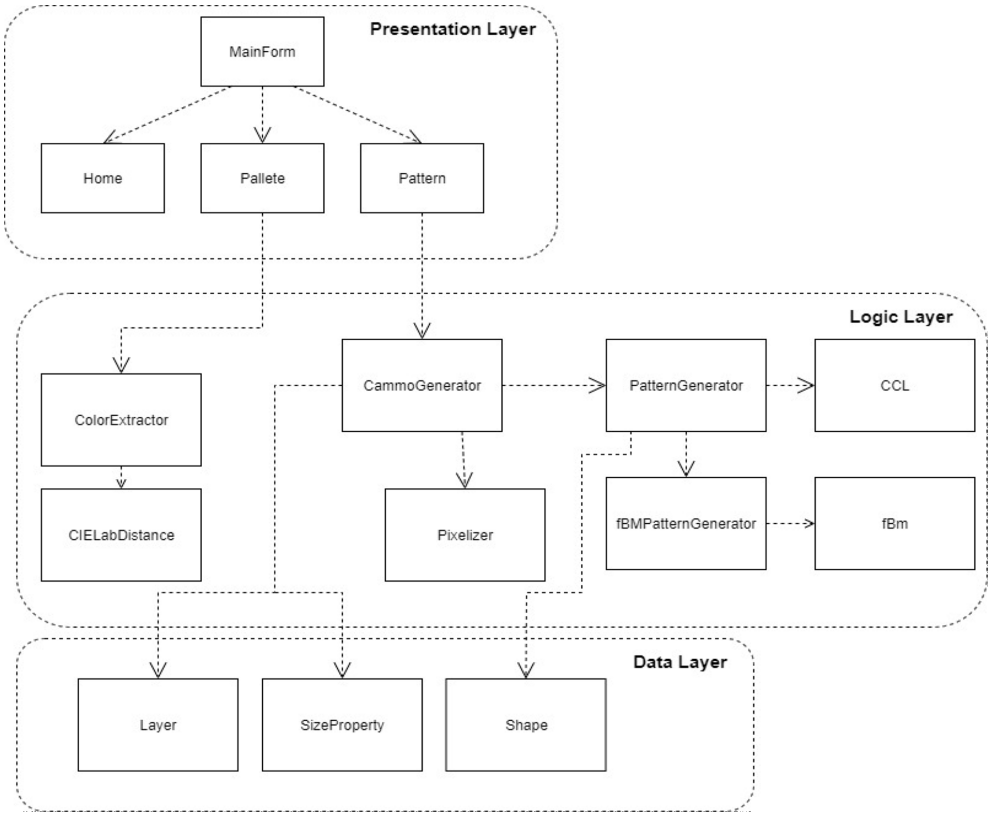


**Figure 9.** The architecture of *CammoGenerator* application.

### *4.1. Data Layer*

Due to the lack of the need to store and load data in the application, the data layer does not communicate with any database or file. This layer consists of three classes, presented in the class diagram in Figure 10, storing the data needed for the generator to operate: (1) *SizeProperty* class is responsible for storing information about the size of the generated camouflage; (2) *Layer* class stores the information needed to generate a single color in a camouflage, hereinafter referred to as a layer; (3) *Shape* class contains information about a single spot in a camouflage. Extracting shapes from a layer is a deliberate procedure that allows for a simple and effective implementation of the effect of interpenetrating color layers, which is described in the next section.

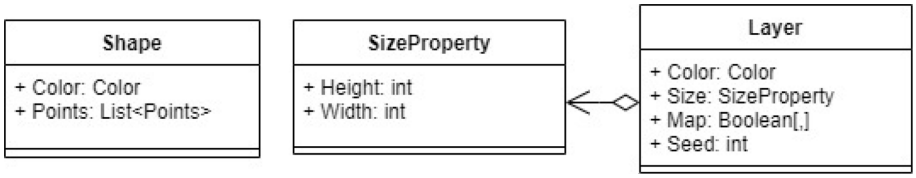


**Figure 10.** Data layer class diagram of *CammoGenerator* application.

### *4.2. Logic Layer*

The logic layer has two components: *ColorExtractor*, which provides an interface for extracting colours from an image, and *CammoGeneratror*, which provides an interface for generating and editing camouflage, and gives the presentation layer access to the data layer. Such a division of logic allows for the separation of functions into independent components, which can be changed and expanded in the future, depending on the needs.

#### 4.2.1. ColorExtractor Component

The *ColorExtractor* component is responsible for providing the function of extracting colors from an images. The class diagram in Figure 11 shows the relationship between the implemented classes and the classes from external packages. The main class of the component is the *ColorExtractor* class with a static *GetPallete()* method that takes a bitmap image, the number of colors the user wants, and BackgroundWorker to monitor and report the progress of the color extraction operation. The *ColorExtractor* class uses the *kMeans* external class from the *Accord.MachineLearing* package. It uses the implemented *Distance()* method from the *CIELabDistance* class, which implements the *IDistance* interface from the *Accord.Math.Distance* package, to calculate the distance between colors. To calculate the distance between colors, the *CIELabDistance* class uses the *CnCComparison* class from the *ColorMine.ColorSpaces* package.

