**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 10, No. 6, December 2020, pp. 5951∼5964
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp5951-5964 

# An effective RGB color selection for complex 3D object structure in scene graph systems

**Chung Le Van[1], Gia Nhu Nguyen[2], Tri Huu Nguyen[3], Tung Sanh Nguyen[4], Dac-Nhuong Le[5]**
[1,2]Duy Tan University, Vietnam
[3,4]Hue University of Medicine and Pharmacy, Vietnam
[5]Haiphong University, Vietnam

**Article Info**

***Article history:***

Received Feb 5, 2020
Revised May 13, 2020
Accepted May 25, 2020

***Keywords:***

3D object
Alpha channel
Color map
Color selection
Map mask
Virtual reality

**ABSTRACT**

Our goal of the project is to develop a complete, fully detailed 3D interactive model of the human body and systems in the human body, and allow the user to interacts in 3D with all the elements of that system, to teach students about human anatomy. Some organs, which contain a lot of details about a particular anatomy, need to be accurately and fully described in minute detail, such as the brain, lungs, liver and heart. These organs are need have all the detailed descriptions of the medical information needed to learn how to do surgery on them, and should allow the user to add careful and precise marking to indicate the operative landmarks on the surgery location. Adding so many different items of information is challenging when the area to which the information needs to be attached is very detailed and overlaps with all kinds of other medical information related to the area. Existing methods to tag areas was not allowing us sufficient locations to attach the information to. Our solution combines a variety of tagging methods, which use the marking method by selecting the RGB color area that is drawn in the texture, on the complex 3D object structure. Then, it relies on those RGB color codes to tag IDs and create relational tables that store the related information about the specific areas of the anatomy. With this method of marking, it is possible to use the entire set of color values (R, G, B) to identify a set of anatomic regions, and this also makes it possible to define multiple overlapping regions.



***Corresponding Author:***

Dac-Nhuong Le,
Haiphong University,
171 Phan Dang Luu, Kien An, Haiphong, Vietnam.
Email: nhuongld@dhhp.edu.vn

## 1. INTRODUCTION

2-D virtualization plays a crucial role in the modern up to date 3-D rasterization. With the aid of 2-D algorithms, rendering can be employed on images of texture and object animations. Textures can be classified into two parts: 1) transparent area on images 2) non-transparent area on the images based on the color channel information. Image without transparent area doesn't have any transparent pixel ("*holes*") because it consists only of RGB color component as well as high pixels' alpha value. This particular is highly beneficial to govern the display logic that means any two objects can be merged with each other without affecting the color pixels. The rendering process will be notably simple and quicker. In the case of non-transparent images, rasterization is quite effortless due to the handling of entire texture via blocks instead of the pixel by pixel. The memory copy operation is used to transfer the memory array of texture into the frame buffer. There is only one norm exist: to evade framebuffer during addressing, attentiveness should be paid to the screen edges. Moreover, based on object position, data should be segmented at the time of copying. A viewport culling is executed row by row at the image texture if there is an existence of any object in any direction out of the screen bounding

layer. Although this process requires further processing but the solution is still rapid enough. So this method was usually popular and preferred in the traditional computer games [1-3].

In the transparent texture, the portion of the image consists of a group of transparent points or pixels of an image that are blurred in some intensity. For the implementation of the first category, colorkey, an unused preselected color is employed to highlight transparent pixel. Nowadays, this is attained with the use of alpha channels associated with the image. The demand for this type of image texture is increased and utilized in many application areas to enhance the experience of visualization. To handle this kind of information is not much complicated but it required high computerization because the variation of the transparent and non-transparent area in the texture image that leads rendering process is done at per-pixel level [4-6]. The graphical objects are assessed pixel by pixel through the graphical engine and create the final image see in Figure 1. The main drawback is that it requires high computation due to per-pixel drawing and calling thousands of functions. For every pixel, color information is fetched from the memory and its necessarily color information should already exist on the memory buffer. This technique is not able to fulfill the demand of today's complex games, where up to a hundred different dynamic objects are drawn on one screen simultaneously [6-9].

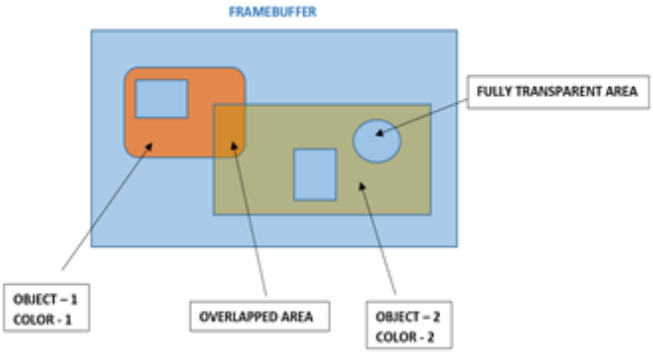



Figure 1. Overlapping RGBA textures

The paper proposed an effective RGB color selection for complex 3D object structure in scene graph systems. Our solution combines a variety of tagging methods, which use the marking method by selecting the RGB color area that is drawn in the texture, on the complex 3D object structure. The article is structured as follows: Section 2 introduces the related works of RGB color selection approaches for complex 3D object structure. Section 3 presents the method proposed and Section 4 implemented our research method and algorhythms. Section 5 illustrates experimental analyze and evaluate the performance. Section 6 conclude the our proposed system and discussed some approaches for the future.

## 2. RELATED WORKS

Many researchers have done some studies related to generate the 3D models from photographs and attract computer vision, photogrammetry, and photographic communities to enhance their application. The focus of this research in the computer vision community is based on recovering camera parameters and surface geometry. Balazs, A. *et al.* presented a novel solution to the gap problem. Vertex programs are used to generate approximately shaded fat borders that fill the gaps between neighboring patched [8]. In [9], Baumberg presented a unique autonomous approach that can help to extend a traditional two-dimensional image blending method to a three-dimensional surface, which generates good quality results at the lowest computational price. This study illustrates an unprecedented system to create texture maps especially for a surface of arbitrary topology from a natural object image clicked with the normal camera and unmanageable light. In recent years, advanced studies on image and complex objects, computer vision methods play a game-changer role to replace the non-automatic techniques of 2D images analysis. In [10], Liu et al. proposed an autonomous object segmentation through the use of RGB-D cameras. The authors developed a tri-state filter that includes boundary information, the value of RGB data, and distance. Richtsfeld et al. implemented and explained the coordination between the patches on the 3D surface image based on the Gestalt principle to construct a learning-based structure [11]. Yalic, H. Y. and Can, A. B. [12] presented a method for autonomous object segmentation on RGB-D

image with the use of surface normal and similarities in the region. In [13], AL-Mousa *et al.* employed a method to enhance the RGB color image encryption-decryption technique through the use of encryption square key combined with a dimensional matrix.

