Software filtering applications for the analysis of dental images

ANTOANELA NAAJI, MARIUS-CONSTANTIN POPESCU, GABRIEL CALIN SARLA
"Vasile Goldis" Western University of Arad
Arad, 94-96 Revolutiei Blvd.
ROMANIA
anaaji@uvvg.ro, mpopescu@uvvg.ro, gabriel_sarla2001@yahoo.comhttp://www.uvvg.ro

Abstract: - Even if digital radiography is performed using modern procedures, which allow viewing and enlarging the image on the computer monitor, sometimes, due to the distortions introduced by the equipment or the need to perform a thorough analysis with a high degree of precision, image processing is required. For this matter, in this paper we present a software application created to help the dental practitioners, when they need a clear image, to discover the dental diseases or other submicroscopic problems on the dental surface, in the incipient phase, bone loss or any other problems that cannot be seen from a medical image, mainly due to the distortions introduced by the equipment. The advantages of dental image processing consist of brightness adjustment (attenuation, accentuation), contrast adjustment (attenuation, accentuation), image negation, histogram equalization, color transformation or noise removal. The paper describes the software accomplishment and implementation of filtering algorithms, which are efficient compared to the existing solutions, which can be used to improve the quality of the medical imaging. The algorithms, implemented in Java, are obtained by modifying the operators and procedures of the fundamental algorithms.

Key-Words: - Image filtering, Filter implementation, Distortion elimination, Software Applications, Performance analysis in Dentistry.

Received: November 2, 2019. Revised: March 31, 2020. Accepted: April 21, 2020. Published: April 30, 2020.

1 Introduction

The processing of digital dental images involves their manipulation through a numerical calculation system (usually a numerical calculator). The images are transmitted to the computer in various forms, depending on the image sensor and the hardware used for image compaction.

In medicine, the most used types of images are: monochrome, binary (with only two levels of brightness), color (represented by three separate monochrome components), multispectral (with more than three monochrome components) [1]. Colored images are constructed from 3 intensity maps superimposed on each other [2]**.** Each map represents the intensity of a primary color. The 3 primary colors are 3 vectors (or axes) that form the color space base. Each color corresponds to a point in the vector space. The color space is formed by the primary red, green and blue colors corresponding to R, G and B on the axes of the color space. An 8-bit resolution corresponds to integers that can take values from 0 to 255. There are no negative values, the color space being a cube. The color space is discrete having 256^3 possible colors, that is 16777216 elements in the cube. The same color has different value GB and CMYK space (C*yan*, M*agenta*, Y*ellow*, K*ey*).

There are a number of operations that are performed in the processing of medical images and they can be classified according to the proposed objectives, as follows [3]: image enhancement (which involves image processing so that the result is convenient for a particular application, including operations of enhancement of contrast or brightness, edge highlighting, noise removal, sharpening, deblurring and blurring due to inadequate focus), image restoration or eliminating the effects of a known cause (such as removing blur caused by the movement of the patient, moving away the optical distortions, removal of the periodic interference) and image segmentation (involves dividing an image into constituent parts or isolating certain features from the image). These classes of

operations are not disjunct. A certain algorithm can be used in the case of image enhancement [4].

Because the result of image processing for detecting dental disease is based on the image taken from the X-ray, there are a number of external and internal factors that make this process more difficult, such as: noise in the image, the presence of bone tissue, etc. The process of removing the noise from the image is quite cumbersome, because by applying various noise elimination algorithms, other information can be removed from images such as very fine tissue images or very small tumors. Regarding the presence of bone tissues in a grayscale image and considering that both bones tissues and tumors have the intensity quite close, the tumors cannot be identified directly, but only after removing the bone tissue from the image. [5].

2 Types of medical dental images

Mainly, the digital system of dental image acquisition contains an optical system, a sensor, a stage for amplifying and filtering the signal from the sensor and an analog-to-digital converter. All these equipment affect the acquired image, introducing various distortions (for example, those caused by the optical system).

Radiographs are obtained when a sensor records the radiation that reacts with the patient's physical area. Radiographs are indispensable for the detection of a wide range of conditions, being necessary in dental clinics. With the modernization of technology, radiography has gradually shifted from conventional to digital film. In establishing a precise diagnosis, the digital dental radiography reveals to the dentist details that can determine the nature of an intervention on the teeth and even an entire treatment plan [6].

Dental radiography is an image of the teeth, jaws, bones and soft tissues as a result of X-ray exposure or digital scanning. Its role is to highlight the patient's dental problems, because without clear signs at the level of the denture or gum, the dentist cannot draw a conclusion based solely on the symptoms indicated by the patients [7].

The retroalveolar (periapical) x-ray shows the tooth in all its size, the crown being visible, the root and even the bone in which it is inserted. In the retroalveolar X-ray, up to three teeth will be included next to the one targeted directly on the mandible or on the jaw.

Panoramic radiography or orthopantomography targets dental arches, as well as the maxillary sinuses or mandibular channels. There are several types of panoramic radiographs: standard, with frontal focus (this increases the degree of focus in the frontal area of the teeth), orthogonal (it helps avoiding the overlapping of the teeth, so that the doctor can have a clearer image), pediatric (used in the case of children and ensures a better image on the permanent teeth buds, still in formation).