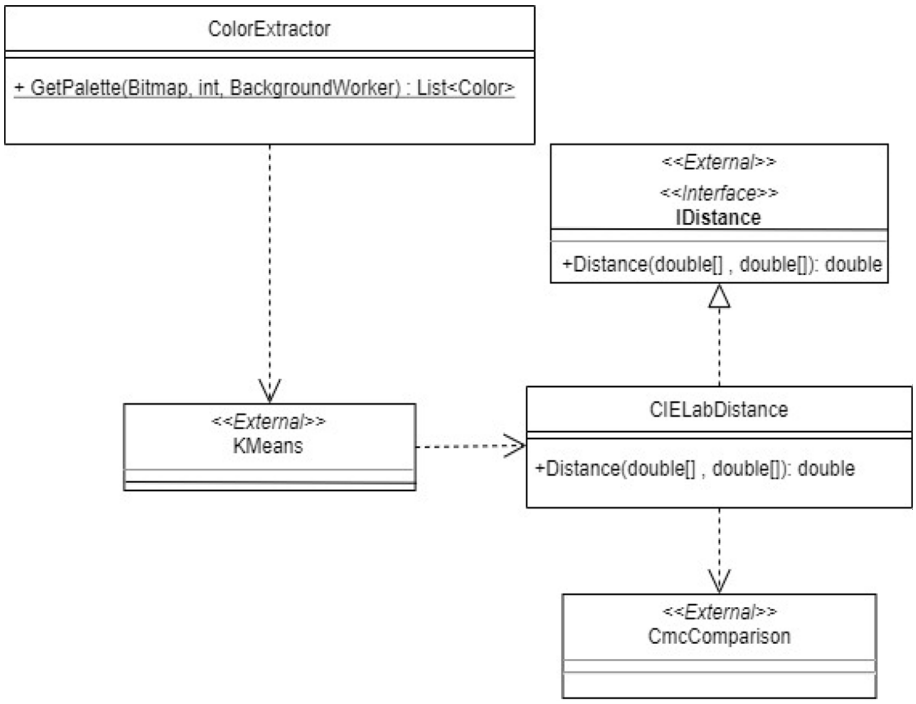


**Figure 11.** Class diagram for *ColorExtractor* component in logic layer of *CammoGenerator* application.

The *color extraction algorithm* is based on the centroid algorithm (*kMeans*), which, thanks to the use of *CIELab* color space, allows to group colors according to their similarity, similar to the perception of color differences in humans. For the grouped colors, the average color

for each of the groups obtained is calculated. The obtained average colors after conversion to the RGB color space are the final result of the algorithm.

The *Kmeans* class from the *Accord.MachineLearnig* library contains the implementation of the centroid algorithm. This algorithm groups similar points concentrated around the searched centroids, being the mean value of the value of the points assigned to a given group (centroid). When the image is analyzed, the algorithm obtains a set of colors that are vectors of the three values. Then, the initial values of the centroids are drawn. After they are drawn, the colors are assigned to the centroid to which they are most similar. When all colors are separated, centroids are assigned the average value of the colors in their set. Then, the mean square error is computed. When this error is greater than the assumed tolerance, we repeat all operations, omitting the initial randomization of centroid positions. In the case when the calculated error is lower than the assumed tolerance, the algorithm is interrupted.

#### 4.2.2. CammoGenerator Component

The *CammoGenerator* component is based on a facade design pattern that provides a simple interface for generating camouflage and managing its properties. The classes included in this component are shown in the class diagram in Figure 12. Facade is the *CammoGenerator* class that provides parameters of the generated camouflage such as size, background color, layers, and *Generate()* method, which is used to generate the camouflage. The facade needs a *PatternGenerator* class that provides *GetShape()* method to generate a spot list for a given layer, and a *Pixelizer* class that has *Pixelize()* method to convert a bitmap image into a square image of a given size. For the *GenerateMap()* method to work, the *PatternGenerator* class requires the use of the *ConectedComponentLabeling* and *fBMPatternGenerator* classes. The *ConectedComponentLabeling* class is used to convert a two-dimensional list of logical type into a list of spots, while *fBMPatternGenerator* is a class for generating the pattern itself, which is achieved using *Simplex Noise* [17] and its transformations using the implemented *fractional Brownian motion* [18].

A simplified schema of camouflage generation algorithm of *Generate()* function is shown in Figure 13. The following parameters were given to the function: the size of created image, the background color, and the layers needed to generate the camouflage. The generation of camouflage begins with the operation of generating the shapes for each color, i.e., layer. This operation consists of two stages: generating a map and extracting the shapes. The map generation function is shown in Algorithm 1 and consists of noise generation operation *SimplexNoise*. This noise is then transformed with function (1), which transforms the shapes generated by noise into the camouflage spots:

$$fbm(fbm(fbm(p + V_1) + V_2) + V_3) \tag{1}$$

where:

$p$—pair $(x; y)$, specifying the coordinates on the two-dimensional array of the camouflage pattern;

$v_1$; $v_2$; $v_3$—appropriately selected vectors, transforming the noise into a camouflage pattern;

$fbm()$—*fractional Brownian motion* function [18].