The author in [14] presented a high-rich annotated, large scale repository of 3D models that provide rigid alignments, and bilateral symmetry planes. Moreover, it used deep learning to encrypt the shape priors of a particular category. In [15], a three-dimensional RCNN model is used to generate the three-dimensional object shapes in the form of a volumetric presentation. H Fan et al, proposed a network called point set generation network that can help to originate the three-dimensional shapes in point cloud from two-dimensional images [16]. Many researchers worked on 3D shapes in volumetric representation without deep learning and deep learning-based approaches. There are some exceptions such as point clouds shapes at the time of generation [17], cuboid primitive, and numerous surfaces [18-23]. In Chengjie Niu *et al.* presented a technique to retrieve three-dimensional contour structure from a mono RGB image, structure means cuboid represented shape parts and parts connection surrounding connectivity and symmetry [24]. A technique of autonomous recognition of structural elements from a photographic camera picture of the construction area [25, 26]. A 2D algorithmic program is being proposed by E. Trucco and A. P kaka [27] to recognize the objects and structures in the construction site image. To detect the column in a given image, an object recognition technique was developed that train classifiers on 100 sets of images of the concrete column [28]. In [29], the authors developed a technique for comparison of the final construction schedule with the updated progress of work done at the construction site. Podbreznik and Rebolj [30] presented an architecture that used to reconstruct the three-dimensional geometric models from a two-dimensional image of the construction area. In [31], Hyojoo Son and Changwan Kim presented a highly productive, autonomous three-dimensional structural element recognition and modeling technique that uses color and three-dimensional data obtained from a stereo vision system deployed on the construction area for monitoring purpose.

## 3. OUR METHODOLOGY

On a 3D anatomical object [32-35], for instance the brain, there are many anatomic landmarks that need to be marked with information for the student, such as pin marking, direct drawing, or selecting the entire brain to see detailed information as shown in Figure 2(a). But there are also many grooves, and areas of surgical information, which may overlap, so the problem is the way to mark them to show all these. Here, we propose the use of a method to plot anatomy on individual textures, based on the RGB color code of the region in its true UV position [7-9, 36]. Our method can be compared with the method of selecting PIN, but this only focuses on a certain point on the 3D object. A second method uses marks by an alpha channel in the texture of the object, which means that the number of selection points is limited to a value between 0 to 255 only as shown in Figure 2(b)) [9, 37, 38].

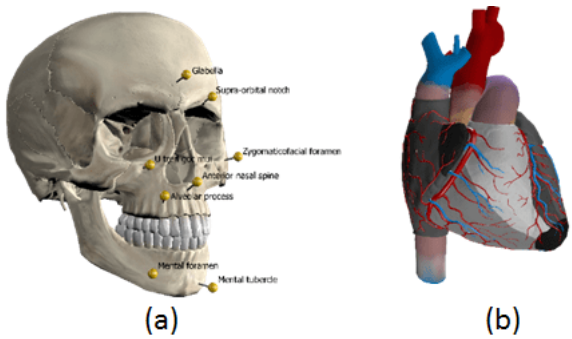



Figure 2. (a) The skull is marked by the PIN and (b) The heart with some regions is marked with alpha channel

### 3.1. Challenges

The requirements for a solution to our problem are as follows:

(a) The colored area for the surgical point of interest; must be able to select only this part.
(b) The clearer the boundaries of the region, the more accurate the pixel density must be.
(c) Overlapping boundaries or blurred boundaries.

(d) Handles interaction with parallax fragment so that the user interface recognizes the selection.

(e) Handles storage and connects to the database accurately to retrieve more relevant regional information.

### 3.2. Methodology

When using the RGB region selection method, we can mark a multitude of regions, as it is a combination of RGB color values, and it still identifies regions that can overlap, which provides the required viewer effects [2]. The result is that it is possible to mark and select areas and areas that overlap each other. This provides us with a very efficient solution to our problem. Figure 3 presents our process of achieving this solution is described in 6 steps. The following steps will explain further details in the implementation.

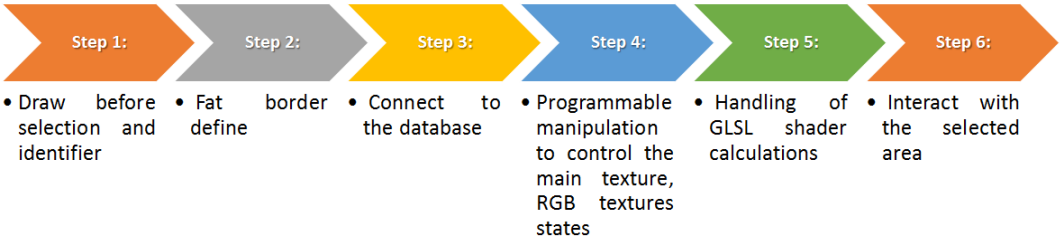



Figure 3. An effective RGB color selection for complex 3D object structure

## 4. OUR IMPLEMENTATION

### 4.1. Draw before selection and identifier

Draw an area and paint the area on a texture, based on the UV position created from the texture map and normal map . RGB color codes will be remembered as $(R, G, B)$ or hexa. In a texture, all RGB color codes of the regions must be different. Two colors $(0, 0, 0)$ and $(1, 1, 1)$ should be removed from the texture, since they are used to show select and deselect the 3D object [8, 9]. Figure 4 shows the draw color areas based on accurate UV.

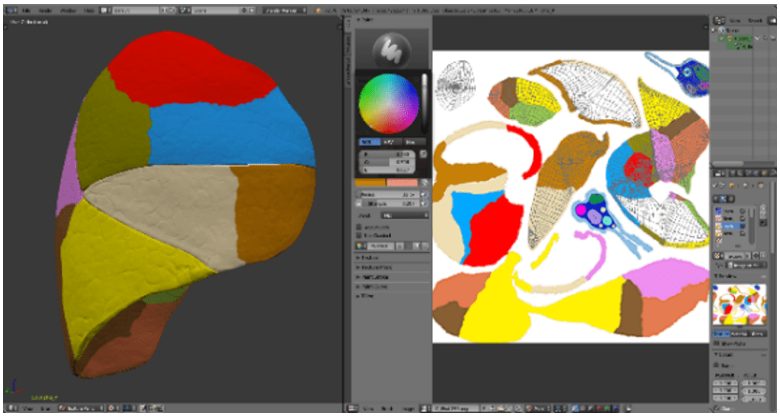


Figure 4. Draw color areas based on accurate UV

### 4.2. Fat border define

In this step, the space between the adjacent patches instigated by the totally non depending triangulations that are fill-up with shaded fat borders. Number of triangle are connected to each other in these borders and their parameters such as orientation, width and colors are view-dependent and analysis each frame through the use of vertex program. Gap filling algorithm input contains $N$ number of level of details (LOD)-sets sets $H_i = \{\hat{M}_i = M_{n_i}, \ldots, M_{n_i}\}$, $i = 1, .., N$ independent patches $\hat{M}_i$ with border.