The occlusal X-ray (maxillary and mandibular) covers the entire dental arch, along with the upper jaw and mandible. The usefulness of this x-ray consists in detecting extra teeth, teeth that have not yet erupted, but also bone fractures or foreign objects, if the patient presents to the dental office after a trauma. Also the occlusal X-ray identifies the presence of cysts or deep abscesses.

Bite-wing radiography (with film-bite) is an intraoral recording. The dental crowns of the posterior teeth of the upper and lower jaws can be seen on the same radiograph, and the teeth are represented without being overlapped. The need to perform a bite-wing X-ray is also required when detecting tooth caries in the contact area of the teeth and under the inlays. Bite-wing X-ray discovers including caries located only in the tooth enamel.

Cephalometric projections or frontal, profile and axial teleradiographs highlight the entire skull and are used to examine teeth in connection to the patient's jaw and profile.

Dental computed tomography (CT) highlights the internal structures of the body in two-dimensional images [8]. These allow high precision measurements to be made in the lateral mandibular area, on the upper arch, in the sinus and nostril area. Dental computer tomography is used to identify problems related to the bones of the face, such as tumors or fractures, by specialists in endodontics, implantology and oral-maxillofacial surgery. The information obtained from these examinations combined with the clinical examination will allow the specialist doctor to establish the diagnosis and elaborate the treatment plan.

3 Standard filtering techniques

The main steps that need to be implemented in a software application, to solve a processing problem are: image acquisition (obtaining the digital image of the teeth), preprocessing (increasing the contrast, removing the noise or identifying the region that may contain the area of interest), segmenting the image (the part containing the affected teeth is extracted from the image [9]), the representation and description of the image (some features that allow the differentiation between the dental areas are extracted), the recognition and interpretation (assigning some labels to the dental areas, according

to the descriptors used in the previous step and associating the meaning with the respective labels). Not all of these steps are present in any image processing and the image compression step was not included [10].

Image acquisition is essential and involves the presence of an image sensor and the ability to digitize the output signal. The image sensor is a sensitive device in a certain range of the energy spectrum, which produces at the output an electrical signal proportional to the energy of this radiation. This signal is subsequently digitized.

The preprocessing stage includes operations that have the effect of improving the image, because during the acquisition there are processes that can cause the image degradation (inadequate focus, either due to the patient's movement or due to stepping outside of the field, geometric distortions, etc.).

The segmentation of the image aims to decompose it into constituent components, possibly with the achievement of certain measurements.

Any image processing operation transforms the pixel value. Image filtering can be achieved by applying a kernel (filter) to an image by transforming the value for each pixel into a value based on the kernel and neighboring pixels in the original image. Mathematically, a convolution occurs between the image and the kernel.

Image filtering is a spatial operation whose purpose is to remove noise. Noise is unwanted information that deteriorates the image quality and comes from the image acquisition process (converting optical image into continuous electrical signal). An image with independent noise of the content of the image can be modeled by a function of 2 variables:

$$g(x,y) = f(x,y) + h(x,y), \quad (1)$$

where, $f(x,y)$ is the input image for the image forming device or the actual image, and $h(x,y)$ represents independent noise of the content of the image, also called additive noise.

In the case that the noise depends on the content of the image (such as monochromatic radiations produced by a surface, which produce wave interference), the noise can be implemented through a nonlinear model. As these mathematical models are complicated, noise is considered, if possible, as being independent of the content of the image.

The elimination of artifacts (image defects) is done with a medium filter, and the intensification of images (restoration of images) with *high-boostfiltering*.

In the case of Gaussian filtering, the program sequence involves entering some data:

```
voidGaussianBlur(InputArraysrc,OutputA
rray dst,
Sizeksize,doublesigmaX,doublesigmaY=0,
intborderType=BORDER_DEFAULT)
```

where, `src` and `dst` represent the source and destination images, `ksize` is the kernel size, `sigmaX` is the standard deviation in the direction of X (in the case that 0 is used, it is automatically calculated from `ksize`), `sigmaY` is similar to `sigmaX`, but in the direction Y, `borderType` affects the pixels from the margin.

Filtering is applied locally at each pixel in the image by replacing the intensity or color value of the current pixel with a value that depends on the intensity or color values of the neighboring pixels. The number of neighbors taken into consideration will determine the size of the filter.

Median filtering is an operation to improve the signal-to-noise ratio of the image if the noise affects the isolated elements of the image. In the case of median filtering, the program contains a series of data, as it follows:

```
voidmedianBlur(InputArraysrc,
OutputArray dst, intksize)
```

where, `src` represents the input image, `dst` is the output image, and `ksize` is the size of the filter (odd and greater than 1).

The operation is nonlinear and acts as a morphological filter, similar to the uniform smoothing filter, preserving the structures edges/margins of the structures. The median filtering algorithm involves ordering the values of the elements in the selected window (rising or decreasing) and replacing the pixel of interest with the value of the one in middle of the ordered string. Median filtering ensures that the initial image resolution is maintained. The results of the median filtering are: good, in the case of binary noise, and unsatisfactory if the noise has a Gaussian distribution and the number of pixels affected by the noise is greater than or equal to half the number of pixels in the current window. If an image is repeatedly filtered with the same filter (for example the median filter) then the image will no longer change.

The noise "salt and pepper" is median filtering.