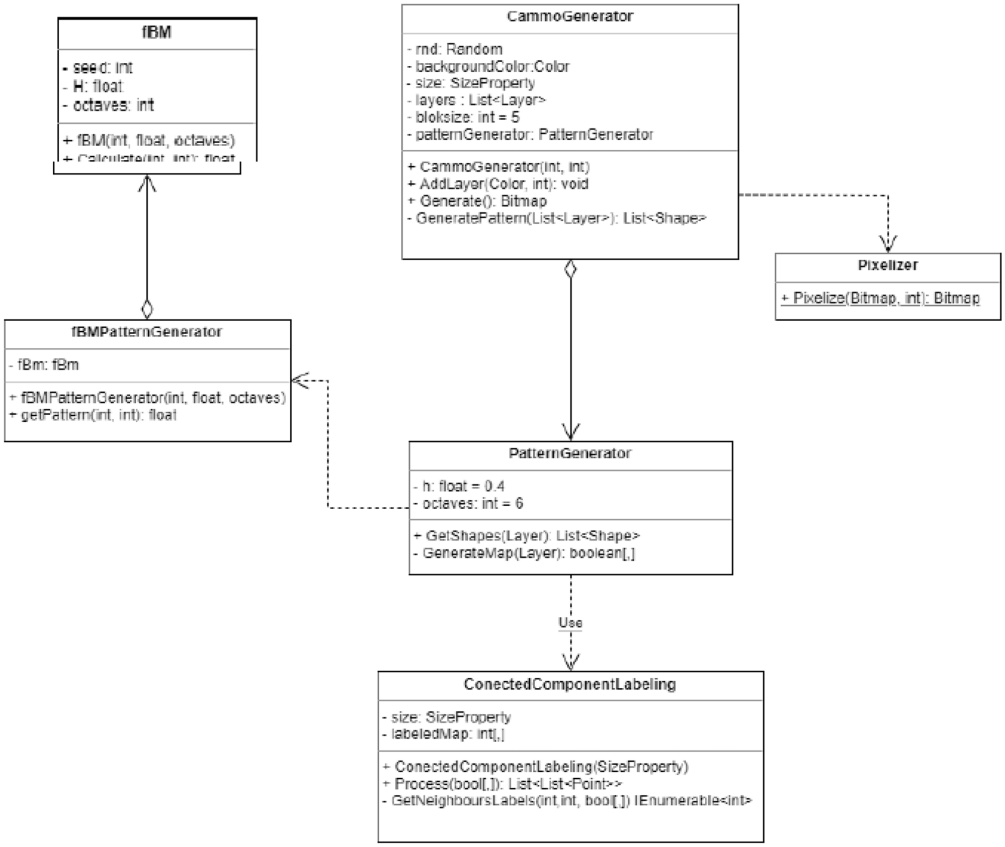


**Figure 12.** Class diagram for *CammoGenerator* component in logic layer of *CammoGenerator* application.

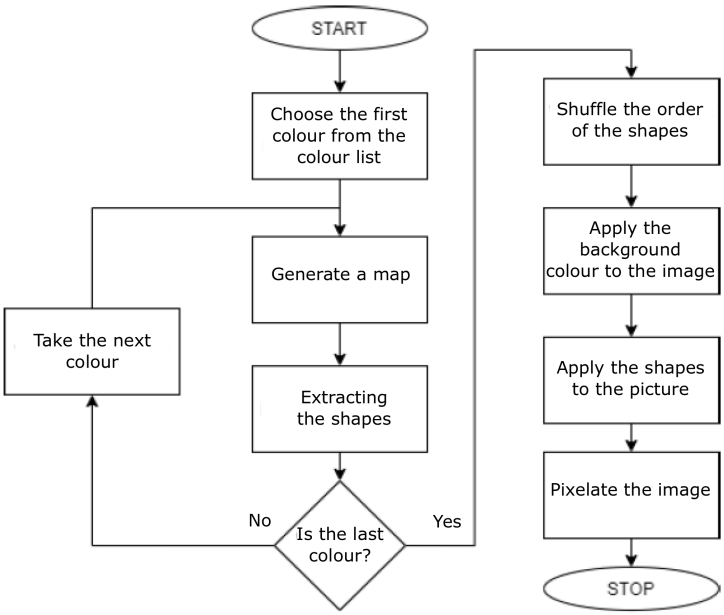


**Figure 13.** Simplified schema of camouflage generation algorithm.

**Algorithm 1:** CammoGenerator component—camouflage map generation function.

```
Function CammoGeneratorComponentFCamouflageMapGeneration(shape_i):
    GenerateSimplexNoise with Given Grain();
    if fbm(fbm(fbm(p + V1) + V2) + V3)%256 < 60 then
        SetValue In [p] array = true;
    else
        SetValue In [p] array = false;
```

The *fbm()* function adds successive less amplitude noise to the base noise while keeping it dependent on the base noise, resulting in increased detail for the base noise. It allows to obtain more varied noise values. This effect is shown in Figure 14. After transforming the noise using function (1), a *modulo 256* operation is performed on the noise values. Depending on whether the value is greater or less than a predetermined threshold, the pattern table for the layer is set to *true* or *false* at the $p$ position.

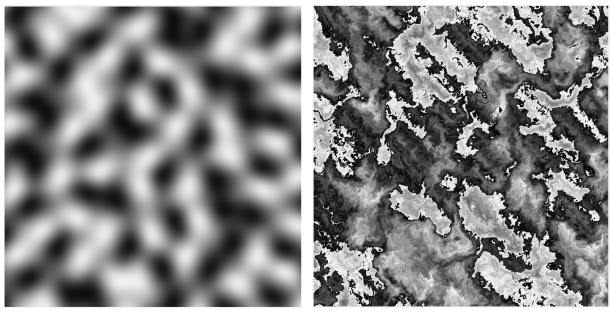

**Figure 14.** Sample SimplexNoise noise (on the **left**) and noise transformed with function (1) (on the **right**).

When the entire pattern table is filled with values, the shapes are created from the *true* values using the algorithm shown in Algorithm 2. This algorithm groups points into shapes by iterating over all *true* values in the pattern table. For a given point in the table, labels of neighbor points are checked. In the case where the neighbors do not have assigned labels, the current value of the label is assigned to the point and the value of the label is increased by one. When a point has at least one neighbor with a label, the smallest neighbor label is assigned to the point and its neighbors' labels, so the joining of labels takes place. An example of the algorithm's operation is presented in Figure 15, where the tested point is marked in red and the neighbors in blue.