There is only one condition on these LOD set is that to choose a LOD $M_{k_i}$, as per the $\hat{M}_i$, such that distance between the approximate surface $(M_{k_i})$ and original surface $(\hat{M}_i)$. At the time of projected on the screen, eimg pixel is greater than everywhere, particularly along the patch border [8]. This loophole can be

eliminate by the condition $H(M_i; M_{k_i}) \leq r$ holds for the Hausdorff distance *H*, where the value of *r* is choose in such a way that $e_{img}$ pixel is always higher than screen-space projection of sphere with radius *r*.

This algorithm required set of polylines each depict a boundary curve as an input. For each segment of the line of the narrow path direction The visual direction is constructed by expanding the line segment in such a way that the image spatial projection maximizes the portion of the $\epsilon_{img}$ pixel predicted by each side, as shown in Figure 5 [8]. To set up the newly introduced triangle is exactly the same as the edges of the actual original tool in the string using the newly generated vertices.

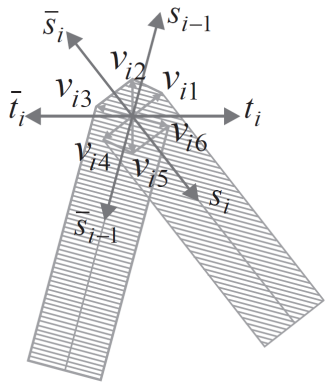



Figure 5. Idea of the vertex program. Vertices are moved to construct a fat border

Vertices of the polylines are iterated according to the algorithms [8, 10] and each polyline are processed in a particular manner as follow:

(a) To calculate the normalized orientations of the surface and its perpendicular of the previous respective i.e. $S_{i-1}$ and $S_i$ and this line segments at the latest vertex $V_i$ together with their dispute counterpart $\overline{s_{i-1}} = -S_{i-1}$ and $\overline{s_i} = -S_i$, respectively.

(b) To calculate the normalized tangent $T_i = \frac{S_{i-1}+S_i}{\|S_{i-1}+S_i\|}$ poly lines at the latest vertex $V_i$ together with counterpart $\overline{T_i} = -\overline{T_i}$.

(c) To generate six new vertices by supplant $T_i$ perpendicular to every direction enumerated in above mentioned steps and viewing direction $D_i = \frac{C-V_i}{\|C-V_i\|}$ (see Figure 6), where *C* is the location of the camera

$$\begin{aligned} V_{i1} &= \varepsilon \tfrac{(S_i \times D_i)}{\|(S_i \times D_i)\|}; V_{i2} = \varepsilon \tfrac{(T_i \times D_i)}{\|(T_i \times D_i)\|}; \\ V_{i3} &= \varepsilon \tfrac{(S_{i-1} \times D_i)}{\|(S_{i-1} \times D_i)\|}; V_{i4} = \varepsilon \tfrac{\left(\overline{S_i} \times D_i\right)}{\left\|\left(\overline{S_i} \times D_i\right)\right\|}; \\ V_{i5} &= \varepsilon \tfrac{\left(\overline{T_i} \times D_i\right)}{\left\|\left(\overline{T_i} \times D_i\right)\right\|}; V_{i6} = \varepsilon \tfrac{\left(\overline{S_{i-1}} \times D_i\right)}{\left\|\left(\overline{S_{i-1}} \times D_i\right)\right\|}; \end{aligned} \tag{1}$$

where $\epsilon$ is the object space geometric error guaranteeing a screen space error of $\epsilon_{img}$ pixel.

(d) To push the recently developed vertices apart from the viewer together with the viewing direction again by $\epsilon$ .

(e) To generate new triangles by attaching the resulting vertices (see in Figure 5). A single quad strip can be explained for every boundary curve because of fat border simple structure.

(f) To calculate the color of every latest vertices from allocating the shading parameter of the native border vertices. The orientation of the fat borders provide gap filling service, and they don't affect on the original shading.

It is required to continuously update the position of new verticles because of dynamic nature of viewing direction i.e. pixel to pixel changes.

**Algorithm 1** . Vertex program to render fat borders. The tangent vectors are stored as texture coordinates and the approximation error is given as local program parameter.

```
BEGIN
    uniform vec3 length; vec4 construct fat border(); vec3 view, offset; vec4 pos;
        view = normalize(vec3(-1,-1,-1) × gl NormalMatrix[2]);
        pos = gl Position;
        offset = normalize(cross(view,gl MultiTexCoord0.xyz));
        offset += gl MultiTexCoord0.xyz; pos.xyz += length × offset;
    return pos;
END
```

Although six points are necessary to confirm the enlargement of the border even at sharp corners, usually in real-time only 4 or 2 new vertices are enough to deliver the good results and have a vast impact on the rendering performance. The relative fat border origination schemes for 2, 4, and 6 vertices are discussed in Figure 6. In this case, we employ four vertices, $V_{i2}$ and $V_{i5}$ vertices in equation 1 are ruled out, if two vertices are employed only $V_{i2}$ and $V_{i5}$ are originated. Also note that the fat border for the patch is directly proportional to the level of the detail (LOD) level, if there is no change on the LOD level, the fat border will also not change. We can make use of this property to enclose the fat border in the display list and thus remove the requirement of sending this data over to computer graphic hardware in every frame, which makes the fat border practically requirement is equal to zero [8]. Figure 6 represents the different fat border origination algorithm.

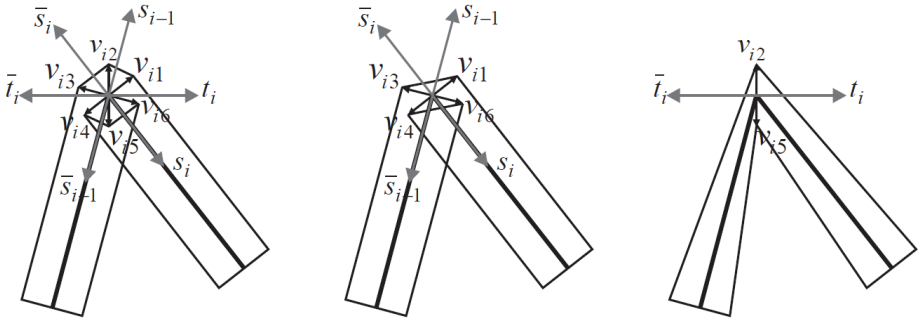



Figure 6. New vertices from left to right 6, 4, and 2 (Algorithm for the fat border origination)

The only pre-essential for this is to issue six dummy vertices and their mutual connection for every border vertex. The vertex program only runs for the border vertices and its run is stopped while the paths are rendering itself. Figure 7 represents the whole process of it.