The most widely used technique for changing the contrast of an input image f, is a linear transformation on portions:

$$g=\begin{cases}\dfrac{\alpha}{A_1}f & for \quad 0\le f<A_1\\ \alpha+\dfrac{\beta-\alpha}{A_2-A_1}(f-A_1) & for \quad A_1\le f<A_2.\\ \beta+\dfrac{B-1-\beta}{B-1-A_2}(f-A_2) & for \quad A_2\le f<B\end{cases} \qquad (2)$$

where, A_1, A_2, α and β are control parameters, which define the points (A_1,α), and (A_2,β), and the fixed points $(0,0)$, and $(B-1, B-1)$ define the 3 segments of line which appear in connection.

The result of applying point operations is obtained by changing the value (gray level) of each pixel of the initial image *f*, according to (2), obtaining the new gray level *g*.

The disadvantage of the linear technique is that the change of the contrast is the same over the whole range of grayscale, a non-uniform change over the entire range of grayscale or around a certain grayscale level not being possible. Nonlinear techniques have these properties. Within the operations of image enhancement are also simple arithmetic operations, such as the negation described by the relation:

$$g = A(f) = B - 1 - f. \qquad (3)$$

The effect of changing the contrast is based only on the characteristics of the human visual system, for which the contrast depends on the difference in brightness between pixels belonging to an object, respectively to the background, relative to the average luminance of the background. Such inversions are useful in the analysis of medical images, where for an automatic analysis they must be reversed. Regarding the human perception of the images, we mention that the intensity observed is influenced by the background. The observer perceives differently, because the perception is made by difference from the environment.

In image processing applications, a (L*ook* U*p* T*able* - LUT), is used to transform the input data into a desired output format. For example, a gray-scale image of the dental arch will be transformed into a color image to highlight differences. The search tables provide an output value for each of a range of index values. The LUT method is applied if the number of inputs is relatively small or if the computation time of the analytical form is too high.

A common LUT, called a *colormap*, is used to determine the color and intensity values with which a particular image will be displayed. In the computed tomography, "windows" refers to an afferent concept for determining the way of displaying the intensity and measures radiations.

In the case of dental computed tomography, the application is able to upload and open a DICOM (D*igital* I*maging and* C*ommunications in* M*edicine*) file, in order to be able to extract from it the image and the additional information that accompany it. The image is transformed into a format that can be easily processed.

4 Implementation of the filtering algorithms

Noise modeling, in the case of filtering medical images, can be done by distribution: Gaussian, uniform, or "salt and pepper" type [11]. In the paper it was opted for the creation and implementation in Java of a "salt and pepper" type filter (Fig. 1).

```
public class SaltAndPepper {
private String workingDirPath;
private String imagePath;privateBufferedImage
image;private String imageName; private static
final String PROCESS="saltAndPepperFilter";
public static final String PROCESS_TYPE=
"oneImageProcessor";private ScrollableLogArea
log;
publicSaltAndPepper(String workingDirPath,
ScrollableLogArea log)
{this.workingDirPath=
workingDirPath;this.log=log;
log.setLoggedClass(HighPassFilter.class.getName(
));}
public void readImage(String imagePath)
{this.imagePath=imagePath;
try {File file=new File(this.imagePath);
imageName=file.getName().substring(0,file.getNam
e().length()-4)+"_";
image=ImageIO.read(file);} catch (IOException e)
{
log.error("Could not read image.",
e.getMessage());e.printStackTrace();}}
public String processImage(String imagePath) {
StopWatchcronometer=new StopWatch();
cronometer.start();
log.info("Salt&Pepper Filter processing
started.");
readImage(imagePath);
BufferedImageresultedImage=
removeSaltAndPepperNoise(); String
exportedImagePath =exportImage(resultedImage);
cronometer.stop();
log.appendInfo("Salt&Pepper Filter processing
finished in "+cronometer.getTime()+"ms.");
returnexportedImagePath;}
publicBufferedImageremoveSaltAndPepperNoise() {
for (int y=1;y<image.getHeight()-1;y++) {
for (int x=1; x<image.getWidth()-1; x++) {
int center=image.getRGB(x, y);
intcenterColor=(center & 0x00ff0000) >> 16;
if (centerColor>200||centerColor<50) {
intleft=image.getRGB(x-1,y);
int right=image.getRGB(x+1,y);
int down=image.getRGB(x,y+1);
int up=image.getRGB(x,y-1);
intleftColor=(left&0x00ff0000)>>16;
intrightColor=(right&0x00ff0000)>>16;
```

```
intdownColor=(down & 0x00ff0000)>>16;
intupColor=(up & 0x00ff0000)>>16;
intvalue=(leftColor+rightColor+upColor+downColor
)
/4;Color color=new Color(value,value,value);
intrgb=color.getRGB();image.setRGB(x,y,rgb);}
else {image.setRGB(x,y,center);}}}return image;}
public                                    String
exportImage(BufferedImageresultedImage)    {File
outputImage=new
File(workingDirPath+"/"+imageName+PROCESS+".jpg"
);
try{ImageIO.write(resultedImage,"jpg",outputImag
e);
} catch (IOException e) {log.error("Could not
save intermediate image.",e.getMessage());
e.printStackTrace();}
returnoutputImage.getAbsolutePath();}}
```

Figure 1. Sequence in the program *SaltAndPepper.java*.