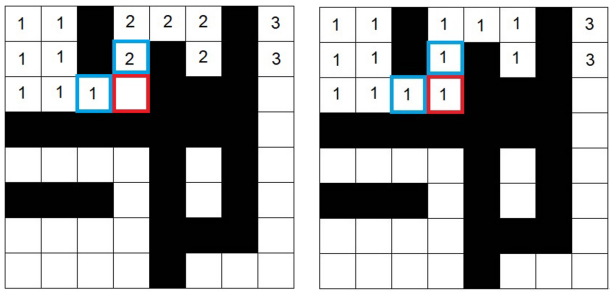


**Figure 15.** Example of shape extraction algorithm (on the **left**) and example of combining labels of shape extraction algorithm (on the **right**).

**Algorithm 2:** CammoGenerator component—shape extraction function.

```
Function CammoGeneratorComponentFShapeExtraction(shape_i):
    label = 1;
    PickPoint();
    CheckNeighboursLabels();
    if LabelList = empty then
        SetLabelApoint();
        label++;
    else
        ChooseTheLowestValueFromLabels();
        AssignNeighborsTheirGroupLabel();
    if there is another point = true then
        ChooseAnotherPoint();
        GoTo CheckNeighboursLabels();
    else
        GroupPointsByLabels();
```

After generating the shapes for each layer, a blank image is generated, over which the background color is applied. The shapes are then superimposed on the image in random order. The overlapping effect was easily achieved by mixing the order in which the stains were applied. The final stage of camouflage generation is its deformation, revealing squares of colors, called pixelation. This effect is achieved by dividing the image into squares of the selected size (for the CammoGenerator application, the square is 5), and then by determining the dominant color for each square. When selected, the squares are filled with their dominant color. The deformation function is shown in Algorithm 3.

**Algorithm 3:** CammoGenerator component—deformation function.

```
Function CammoGeneratorComponentFDeformation(image_i):
    divideImageIntoSquares();
    choose Square;
    while there is another square do
        FindDominantColorInSquare();
        SetColorInSquareToDominantColor();
        choose Another Square;
```

### *4.3. Graphic Layer*

The graphic layer of the *CammoGenerator* application was created using the Windows Forms library. It consists of the *FormMain* main view, which contains two objects: a color palette, which is a list of colors, and an object of the *CammoGenerator* class. The main view has the ability to display three views: Home, Pallete, and Pattern. The *Home* view contains information about the application and instructions for generating the camouflage, *Pallete* is the view for creating and editing the color palette, and *Pattern* is for editing and generating the camouflage. The *FormMain* class gives other views access to the palette and the generator, thanks to which the *Pallete* and *Pattern* views are able to be dynamically replaced without data loss (Figure 16).

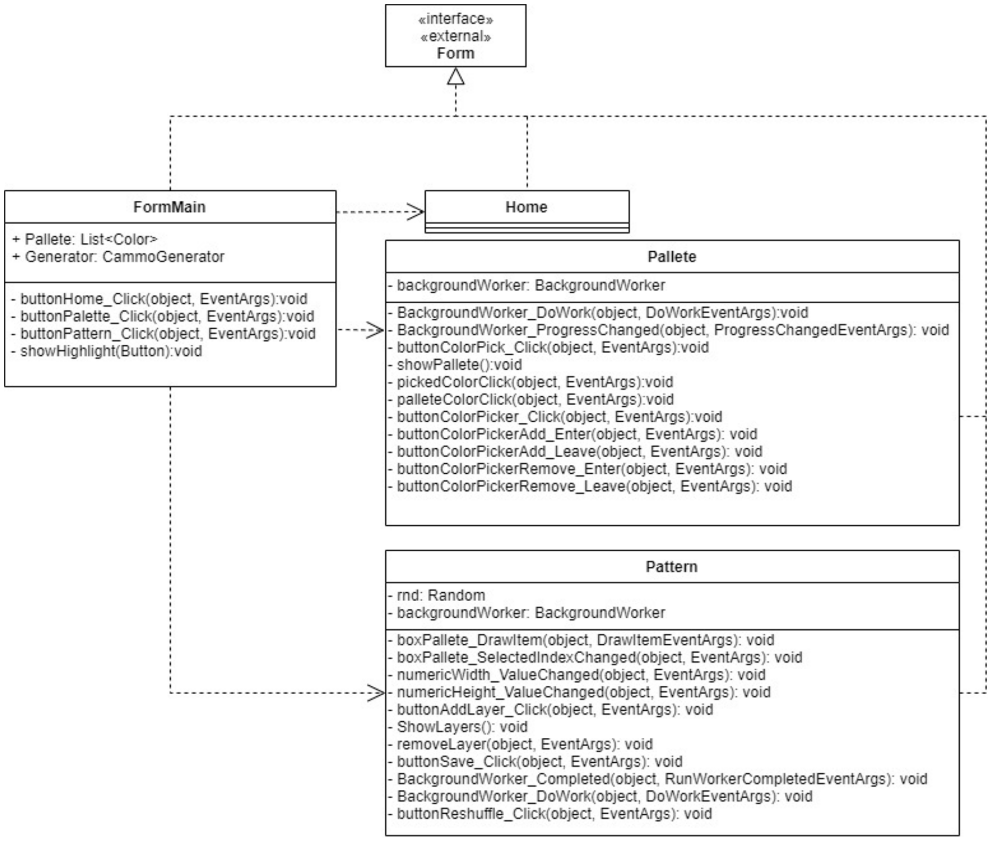


**Figure 16.** Graphic layer class diagram of *CammoGenerator* application.

## 5. Analysis and Verification of Camouflage Generation Algorithm and Application

In order to estimate the quality of generated camouflages and compare them with those generated by other camouflage generators, we used the method of estimating the quality of camouflage based on color difference and size of gradient [19]. This method is shown in Figure 17.