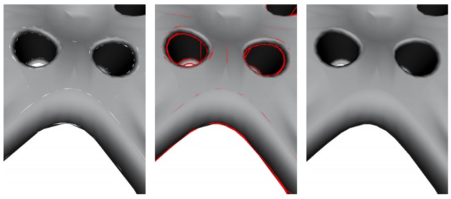
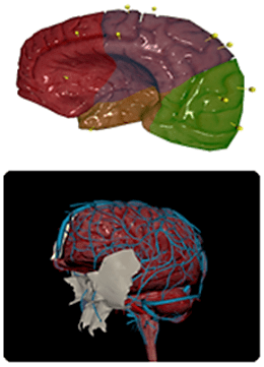


Figure 7. Rendered wheel rim model a) without fat borders b) fat border imposed c) final output: gaps covered by fat border

### 4.3. Connect to the database

For each RGB color code that has been created, it should be numbered and named after the name of the anatomy, and also store many data fields to contain the relevant information of the anatomy (e.g., names of the parts, names in different languages (*currently Vietnamese, Latin and English*), detailed description of anatomy, RGB color code, etc.). Create relational data table (*using MySQL database system*), to later query the information. Create tables to save the path to get the texture matching the required 3D object as shown in Figure 8.



| ColorID | Name | ENName | SName | ColorValue |
|---|---|---|---|---|
| 1 | 3_brain_hoi_tran_tren_nao_R | Superior frontal gyrus | Gyrus frontalis superioris | 255 0 0 |
| 2 | 3_brain_tran_trong | Media frontal gyrus | Gyrus frontalis medialis | 114 231 229 |
| 3 | 3_brain_cuc_tran_nao_R | Fronta pole | Polus frontalis | 193 0 255 |
| 4 | 3_brain_ranh_tran_tren_nao_R | Superior frontal sulcus of the brain | Sulcus cerebri frontalis superior | 0 194 168 |
| 5 | 3_brain_hoi_tran_giua | Middle frontal gyrus | Gyrus frontalis medius | 255 255 0 |
| 6 | 3_brain_hoi_tran_duoi_nao_R | Inferior frontal gyrus | Gyrus frontalis inferior | 141 173 217 |
| 7 | 3_brain_phan_nap_nao_R | Opercular part | Pars opercularis | 217 210 140 |
| 8 | 3_brain_phan_tam_giac_nao_R | Triangular part | Pars triangularis | 0 43 137 |
| 9 | 3_brain_phan_o_mat_nao_R | Orbital part | Pars orbitalis | 0 231 79 |
| 10 | 3_brain_cac_hoi_o_o_mat_nao_R | Orbital circonvolution | Gyrus occipitalis | 206 96 180 |
| 11 | 3_brain_hoi_thai-duong_tren | Superior temporal gyrus | Gyrus temporalis superior | 144 114 49 |
| 12 | 3_brain_hoi_thai_duong_giua_na... | Middle temporal gyrus | Gyrus temporalis medius | 234 255 40 |
| 13 | 3_brain_hoi_thai_duong_duoi_na... | Inferior temporal gyrus | Gyrus temporalis inferior | 210 53 53 |
| 14 | 3_brain_hoi_truoc_trung_tam_n... | Anterior central gyrus (Precentral... | Gyrus precentralis (Gyrus prae... | 56 45 104 |
| 15 | 3_brain_hoi_sau_trung_tam_nao_R | Ascending parietal gyrus (Postcen... | Gyrus postcentralis | 42 135 184 |
| 17 | 3_brain_hoi_tren_vien_nao_R | Supramarginal gyrus | Gyrus supramarginalis | 137 173 36 |
| 18 | 3_brain_moc_nao_R | Uncus, hook | Uncus | 21 83 161 |
| 19 | 3_brain_hoi_goc_nao_R | Angular gyrus | Gyrus angularis | 75 180 189 |
| 20 | 3_brain_cuc_cham_nao_R | Occipital pole | Polus occipitalis | 99 78 16 |
| 21 | 3_brain_ranh_sau_trung_tam_n... | Ascending parietal sulcus | Sulcus postcentralis | 102 99 2 |
| 22 | 3_brain_hoi_chem_nao_R | Cuneus | Cuneus | 242 186 252 |
| 23 | 3_brain_hoi_truoc_chem_nao_R | Precuneus (quadrate lobule) | Precuneus | 226 114 63 |
| 24 | 3_brain_ranh_dai_nao_R | Cingulate sulcus | Sulcus cinguli | 58 203 4 |
| 25 | 3_brain_tieu_thuy_canh_trung_t... | Paracentral lobule | Lobulus paracentralis | 225 210 138 |
| 26 | 3_brain_ranh_trung_tam_nao_R | Precentral sulcus of the brain | Sulcus cerebri centralis anteri... | 186 254 27 |
| 27 | 3_brain_hoi_thang_nao_R | Straight circonvolution | Gyrus rectus | 239 183 21 |
| 28 | 3_brain_cuc_thai_duong_nao_R | Temporal pole | Polus temporalis | 206 96 180 |
| 29 | 3_brain_cac_ranh_o_mat_nao_R | Orbital sulcus | Sulcus occipitalis | 255 51 0 |
| 30 | 3_brain_ranh_ben_sylvius_nhan... | Lateral sulcus anterior ramus | Sulcus lateralis ramus anterior | 255 9 251 |
| 31 | 3_brain_ranh_ben_sylvius_nhan... | Lateral sulcus ascending ramus | Sulcus lateralis ramus ascendens | 0 43 137 |
| 32 | 3_brain_ranh_ben_sylvius_nhan... | Lateral sulcus posterior ramus | Sulcus lateralis ramus posterior | 0 248 185 |

Figure 8. The storage of color code and relational values

### 4.4. Programmable manipulation to control the main texture, RGB textures states

It is recommended to have the main texture always load on a certain channel (e.g., index 1). RGB textures should be on channel 0, and load only one RGB texture at a time, then load other RGB texture changes if needed. We use high pixel density of $2048 \times 2048$ pixels to create beautiful color selections, and the edges of the region are less distorted [37].