The purpose of the software implementation of a *low-pass* type of image filter is to reduce the noise (Fig. 2), and of a *high-pass* filter (Fig. 3), to accentuate the details (identifying the edges). In the situation of filtering the images with *low-passfiltering* [12], the high frequencies (values above the threshold frequency) are attenuated or eliminated (reduced amplitude), while with *high-passfiltering* [13], the high frequencies are accentuated or amplified.

```
public class LowPassFilterimplementsFilter
{private StringworkingDirPath;
private StringimagePath;
private BufferedImageimage;
  private StringimageName;
private static final int LOW_PASS_VALUE=127;
private   static   final   String   PROCESS   =
"lowPassFilter";
public static final String PROCESS_TYPE=
"oneImageProcessor";
private ScrollableLogArea log;
public       LowPassFilter(StringworkingDirPath,
ScrollableLogArea  log)  {this.workingDirPath  =
workingDirPath;this.log =log;
log.setLoggedClass(LowPassFilter.class.getName()
);}
public voidreadImage(StringimagePath) {
this.imagePath=imagePath;
try { File file=new File(this.imagePath);
imageName=file.getName().substring(0,
file.getName().length()-4) + "_";
image=ImageIO.read(file);}
catch              (IOException              e)
{log.error("Couldnotreadimage.",e.getMessage());
e.printStackTrace();}}
public StringprocessImage(StringimagePath) {
StopWatchcronometer = newStopWatch();
cronometer.start();
log.info("LowPassFilterprocessingstarted.");
readImage(imagePath);
BufferedImageresultedImage = getLowPassImage();
StringexportedImagePath =
exportImage(resultedImage); cronometer.stop();
log.appendInfo("LowPassFilterprocessingfinished
in "+cronometer.getTime()+"ms.");
returnexportedImagePath;}
```

```
public
StringexportImage(BufferedImageresultedImage)
{File                              outputImage=new
File(workingDirPath+"/"+imageName+
PROCESS+".jpg");
try{ImageIO.write(resultedImage,"jpg",
outputImage);} catch (IOException e) {
log.error("Couldnotsave   intermediate   image.",
e.getMessage()); e.printStackTrace();}
returnoutputImage.getAbsolutePath();}
private BufferedImagegetLowPassImage() {
 for (int y=0;y<image.getHeight(); y++) {
 for (int x=0;x<image.getWidth(); x++) {
intclr=image.getRGB(x, y);
intpixelValue=(clr& 0x00ff0000)>>16;
if(pixelValue<=LOW_PASS_VALUE){ } else {
Color                                  color=new
Color(0,0,0);intrgb=color.getRGB();
image.setRGB(x,y,rgb);}}}returnimage;}}}
```

Figure 2. Sequence in the program *LowPassFilter.java*.

```
public class HighPassFilter implements Filter {
private String workingDirPath;
private String imagePath;
privateBufferedImage image;
private String imageName;
private static final int HIGH_PASS_VALUE = 128;
private               static               final
StringPROCESShighPassFilter";
public static final String PROCESS_TYPE=
 "oneImageProcessor";private   ScrollableLogArea
log;
publicHighPassFilter(String      workingDirPath,
ScrollableLogArea log)
{this.workingDirPath=workingDirPath;
this.log=log;
log.setLoggedClass(HighPassFilter.class.getName(
));}
public void readImage(String imagePath) {
this.imagePath=imagePath;  try  {File  file=new
File(this.imagePath);imageName=file.getName().su
bstring(0,file.getName().length()-4)+"_";image=
ImageIO.read(file);}
catch (IOException e) {log.error("Could not read
image.", e.getMessage());e.printStackTrace();}}
public String processImage(String imagePath)
{StopWatchcronometer=new StopWatch();
cronometer.start();
log.info("HighPass Filter processing started.");
readImage(imagePath);
BufferedImageresultedImage=getHighPassImage();
String
exportedImagePath=exportImage(resultedImage);
cronometer.stop();log.appendInfo("HighPassFilter
processing                                finished
in"+cronometer.getTime()+"ms.");
returnexportedImagePath;}
public                                    String
exportImage(BufferedImageresultedImage){File
outputImage=new
File(workingDirPath+"/"+imageName+
PROCESS+".jpg");
try{ImageIO.write(resultedImage,"jpg",outputImag
e);} catch (IOException e) {log.error("Could not
save intermediate image.", e.getMessage());
e.printStackTrace();}
returnoutputImage.getAbsolutePath();}
privateBufferedImagegetHighPassImage() {
for (int y=0;y<image.getHeight();y++) {
for (int x=0;x<image.getWidth();x++) {
intclr=image.getRGB(x,y);
intpixelValue=(clr& 0x00ff0000)>>16;
if(pixelValue>=HIGH_PASS_VALUE){
```

```
Color color = new Color(0,0,0);
intrgb=color.getRGB();image.setRGB(x,y,rgb);}}}
return image;}}
```

Figure 3. Sequence in the program *HighPassFilter.java*.