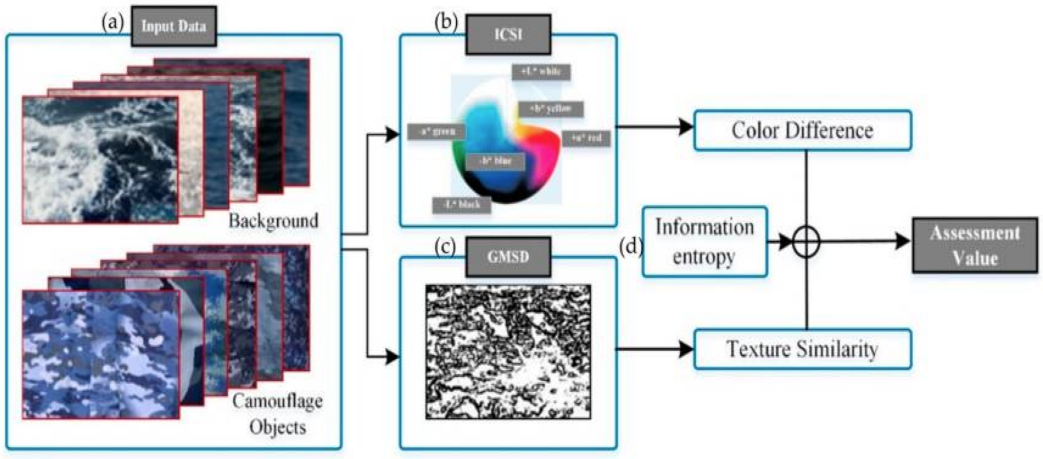


**Figure 17.** Method of estimating the camouflage quality [19].

The method receives as an input (point (a) in Figure 17) a background slice in which the quality of the camouflage is tested, and the tested camouflage itself (both images must be of the same size). Then, the ICSI (image color similarity index) [19] (point (b) in Figure 17) and GMSD (gradient magnitude similarity deviation) [19] (point (c) in Figure 17) values are calculated between the background and the camouflage. We add the obtained results together with appropriate weights denoting the importance of the given information (point

(d) in Figure 17) and we obtain an estimated value of the camouflage quality, whereby the closer it is to the "0" value, the better the quality of the camouflage is. The received value does not have a "1".

Several background scenes (N) and camouflage images (M) are introduced into the camouflage quality estimation method. Color and texture are analyzed using ICSI and GMSD indicators. When calculating ICSI and GMSD, the resolutions of the camouflage and background images are aligned to facilitate comparison. Since ICSI and GMSD are computed for paired images, the $M \times N$ color and texture matrices are obtained separately. The weights of each metric are determined using the information entropy method. The final performance rating for each camouflage pattern is determined by the weighted average of the difference between the camouflage image and all background images.

Three types of terrain were selected for the camouflage analysis: desert, temperate forest, and steppe. For each area, on the basis of digital Earth images, the camouflages were generated using the developed application and previously analyzed applications. In addition, military camouflages intended for operations in the abovementioned areas were analyzed in order to compare the camouflages created by the *CammoGenerator* application. The images used for the analysis (digital Earth and images of the area) were obtained from Google Maps [20]. When performing the calculations, the weight of each of the parameters (ICSI and GMSD) was assumed to be the same, i.e., it amounts to 0:5, and the calculations were performed in Matlab [21]. Figure 18 shows the digital Earth images used to generate the camouflage and the images of the areas from digital Earth photo areas needed for quality analysis.

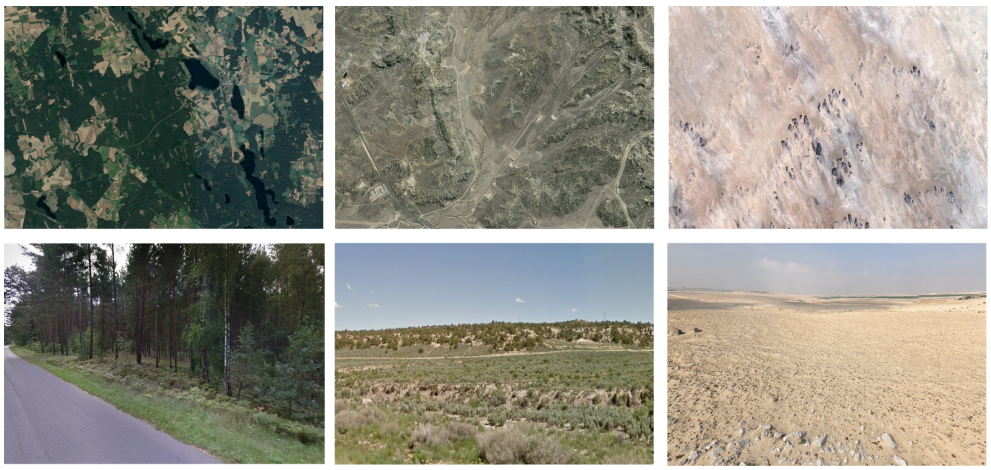

**Figure 18.** Photographs of the terrains (digital Earth and real) used during the camouflage analysis: forest terrain, steppe terrain, and desert terrain.

Figures 19–21 show the camouflages analyzed in these three mentioned areas: forest terrain, steppe terrain, and desert terrain.