### 4.5. Handling of GLSL shader calculations

Load main surface texture and RGB texture into uniform sampler2D [38]. Use uniform `vec3` variable $rgb = vec3(1, 1, 1)$. To signal status when RGB texture is activated. Use color mixing with the `mix function (colorRGB, color, 0.65)`.

**Algorithm 2** . Mix Function (colorRGB, color, 0.65).

```
BEGIN
    if(rgb!=vec3(1,1,1)) gl_FragColor.rgb = mix(color, colorRGB, 0.15);
    else gl_FragColor.rgb = color;
    if (colorRGB.rgb = = (rgb)) gl_FragColor.rgb = mix(colorRGB, color, 0.65);
END
```

### 4.6. Interact with the selected area

Get the RGB color coded value selected by picking up the RGB texture in the handle event [31, 36]. Based on the RGB color code (*red, green, blue*) we send queries to the database to retrieve the `ID` and `Name` of the region, from which all relevant information can be retrieved [8]. The virtual void is declared in algorithm. Figure 9 shows the effect of RGB textures on interacting with the different selected area of head states.

**Algorithm 3** . doUserOperations( osgUtil::LineSegmentIntersector::Intersection & result)

```
BEGIN
    osg::Vec3 tc;
    osg::Texture × texture = result.getTextureLookUp(tc);
    osg::Vec4 textureRGB = myImage → getColor(tc);
    int red = textureRGB.r() × 255; int green = textureRGB.g() × 255; int blue = textureRGB.b() × 255;
END
```

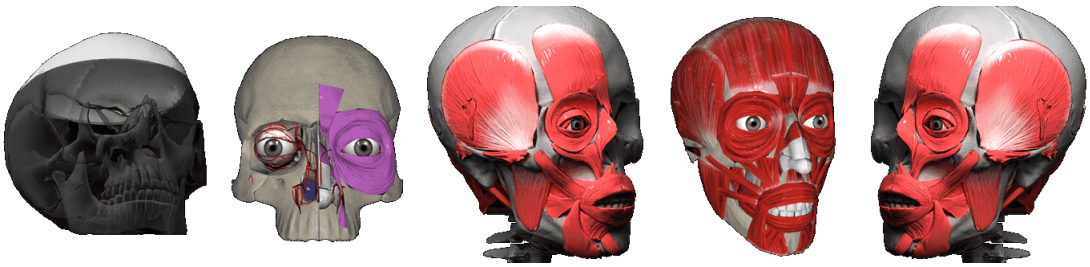


Figure 9. An effective RGB textures of head states.

### 4.7. Rendering

For the general scene representation, the Scene graphs system is employed to render real-time synchronization with hardware rasterization. Users are stopped to engage in low-level problems such as threading and file formatting, texture. At the time of the text formatting and encoding, all systems come with their specific file formats. The binary encoding is a combination of objects scenes with their information and platform handling specialties and serialized the information at the sending time [39]. In addition, Shading has directly employed this technique. The transfer matrices at the vertices have to be magnified with the use of a distant lighting environment in every frame. The coefficient for the particular basis is moved down to the pixel shaders, then the latest normal can be inspected in the normal map and employed to analyze a specific basis. When $M = 4$, shader code below the pixel shader. Shared vertex move down the texture coordinates and three different registers that consists of normal coefficients [9, 37, 40].

**Algorithm 4** . float4 StandardSVDPS (VS_OUTPUT_SVD In)

**BEGIN**

```
float4 RGBColor(0,0,0,1); // sample albedo/normal map
float4 vAlbedo = tex2D(AlbedoTex, In.uv); float4 vNormal = tex2D(NormTex, In.uv);
float4 vU = tex2D(USampler,vNormal); // sample normal dependent texture
RGBColor.r=dot(In.cR,vU); RGBColor.g=dot(In.cG,vU); RGBColor.b=dot(In.cB,vU); //Compute irradiance
return RGBColor × vAlbedo; // scale by albedo and return
```

**END**

Three different textures are sampled that include Texture dependent normal, maps and the albedo of the surface. To shift the workload from CPU to GPU, reduce the amount of data and increase system performance, transfer matrices are fused with the cyclopropane carboxylic acid (CPCA). Figure 10 represents the effective of RGB color selection for complex 3D lungs object from modelling after 6 steps.

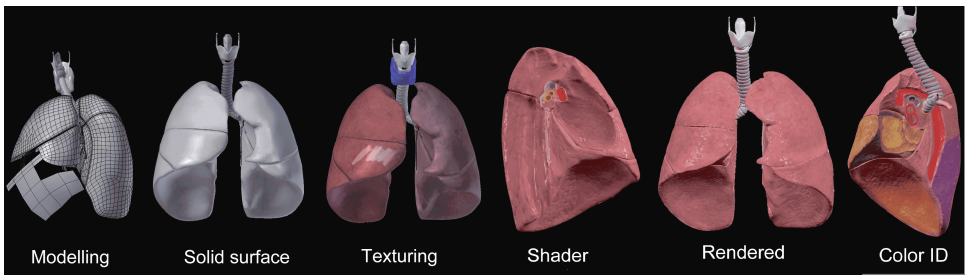



Figure 10. An effective of RGB color selection for complex 3D lungs object.

## 5. RESULT AND DISCUSSION

Our 3D human body simulation system (Anatomy Now) (see https://appadvice.com/app/anatomy-now/1222845241) is developed on two different platforms: 1) OpenGL tools for the simulation of modelled parts in 3D virtual environment 2) Blender for the modeling of limbs. Full human body anatomy includes skeletal system, muscular system, circulatory system, nervous system, respiratory system, digestive system, excretory and genital system, glands and lymph nodes are implemented on the virtual 3D interactive system.

Anatomy professors at universities and hospitals are evaluated our application through the different parameters as well as analysis the accuracy of each and every organs system proposed in our application [41, 42]. The 3D model on the right side of the navigation page has no function at all, therefore is a waste of space. This waste results in a narrow navigation section, which only contains three of the ten body systems. Users need to scroll down to access the remaining systems. The original 3D model page has a total of twenty-one buttons, each with their own unique function [43-45]. However, there is no focus of functions in this design. We organized the functions and tried to emphasize on nine core functions which are related to the 3D model interaction. They are highlighted in the Figure 11. The result is that it is possible to mark and select areas and areas that overlap each other. We also apply the proposal method in the anatomy simulation system, as shown in Figure 12.