In order to determine the contours of a medical image, a transition to a grayscale image is made. This simplifies the coding of the image, but also the access to a wider palette of processing techniques (Fig. 4).

```
public class GrayScaleFilterimplementsFilter{
private StringworkingDirPath;
private StringimagePath;
private BufferedImage image;
private String imageName;
private    static    final    String    PROCESS=
"grayScaleFilter";
public   static   final   String   PROCESS_TYPE=
"oneImageProcessor";private    ScrollableLogArea
log;
publicGrayScaleFilter(String     workingDirPath,
ScrollableLogArea   log)   {this.workingDirPath=
workingDirPath;this.log=log;
log.setLoggedClass(GrayScaleFilter.class.getName
())}
public void readImage(String imagePath) {
this.imagePath = imagePath;
try {File file=new File(this.imagePath);
imageName=file.getName().substring(0,file.getNam
e().length()-4)+"_";
image=ImageIO.read(file);} catch (IOException e)
{
log.error("Could      not      read      image.",
e.getMessage());e.printStackTrace();}}
public String processImage(String imagePath) {
StopWatchcronometer=new StopWatch();
cronometer.start();
log.info("Grayscale processing started.");
readImage(imagePath);
BufferedImageresultedImage=getGrayScaleImage();
  String exportedImagePath=exportImage
(resultedImage);cronometer.stop();
log.appendInfo("Grayscale processing finished in
"+cronometer.getTime()+"ms.");
returnexportedImagePath;}
public                                    String
exportImage(BufferedImageresultedImage)    {File
outputImage=new              File(workingDirPath
+"/"+imageName+PROCESS+".jpg");
try     {ImageIO.write(resultedImage,     "jpg",
outputImage);} catch (IOException e) {
log.error("Could not save intermediate image.",
e.getMessage());e.printStackTrace();}
returnoutputImage.getAbsolutePath();}
privateBufferedImagegetGrayScaleImage() {
for (int y=0;y<image.getHeight();y++) {
for (int x=0;x<image.getWidth();x++) {
intclr=image.getRGB(x,y);
int red=(clr&0x00ff0000)>>16;
int green=(clr& 0x0000ff00)>>8;
int blue=clr& 0x000000ff;
intgrayScaleValue=(red+green+blue)/3;
   Color     color=new     Color(grayScaleValue,
grayScaleValue, grayScaleValue);
intrgb=color.getRGB();
image.setRGB(x,y,rgb);}}return image;}}
```

Figure 4. Sequence in the program *GrayScaleFilter.java.*

The contrast of an image is the result of the difference between the maximum and minimum brightness of the pixels of the digital image. A reduced contrast image implies a small variation in brightness in most of its spatial regions. By changing the contrast of an image it is desired to change its characteristics so that the new image is clearer to the human eye or so that can be processed more easily (Fig. 5).

```
public   class   ContrastStretching   implements
Filter {
private String workingDirPath;
private String imagePath;
privateBufferedImage image;
private String imageName;private intlMax;
privateintlMin;private static final int MP=255;
private    static    final    String    PROCESS=
"contrastStretchingFilter";
public   static   final   String   PROCESS_TYPE=
"oneImageProcessor";private    ScrollableLogArea
log;
publicContrastStretching(String  workingDirPath,
ScrollableLogArea   log)   {this.workingDirPath=
workingDirPath;this.log=log;
log.setLoggedClass(HighPassFilter.class.getName(
));}
public void readImage(String imagePath) {
this.imagePath=imagePath;try    {File    file=new
File(this.imagePath);imageName=file.getName().su
bstring(0,file.getName().length()-4)+"_";
image=ImageIO.read(file);} catch (IOException e)
{log.error("Could            not            read
image.",e.getMessage());
e.printStackTrace();}}
public String processImage(String imagePath) {
StopWatchcronometer=new StopWatch();
cronometer.start();log.info("Contrast Stretching
processing
started.");readImage(imagePath);BufferedImageres
ultedImage
=getContrastStretchingImage();StringexportedImag
ePath=exportImage(resultedImage);cronometer.stop
();
log.appendInfo("Contrast  Stretching  processing
finished in "+cronometer.getTime()+"ms.");
returnexportedImagePath;}
public                                    String
exportImage(BufferedImageresultedImage)    {File
outputImage=new
File(workingDirPath+"/"+imageName+PROCESS+
".jpg");
try          {ImageIO.write(resultedImage,"jpg",
outputImage);}     catch     (IOException     e)
{log.error("Could not save intermediate image.",
e.getMessage());e.printStackTrace();}
returnoutputImage.getAbsolutePath();}
publicBufferedImagegetContrastStretchingImage(){
setLimits();
log.appendInfo("The    limits    for    Contrast
Stretching:
Max="+getlMax()+"Min="+getlMin()+".");
for (int y=0; y<image.getHeight();y++) {
for (int x=0; x<image.getWidth();x++) {
intclr=image.getRGB(x,y);
intpixelValue=clr& 0x000000ff;
 float
contrastStretchingValue=((float)(pixelValue    -
getlMin())/(getlMax()-getlMin()))*MP;Color
color=new
Color(Math.round(contrastStretchingValue),
Math.round(contrastStretchingValue),
Math.round(contrastStretchingValue));
```

```
intrgb=color.getRGB();image.setRGB(x,y,rgb);}}
return image;}
private                                    void
setLimits(){setlMax(0);setlMin(255);
for (int y=0;y<image.getHeight();y++) {
for (int x=0; x<image.getWidth();x++) {
intclr=image.getRGB(x,y);
intpixelValue=clr& 0x000000ff;
if(pixelValue>lMax)setlMax(pixelValue);
if(pixelValue<lMin)setlMin(pixelValue);}}}
publicintgetlMax() {return lMax;}
public void setlMax(intlMax) {this.lMax=lMax;}
publicintgetlMin() {return lMin;}
public voidsetlMin(intlMin) {this.lMin=lMin;}}
```

Figure 5. Sequence in the program *ContrastStretching.java*.