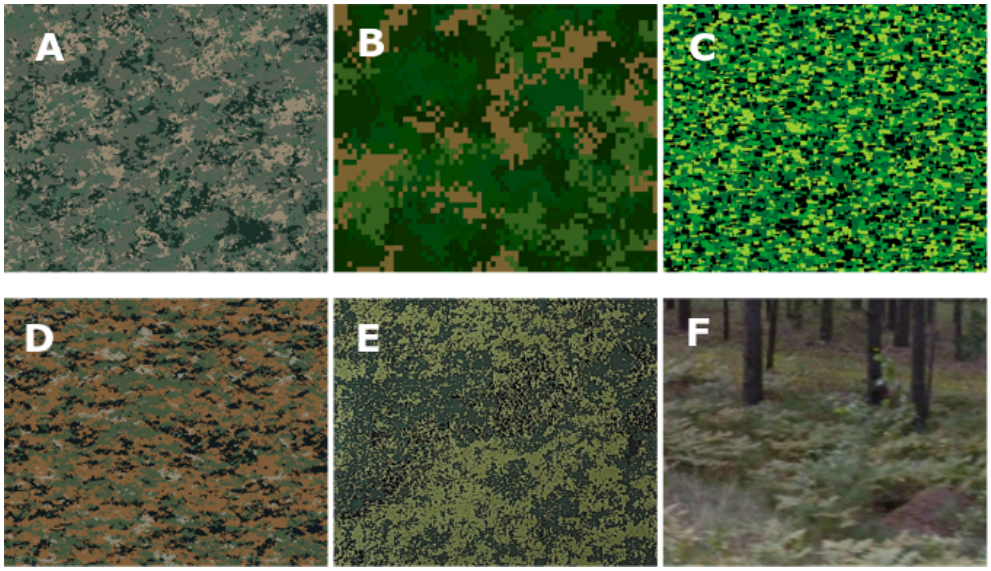


**Figure 19.** Camouflages analyzed in the forest area: (**A**) generated with created *CammoGenerator*, (**B**) generated with *Camouflage Generator* [15], (**C**) generated with *Digital Cammo Generator* [16], (**D**) MARPAT camouflage (Marine pattern) [22], (**E**) EMR camouflage (Edinaya maskrovochnaya rascvetka) [23], (**F**) tested section of terrain.

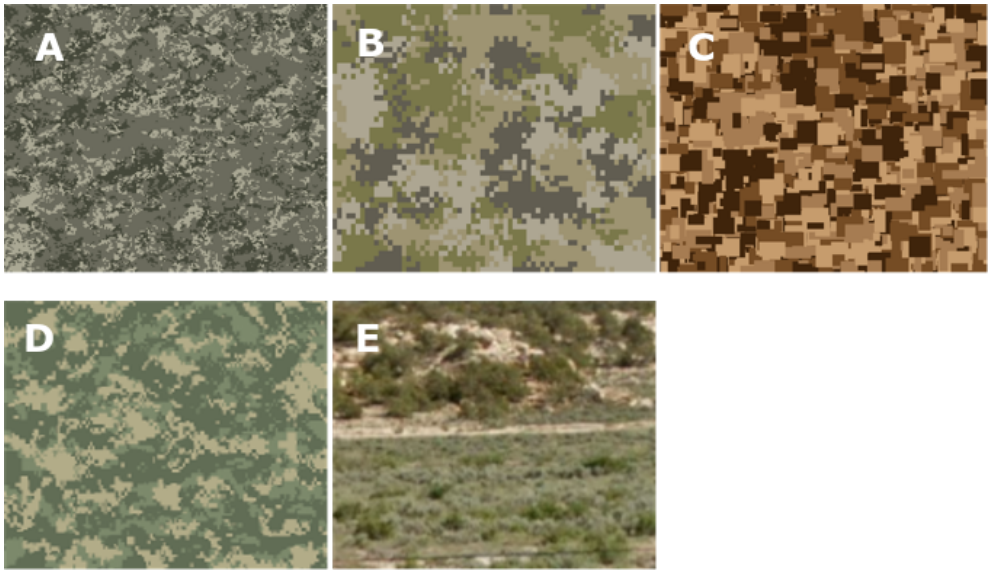


**Figure 20.** Camouflages analyzed in the steppe area: (**A**) generated with created *CammoGenerator*, (**B**) generated with *Camouflage Generator* [15], (**C**) generated with *Digital Cammo Generator* [16], (**D**) UCP camouflage (Universal Camouflage Pattern) [24], (**E**) tested section of terrain.

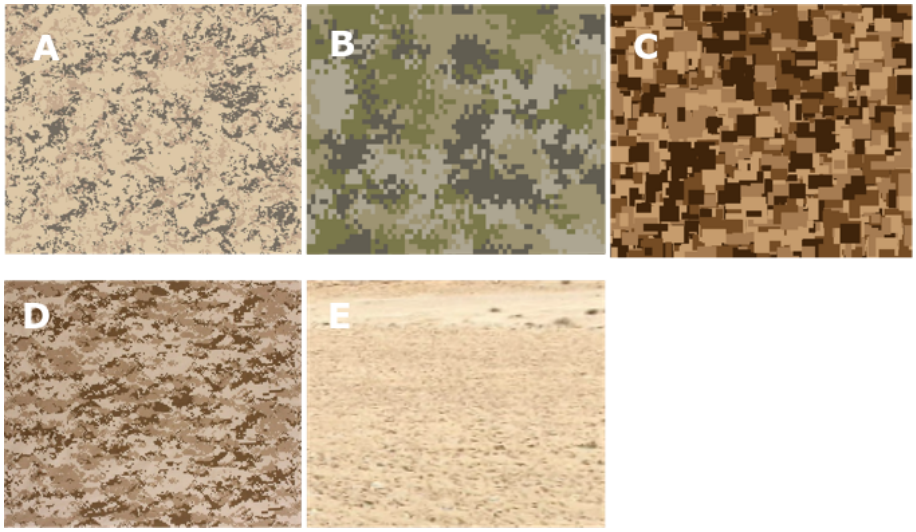


**Figure 21.** Camouflages analyzed in the desert area: (**A**) generated with created *CammoGenerator*, (**B**) generated with *Camouflage Generator* [15], (**C**) generated with *Digital Cammo Generator* [16], (**D**) MARPAT camouflage (Marine pattern) Digital Desert Camouflage [25], (**E**) tested section of terrain.