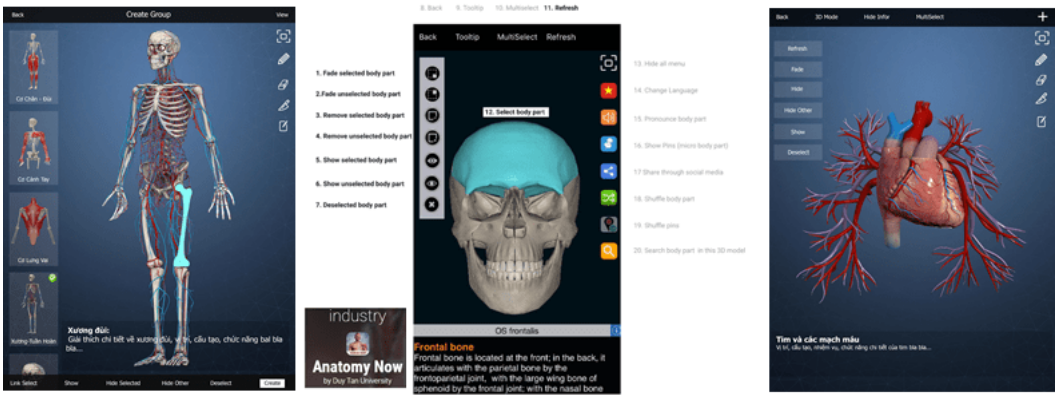



Figure 11. Our 3D human body simulation system (Anatomy Now) and the functions of model interaction.

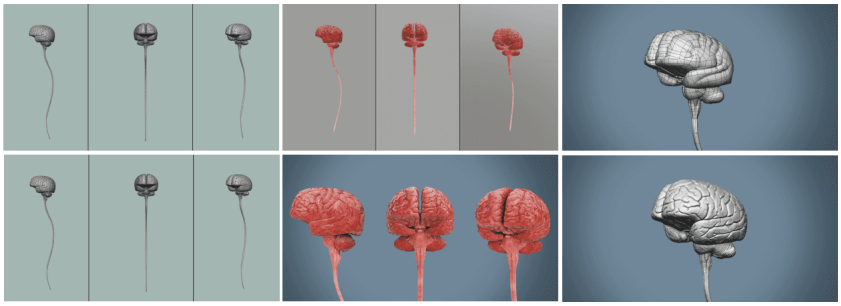


Figure 12. The result of our method and algorithms applied on the brain.

Our method applied to render static and dynamic 3D object. We measured the effective RGB color selection for complex 3D object structure in scene graph systems computation time for several typical texture map resolutions (total number of pixels in all texture maps). The graphics processing units (GPU) based reference implementation was developed with the OpenGL API. The details of experimental computer configuration and test environment are given in Table 1. Full simulation system of human anatomy system in 3D models, anatomical landmarks are shown in Table 2. Diversity of anatomical presentation: Anatomical details, anatomical mold, anatomical region, anatomical group, anatomical area, anatomical system.

Table 1. The details of experimental environment

| **Parameter** | **Description** |
|---|---|
| Processor | Intel Core TM i7 - 6700HQ 2.6 GHz |
| Graphics | NVIDIA GeForce GTX 980M |
| Memory | 16 GB |
| Storage | Samsung PM951 NVMe 256GB |
| OS | Windows 10 64 bit |
| Screen Resolution and Color Depth | 1920 × 1080, 32 |
| Programming Language | OpenSceneGraph C++ |

Table 2. Summary of 3D models in our human anatomy system

| No. | Organ System | Description |
|---|---|---|
| 1 | Skeletal system - ligament | 589 models: 317 skeletal, 272 ligament models |
| 2 | Respiratory system | 6 models |
| 3 | Cardiovascular system | 1517 models: 1472 arterial and venous, 45 cardiac models |
| 4 | Excretory and genital system | 20 models |
| 5 | Muscular system | 510 models |
| 6 | Digestive system | 41 models |
| 7 | Nervous system | 1027 models: 989 neural, 39 brain models |
| 8 | Excretory and denital system | 20 models |
| 9 | Endocrine system | 191 models: 11 glands, 180 lymph node models |

In the first step, a practical test is done to evaluate the performance of pixel formation Table 3 summarizes the results of our algorithm performance on organs systems in human body. The main task of every test was to write the pixel on the full screen, to evaluate the processing time of computation-intensive We have compared and evaluated the parameters: Average speed of rasterization (FPS), GPU and CPU usage percentage (%), GPU dedicated memory, GPU system memory and GPU committed memory.

Table 3. Summary of our method on pixel writing performance

| No. | System | Average FPS | % GPU Usage | % CPU Usage | GPU Dedicated Memory (GB) | GPU System Memory (MB) | Commited GPU Memory |
|---|---|---|---|---|---|---|---|
| 1 | Skeletal system - ligament | 60.01 | 8.59 | 7.98 | 1.01 | 83.3 | 1018.8 MB |
| 2 | Respiratory system | 59.22 | 8.84 | 9.4 | 1.1 | 81.4 | 1.0 GB |
| 3 | Cardio-vascular system | 44.02 | 6.83 | 12.49 | 1.2 | 72.8 | 1.0 GB |
| 4 | Excretory and genital system | 50.5 | 9.11 | 9.28 | 1.2 | 81.1 | 1.1 GB |
| 5 | Muscular system | 47.62 | 12.59 | 12.29 | 1.8 | 75.1 | 1.7 GB |
| 6 | Digestive system | 48.68 | 11.16 | 10.82 | 1.4 | 81.4 | 1.4 GB |
| 7 | Nervous system | 43.52 | 7.03 | 12.49 | 1.2 | 70.8 | 1.2 GB |
| 8 | Endocrine system | 49.69 | 10.68 | 9.89 | 1.2 | 81.2 | 1.1 GB |
| | **Average** | **50.4075** | **9.35375** | **10.58** | **1.26375** | **78.3875** | **1.186865** |

From the results in Table 4 shows that the average speed of rasterization are always in the range from 44.02 to 60.01, the average of GPU usage is 9.35%, the average of CPU usage is 10.58%, the the average of GPU dedicated memory is 1.26 GB, the average of GPU system memory is 78.38 MB and the average of GPU commited memory is 1.186 GB. The results clearly show the advantages of our solution. In the following our objective is take an effective RGB color selection for complex 3D object structure in scene graph systems. The measurements were performed show that our method well with objects complexity, see Table 4 and Figure 13. The optimized GPU deployment provide the fastest performance. But analysis is done through the dedicated hardware. So, there is no need to exchange data between the main memory and GPU.

Our proposed solution allows for effective rendering of complex structured 3D objects on the different devices (see Figure 14). 3D virtual environment creates the full human body that allows users to interact with the body parts, organs and get a narrow-certified detail from this environment. Moreover, this application setup a bridge between the learning and implementation.