The following paragraph will analyze the results obtained after applying the software on different types of radiographs. The images are transformed into a format that can be easily processed with various image processing algorithms, passing through a few essential steps. After the image of the consult was processed and diagnosed with the help of the application, under the observation of the dentist, it may be saved together with its additional information, so that it can be further used in comparison with results that will appear.

5 Results and discussions

Although the panoramic radiography captures the image of difficult-to-reach places (such as the wisdom tooth or the temporal-mandibular joints), highlighting the mandibular fractures and other problems in this area, the panoramic image is not effective for assessing the bone density and caries.

Figure 6 shows the image of the patient's dentition subjected to the doctor's observation and figure 7 the intermediate phase of treatment that presents the medical decision taken following the analysis of the first filtered image.

In Fig. 6a and Fig. 7a are illustrated panorama radiographs, which will be filtered in order to eliminate the noises and to highlight the edges. For eliminating the noise during capture, Salt & Pepper filter was implemented (Fig.6b and Fig.7b). Using the low pass filter (Fig. 6c and Fig. 7c) the image spectrum is uniformized. The image resulting from the application of the *low pass* filter is more blurred than the initial image (Fig. 6a). The filter attenuates sudden intensity transitions, leaving the impression that the image has fewer details. Figure 6d and Fig. 7d shows the contours of the initial image, due to the soft implementation of the derivative behavior. A *high pass* filter emphasizes high frequency components, having a small effect on low frequency ones. Therefore, in the image resulting from the application of the high pass filter, the intensity differences (details) in the intensity transition zones are emphasized.

In Fig.6e and Fig.7e the input image is converted to grayscale. Contrast enhancement can be achieved, either by histogram equalization or by intensity scaling. Histogram equalization tends to amplify noise, and intensity scaling allows the user to focus the attention on certain levels of intensity in the image, modifying the image so that the band of interest extends across the entire image. The basic purpose of this filtering is to eliminate from picture the noise that may occur during the capture. The filter is nonlinear, reducing noise and keeping edges more efficient than the median filter (represented in Fig 6b and Fig 7b).

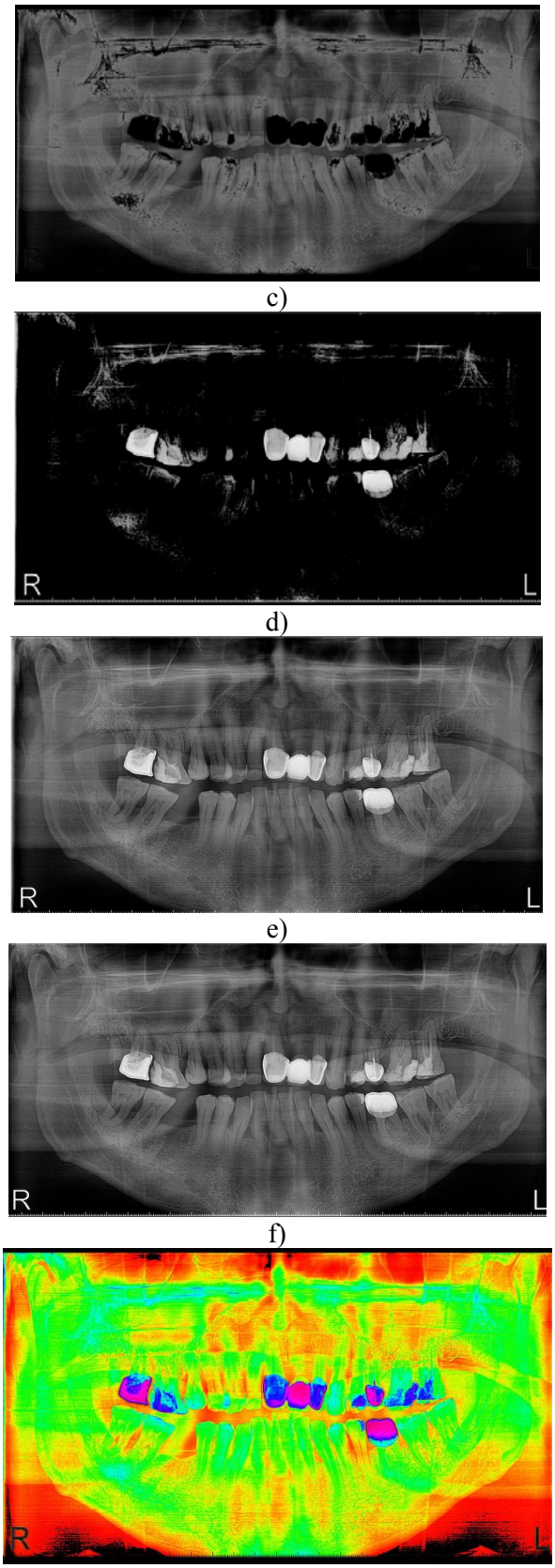
Figure 6f, respectively Fig. 7f illustrate an improvement of the contrast in the images presented in Fig. 6a and Fig. 7a, by "extending" the range of intensity values that it contains.