The camouflages analyzed in selected areas are as follows: (1) generated with created *CammoGenerator* (Figure 22), (2) generated with *Camouflage Generator* [15], (3) generated with *Digital Cammo Generator* [16], (4) MARPAT camouflage (Marine pattern) [22], and (5) EMR camouflage (Edinaya maskrovochnaya rascvetka) [23]. For each camouflage, its quality was estimated on a given image part using the method described above and the results are presented in Table 1.

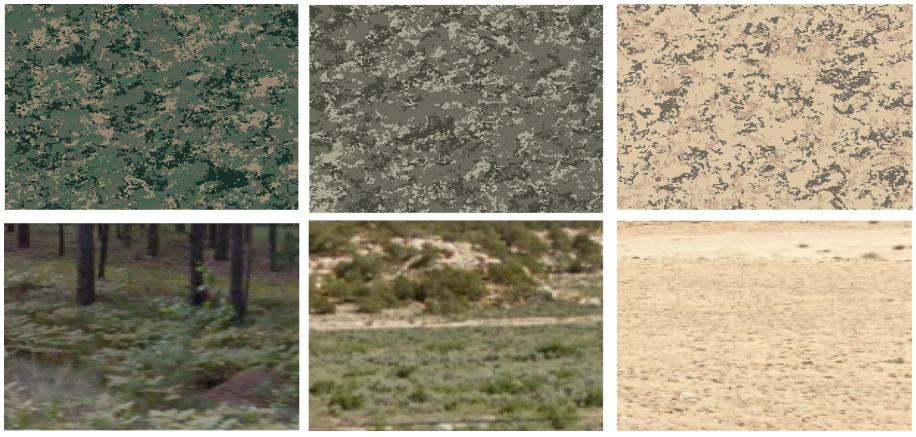

**Figure 22.** Camouflages generated by CammoGenerator application (**first row**) and the real terrains (**second row**) in forest terrain, steppe terrain, and desert terrain.

In all three cases, the camouflage analysis showed that the camouflage generated by *CammoGenerator* outperformed Camouflage Generator and Digital Cammo Generator. Additionally, only during the test in the steppe terrain did it achieve a result slightly worse than the dedicated military camouflage, which was developed for such terrains. In other cases, our camouflage received an estimated quality better than the other tested patterns. This situation is the result of the right choice of colors for each area, which gives an advantage over military camouflages designed to work more universally in areas of a given type, focusing more on a larger area than on a smaller area. The results obtained by the Digital Cammo Generator, as assumed, are the worst compared to the other tested camouflages. This is due to the inability to change the colors and limiting the generated pattern to two defined sets of colors, which does not allow the camouflage to be adapted to the intended area of operation.

**Table 1.** Results of camouflage quality analysis for selected areas.

| | Forest Terrain | Steppe Terrain | Desert Terrain |
|---|---|---|---|
| CammoGenerator | 1339.5 | 1399.6 | 1279 |
| Camouflage Generator | 2649.5 | 1431.5 | 2627.6 |
| Digital Cammo Generator | 4124.8 | 2033.8 | 3398.2 |
| MARPAT camouflage | 1530.3 | 1373.1 | 2032.5 |
| EMR camouflage | 1551.1 | NA | NA |

## 6. Discussion and Conclusions

An analysis of the camouflage generation software revealed a very small number of them. Moreover, they are mostly private solutions, which are simple tools that do not generate effective camouflage patterns. Such generators do not meet the requirements of users who would actually like to use such generated camouflages. In addition, no solution provides the functions that facilitate the selection of colors, which makes it difficult to create a camouflage for users unfamiliar with this subject and significantly extends the time required to create a good pattern.

The created algorithm and the CammoGenerator application built on its basis was designed to simplify and accelerate the color selection stage, as well as create good-quality camouflages. The solution was created in a three-tier architecture, which allows for any modifications in each of the layers without affecting the operation of the other layers. Additionally, in the logic layer, each function, i.e., color extraction and camouflage generation, was implemented as separate components, which makes it possible to change the algorithms of each component separately.

The results of the solution's operation were examined and compared with other selected generators, as well as selected used military camouflages adapted to the tested areas. The results prove that the proposed approach creates the camouflages qualitatively better than those generated by generally available generators. In addition, it generates camouflages qualitatively similar to, or even better than, those developed and used by the military, which was also supported by the results of the analysis, where the results obtained by our camouflages were similar or better. The obtained results prove that the proposed approach easily generates good-quality camouflages based on digital Earth images. The CammoGenerator application solves the problems of tested existing generators and provides users with a simple and effective tool for generating camouflage.

The proposed approach, together with the designed algorithms and developed CammoGenerator application, meets the assumed requirements; however, its operation can be improved, which could raise the quality of generated camouflages and add more editing possibilities.

The first element that can be improved is color extraction. The current color extraction algorithm uses the centroid algorithm and the CIELab color space. Such a solution when analyzing large images is time-consuming and does not always give satisfactory results. The proposed solution is to create a neural network and teach it to pick the right colors from the given digital Earth images.