Table 4. The comparision of benchmark FPS on the human anatomy system

| No. | Solutions | Rasterization Speed (FPS) |
|---|---|---|
| 1 | Normal Rasterizer | 16.1822 |
| 2 | Incremental Rasterizer | 27.0873 |
| 3 | Block-based Rasterizer | 34.9521 |
| 4 | Non-Optimized GPU implementation | 45.1296 |
| 5 | Our method | **50.4075** |
| 6 | Optimized GPU implementation | 112.2538 |

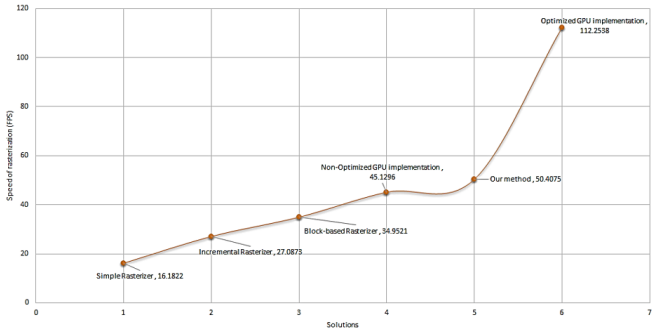



Figure 13. The comparison of the existing techniques.

Figure 14. The effective of our method in practicing in virtual reality of the Anatomy Now system.

## 6. CONCLUSION

Adding many different items of information to a complex 3D object is challenging when the area to which the information needs to be attached is very detailed and overlaps with all kinds of other medical information related to the area. Existing methods to tag areas was not allowing us sufficient locations to attach the information to. Our solution combines a variety of tagging methods, which use the marking method by selecting the RGB color area that is drawn in the texture, on the complex 3D object structure. Then, it relies on those RGB color codes to tag IDs and create relational tables that store the related information about the specific areas of the anatomy. With this method of marking, it is possible to use the entire set of color values (R, G, B) to identify a set of anatomic regions, and this also makes it possible to define multiple overlapping regions.

## REFERENCES

[1] Ham, H., and Wesley, J., "Computer vision based 3D reconstruction: A review," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 4, pp. 2394-2402, 2019.

[2] Rocchini, C., Cignoni, P., Montani, C., and Scopigno, R., "Multiple textures stitching and blending on 3D objects," *Rendering techniques' 99*, pp. 119-130, 1999.

[3] Izzaty, S., Tolle, H., Dermawi, R., and Permana, F., "Augmented reality objects design in augmented story book mobile application for better engagement," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 1, pp. 570-576, 2019.

[4] Rosman, A. N., et al., "Augmented reality application for location finder guidance," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 3, pp. 1237-1242, 2019.

[5] Roth, R. E., Woodruff, A. W., and Johnson, Z. F., "Value-by-alpha maps: An alternative technique to the cartogram," *The Cartographic Journal,* vol. 47, no. 2, pp. 130-140, 2010.

[6] Mileff, P., and Dudra, J., "Advanced 2D rasterization on modern CPUs,*Applied Information Science, Engineering and Technology,* pp. 63-79, 2014.

[7] Son, H., and Kim, C., "3D structural component recognition and modeling method using color and 3D data for construction progress monitoring," *Automation in Construction,* vol. 19, no. 7, pp. 844-854, 2010.

[8] Balazs, A., Guthe, M., and Klein, R., "Fat borders: gap filling for efficient view-dependent LOD NURBS rendering," *Computers and Graphics,* vol. 28, no. 1, pp. 79-85, 2004.
[9] Baumberg, A., "Blending Images for Texturing 3D Models," *Procedings of the British Machine Vision Conference,* 2002.
[10] Liu, H., Philipose, M., and Sun, M. T., "Automatic objects segmentation with RGB-D cameras," *Journal of Visual Communication and Image Representation,* vol. 25, no. 4, pp. 709-718, 2014.
[11] Richtsfeld, A., et al., "Learning of perceptual grouping for object segmentation on RGB-D data," *Journal of visual communication and image representation,* vol. 25, pp. 64-73, 2014.
[12] Yalic, H. Y., and Can, A. B., "Automatic Object Segmentation on RGB-D Data using Surface Normals and Region Similarity," *VISIGRAPP (4: VISAPP),* pp. 379-386, 2018.
[13] AL-Mousa, M. R., Al-salameen, F., and Al-Qawasmi, K., "Using Encryption Square Keywith One-dimensional Matrix for Enhancing RGB Color Image Encryption-Decryption," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 9,no. 3, pp. 771-777, 2018.
[14] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., and Xiao, J., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012,* 2015.
[15] Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S., "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," *European conference on computer vision,* pp. 628-644, 2016.
[16] Fan, H., Su, H., and Guibas, L. J., "A point set generation network for 3d object reconstruction from a single image," *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 605-613, 2017.
[17] Girdhar, R., Fouhey, D. F., Rodriguez, M., and Gupta, A., "Learning a predictable and generative vector representation for objects," *European Conference on Computer Vision,* pp. 484-499, 2016.
[18] Tulsiani, S., et al., "Learning shape abstractions by assembling volumetric primitives," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2635-2643, 2017.
[19] Wu, J., Zhang, et al., "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," *Advances in neural information processing systems,* pp. 82-90, 2016.
[20] Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., and Xiong, Y., "Photo-inspired model-driven 3D object modeling," *ACM Transactions on Graphics (TOG),* vol. 30, no. 4, pp. 1-10, 2011.
[21] Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L., "Grass: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics (TOG),* vol. 36, no. 4, pp. 1-14, 2017.
[22] Setiawan, A., Rostianingsih, S., and Widodo, T. R., "Augmented reality application for chemical bonding based on android," *International Journal of Electrical and Computer Engineering,* vol. 9, no. 1, 2019.
[23] Shujaa, M. I., and Abdulmajeed, A. A., "Implementing bezier surface interpolation and NN in shape reconstruction and depth estimation of a 2D image," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 16, no. 3, pp. 1609-1616, 2019.
[24] Niu, C., Li, J., and Xu, K., "Im2struct: Recovering 3d shape structure from a single rgb image" *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 4521-4529, 2018.
[25] Abeid, J., and Arditi, D., "Time-lapse digital photography applied to project management," *Journal of Construction Engineering and Management,* vol. 128, no. 6, pp. 530-535, 2002.
[26] Wu, Y., and Kim, H., "Digital imaging in assessment of construction project progress," *Proc. Of the 21th ISARC,* pp. 537-542, 2004.
[27] Trucco, E., and Kaka, A. P., "A framework for automatic progress assessment on construction sites using computer vision," *International Journal of IT in Architecture Engineering and Construction,* vol. 2, pp. 147-164, 2004.
[28] Lukins, T. C., and Trucco, E., "Towards Automated Visual Assessment of Progress in Construction Projects" *BMVC,* pp. 1-10, 2007.
[29] Shih, N. J., and Wang, P. H., "Study on construction inaccuracies between point-cloud and building construction models," *Proceedings of 9th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2004),* pp. 623-632, 2004.
[30] Podbreznik, P., and Rebolj, D., "Real-time activity tracking system–the development process," *24th CIB W78 Conf. Bringing ICT Knowledge to Work,* 2007.
[31] Son, H., and Kim, C., "3D structural component recognition and modeling method using color and 3D data for construction progress monitoring," *Automation in Construction,* vol. 19, no. 7, pp. 844-854, 2010.
[32] Netter, F. H., "Atlas of Human Anatomy," *Saunders Elsevier. Bukupedia,* 2014.