The module Ramp LUT (Fig. 6g, Fig. 7g) generates an RGB and alpha ramp lookup table object [14, 15]. LUT is a data set that allows a numeric transformation of pixel values to change the way a picture looks.

The LUT filter is used to stylize images with a certain "look". You can mix the color to enhance the details in the image. Color adjustments can change the perception of the dental condition.

a)

b)



c)

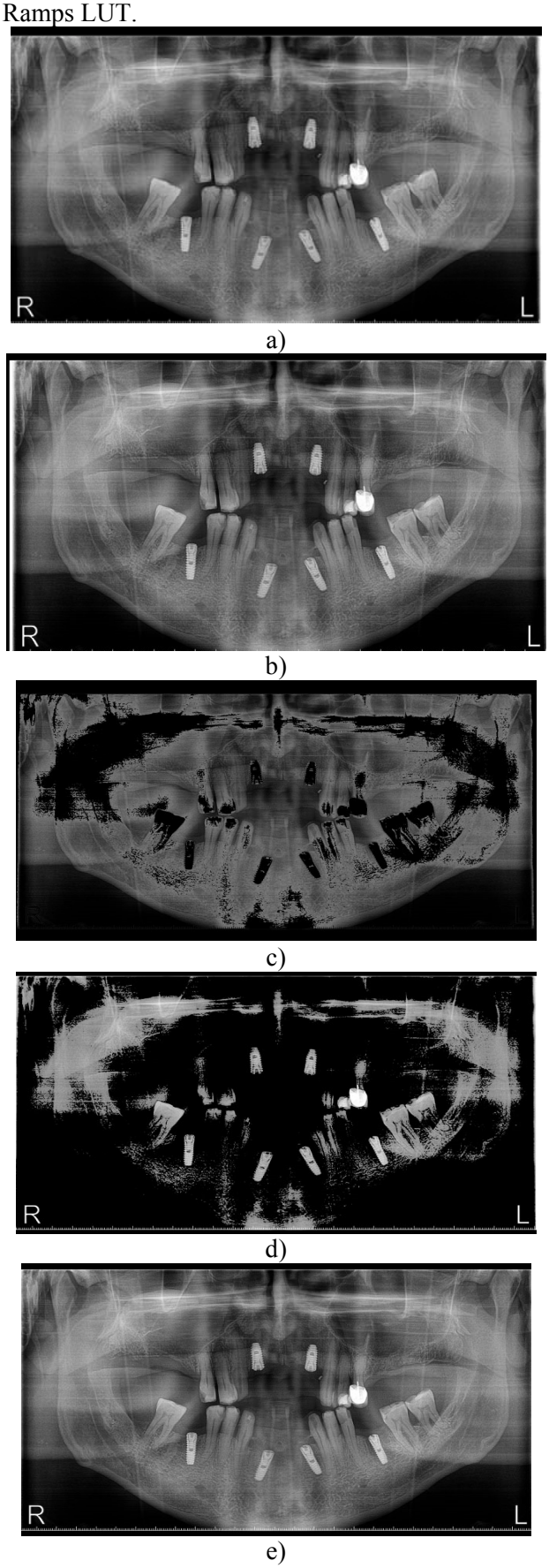
d)

e)

f)

g)

Figure 6. Analysis of dental radiography subjected to the doctor's observation: a) initial radiograph; b) Salt&Pepper filter; c) Low pass filter; d) High pass filter; e) Gray scale filter; f) Contrast stretching; g) Ramps LUT.



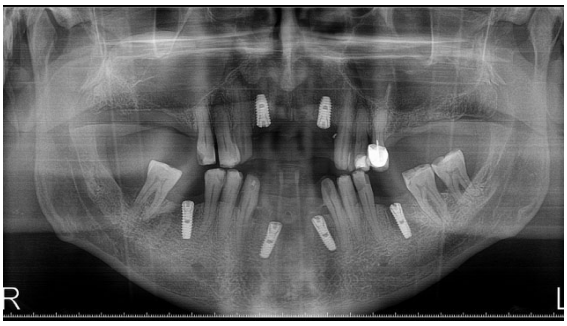
a)

b)

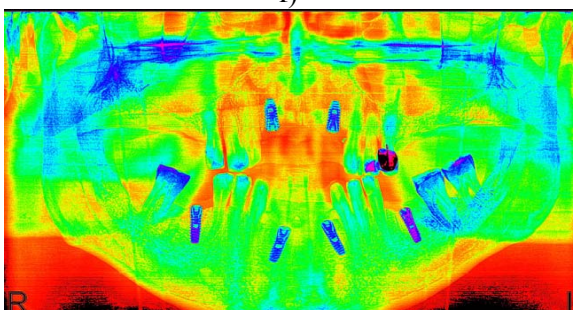
c)

d)

e)



f)

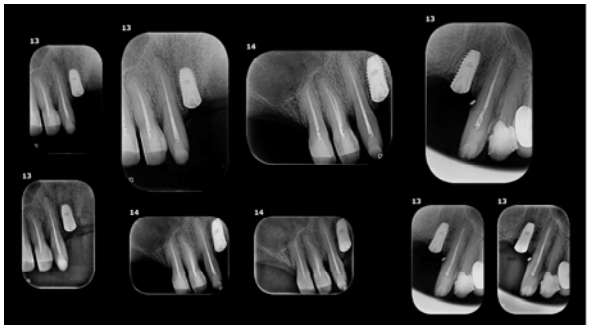


g)