The neural network can also be used to develop an algorithm for generating camouflage patterns. Such a network can extract shapes that occur naturally in the photo, and then compose patterns from them. Thanks to this solution, the sizes and shapes of the spots will be adapted to the conditions in the given area and will not be limited to the ones implemented in the algorithm.

Another element related to the development of the proposed generator may be the development of a new method for comparing the quality of the camouflage, which will allow for a more accurate study of the generator's performance. Since the method used to test the quality did not take into account factors such as lighting, shadow, and the texture of the object on which the camouflage is applied, a new method can be developed that

will allow mathematically estimating the quality of the pattern, taking into account the aforementioned factors.

**Author Contributions:** Conceptualization, M.S. and A.P.M.; methodology, M.S. and A.P.M.; validation, A.P.M. and K.S.; formal analysis, M.S., A.P.M. and K.S.; investigation, M.S.; resources, M.S.; data curation, M.S.; writing—original draft preparation, M.S., A.P.M. and K.S.; writing—review and editing, A.P.M. and K.S.; visualization, A.P.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Kstrom, M.P. *Digital Image Processing Techniques*; Academic Press: Cambridge, MA, USA, 2012; Volume 2.
2. Umbaugh, S.E. *Computer Imaging: Digital Image Analysis and Processing*; CRC Press: Boca Raton, FL, USA, 2005.
3. Riber-Hansen, R.; Vainer, V.B.; Steinche, T.; Digital image analysis: A review of reproducibility, stability and basic requirements for optimal results. *Apmis* **2012**, *120*, 276–289. [CrossRef] [PubMed]
4. Pettersson, R. Visual camouflage. *J. Vis. Lit.* **2018**, *37*, 181–194. [CrossRef]
5. Newark, T. *The Little Book of Camouflage*; Osprey Publishing: Oxford, UK, 2013; ISBN: 978-1-78200-831-6.
6. Foster H.-F. *Fm5-20 Camouflage, Basic Principles*; War Department: Washington, DC, USA,1944.
7. Patton, N.; Aslam, T.M.; MacGillivray, T.; Deary, I.J.; Dhillon, B.; Eikelboom, R.H.; Constable, I.J. Retinal image analysis: Concepts, applications and potential. *Prog. Retin. Eye Res.* **2006**, *25*, 99–127. [CrossRef] [PubMed]
8. Lin, C.J.; Chang, C.C.; Lee, Y.H. Developing a similarity index for static camouflaged target detection. *Imaging Sci. J.* **2013**, *62*, 337–341. [CrossRef]
9. Lin, C.J.; Prasetyo, Y.T.; Siswanto, N.D.; Jiang, B.C. Optimization of color design for military camouflage in CIELAB color space. *Color Res. Appl.* **2019**, *44*, 367–380. [CrossRef]
10. Patil, K.V.; Pawar, K.N. Method for improving camouflage image quality using texture analysis. *Int. J. Comput. Appl.* **2017**, *180*, 6–8.
11. Volonakis, T.N.; Matthews, O.E.; Liggins, E.; Baddeley, R.J.; Scott-Samuel, N.E.; Cuthill, I.C. Camouflage assessment: Machine and human. *Comput. Ind.* **2018**, *99*, 173–182. [CrossRef]
12. Lin, C.J.; Chang, C.C.; Liu, B.S. Developing and evaluating a target background similarity metric for camouflage detection. *PLoS ONE* **2014**, *9*, e87310. [CrossRef]
13. Morin, S.A.; Shepherd, R.F.; Kwok, S.W.; Stokes, A.A.; Nemiroski, A.; Whitesides, G.M. Camouflage and display for soft machines. *Science* **2012**, *337*, 828–832. [CrossRef]
14. Nyberg, S.; Bohman, L. Assessing camouflage methods using textural features. *Opt. Eng.* **2001**, *40*, 60–71. [CrossRef]
15. Astrom, U. Camouflage Generator. Available online: http://www.happyponyland.net/camogen.php (accessed on 10 September 2022).
16. Chandler, A. Digital Camo Generator. Available online: http://cowdd.com/game/canvas/index.php (accessed on 10 September 2022).
17. Gustavson, S. Simplex Noise Demystified. 2005. Available online: https://weber.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf (accessed on 10 July 2022).
18. Shevchenko, G. Fractional Brownian motion in a nutshell. *Int. J. Mod. Phys. Conf. Ser.* **2015**, *36*, 1560002. [CrossRef]
19. Bai, X.; Liao, N.; ; Wu, W. Assessment of Camouflage Effectiveness Based on Perceived Color Difference and Gradient Magnitude. *Sensors* **2020**, *20*, 4672. [CrossRef]
20. Google Maps. Available online: https://www.google.com/maps (accessed on 10 September 2022).
21. Matlab. Available online: https://www.mathworks.com/products/matlab.html (accessed on 10 September 2022).
22. MARPAT Camouflage. Available online: https://pl.pinterest.com/pin/802555596073688790/ (accessed on 10 September 2022).
23. EMR Camouflage. Available online: https://militaristwear.com/image/cache/data/05/3339298-600x600.jpg (accessed on 10 September 2022).
24. CP Camouflage. Available online: https://www.deviantart.com/bradvickers/art/Camouflage-United-States-UCP-603194107 (accessed on 10 September 2022).
25. DDC Camouflage. Available online: https://www.walmart.com/ip/Rothco-Large-Digital-Camo-Bandana-27-x-27-Desert-Digital-Camo/104367138 (accessed on 10 September 2022).