[33] Ashour, A. S., Beagum, S., Dey, N., Ashour, A. S., Pistolla, D. S., Nguyen, G. N., and Shi, F., "Light microscopy image de-noising using optimized LPA-ICI filter," *Neural Computing and Applications,* vol. 29, no. 12, pp. 1517-1533, 2018.
[34] Nayyar, A., Mahapatra, B., Le, D., and Suseendran, G., "Virtual Reality (VR) and Augmented Reality (AR) technologies for tourism and hospitality industry. *International Journal of Engineering and Technology,* vol. 7, vol. 2.21, pp. 156-160, 2018.
[35] Tortora, G. J., and Derrickson, B. H., "Principles of Anatomy and Physiology," *John Wiley and Sons,* 2018.
[36] Gross, M., and Pfister, H. (Eds.)., "Point-based Graphics," *Elsevier,* 2011.
[37] Bernardini, F., Martin, I. M., and Rushmeier, H., "High-quality texture reconstruction from multiple scans," *IEEE Transactions on Visualization and Computer Graphics,* vol. 7, no. 4, pp. 318-332, 2001.
[38] Bailey, M., and Cunningham, S., "Graphics shaders: Theory and Practice," *AK Peters/CRC Press,* 2016.
[39] Fünfzig, C., "Spherical Techniques and their Applications in a Scene Graph System: Collision Detection and Occlusion Culling," *Cuvillier Verlag,* 2007.
[40] Wang, G., Zhang, X., Zhu, C., Wang, H., Peng, L., and Hou, Z., "Research on 3D Terminal Rendering Technology Based on Power Equipment Business Features," *Recent Trends in Intelligent Computing, Communication and Devices,* pp. 175-181, 2020.
[41] Le, D. N., Van Le, C., Tromp, J. G., and Nguyen, G. N. (Eds.)., "Emerging technologies for health and medicine: virtual reality, augmented reality, artificial intelligence, internet of things, robotics, industry 4.0," *John Wiley and Sons,* 2018.
[42] Le, D. N., Nguyen, G. N., Bhateja, V., and Satapathy, S. C., "Optimizing feature selection in video-based recognition using Max-Min Ant System for the online video contextual advertisement user-oriented system," *Journal of computational science,* vol. 21, pp. 361-370, 2017.
[43] Tromp, J., Le, D. N., and Le, C., "Emerging Extended Reality Technologies for Industry 4.0: Experiences with Conception, Design, Implementation, Evaluation and Deployment," *John Wiley and Sons,* 2020.
[44] Tromp, J., Le, C., Le, B., and Le, D. N., "Massively multi-user online social virtual reality systems: ethical issues and risks for long-term use," *Social Networks Science: Design, Implementation, Security, and Challenges,* pp. 131-149, 2018.
[45] Hore, S., et al., "An Integrated Interactive Technique for Image Segmentation using Stack based Seeded Region Growing and Thresholding," *International Journal of Electrical and Computer Engineering,* vol. 6, no. 6, pp. 2773-2780, 2016.

## BIOGRAPHIES OF AUTHORS

**Chung Le Van** received the MSc degree in computer science of Duy Tan University, Vietnam. He is studying PhD. He currently works in CVS (Center for Visualization & Simulation) lab of Duy Tan University, Da Nang, Vietnam. He has a total academic teaching and research experience of 17 years; He currently is technical director of CVS. He researches on field medical image processing, eHealth, AR/VR, Social VR/AR, Multi-modal VR/AR

**Gia Nhu Nguyen** received the PhD degree in Mathematical for Computer Science from Ha Noi University of Science- VNU, Vietnam. Currently, he is Dean of Graduate School- Duy Tan University, Viet Nam. He has a total academic teaching experience of 19 years with more than 60 publications in reputed international conferences, journals and online book chapter contributions (Indexed By: SCIE, SSCI, Scopus, ACM, DBLP). His area of research includes: Healthcare Informatics, Network Performance Analysis and Simulation, Computational Intelligence. Recently, he has been the technical program committee, review committee, track chair for serval international conferences under Springer-ASIC/LNAI Series. 6 Computer Science books published in Springer, IGI Global, CRC, Wiley Publication. Presently he is Associate Editor of the IGI-Global: International Journal of Synthetic Emotions (IJSE).

**Tri Huu Nguyen** received the MD. and PhD. degree in Digestive Surgery from Hue University of Medicine and Pharmacy, Vietnam. Currently, he is Head of Department of Anatomy and Surgical Training, Hue University of Medicine and Pharmacy; he is Vice head of Digestive Department, Hue University of Medicine and Pharmacy Hospital also. He has a total academic teaching of twenty years. He researches on Anatomy and Surgery with 17 publications

**Tung Sanh Nguyen** had graduated from Hue University of Medicine and Pharmacy in 1983, received the PhD in Medicine in Hanoi in 2010. He has been a lecturer of Human Anatomy since 1987 and a Surgery in Thoracic - Vascular Surgery since 1998. He is currently the Vice Chairman of the Vietnam Association of Morphology, a lecturer in Anatomy of Hue University of Medicine and Pharmacy, Vietnam. His research focus on clinical anatomy, thoracoscopic and vascular surgery.

**Dac-Nhuong Le** has a M.Sc. and PhD. in computer science from Vietnam National University, Vietnam in 2009 and 2015, respectively. He is Associate Professor, Deputy Head of Faculty of Information Technology, Haiphong University, Haiphong, Vietnam. He has a total academic teaching experience of 15+ years. His researches are in fields of evolutionary multi-objective optimization, network communication and security, VR/AR. He has 50+ publications in the reputed international conferences, journals and book chapter contributions (Indexed by: SCIE, SSCI, ESCI, Scopus, ACM, DBLP). Recently, he has been the technique program committee, the technique reviews, the track chair for international conferences under Springer Series. Presently, he is serving in the editorial board of international journals and 15+ computer science edited/authored books which published by Springer, Wiley, CRC Press. Further info on his homepage: http://dhhp.edu.vn/nhuongld/