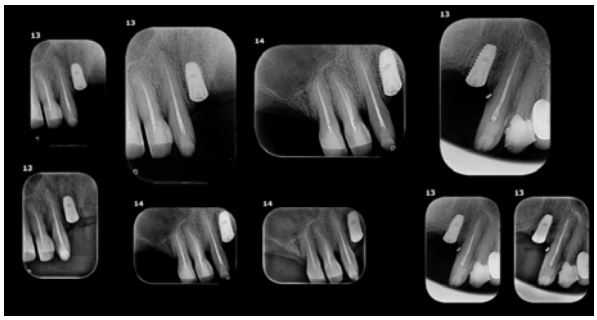
Figure 7. Analysis of dental radiograph in the intermediate phase of treatment: a) initial radiograph; b) Salt&Pepper filter ; c) Low pass filter; d) High pass filter; e) Gray scale filter; f) Contrast stretching; g) Ramps LUT.

In the medical practice of implantology, the maintained dental area is devitalized, as shown in Fig. 8a. In Fig. 8b-8g are illustrated the results obtained by filtering the retroalveolar radiography, after which the doctor can make the decision that certain premolars, although presenting granuloma-like structures, should be maintained and treated.

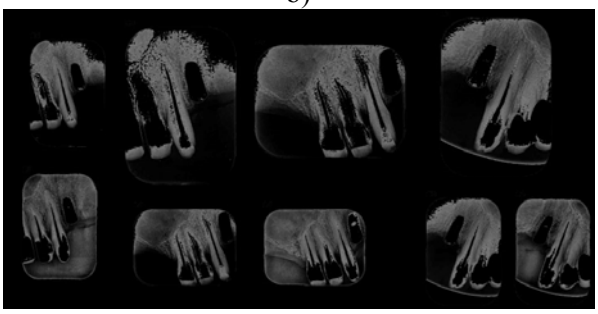
The software user can select the results obtained from a previous examination of that patient, in order to be able to compare the two situations. This allows the dentist to have a better view on the evolution of the dental caries, the structures of the surface of the jaw teeth, the wisdom teeth, the bone loss or any other condition throughout the treatment period, to see if the patient's condition has improved or, on the contrary, worsened.



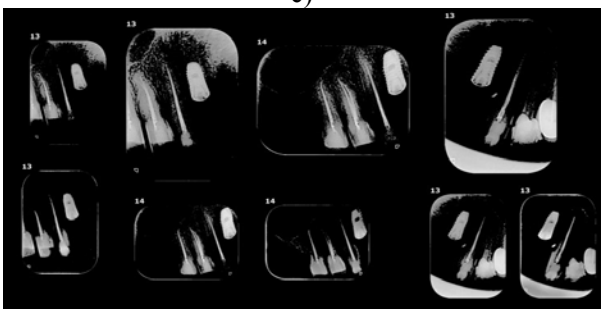
a)



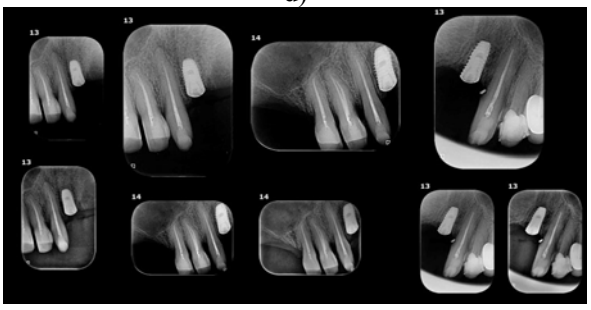
b)



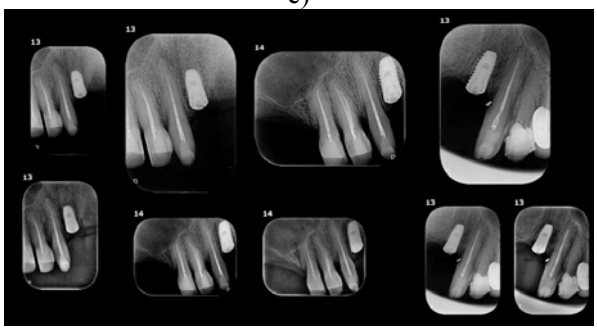
c)



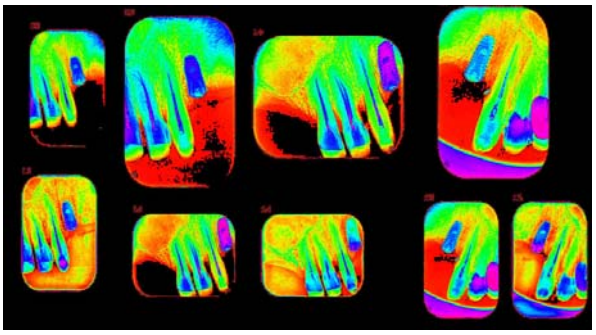
d)



e)



f)



g)

Figure 8. Analysis of the retroalveolar dental radiograph: a) initial radiograph; b) Salt&Pepper filter; c) Low pass filter; d) High pas filter; e) Gray scale filter; f) Contrast stretching; g) Ramps LUT.

Thus, we can conclude that software filtering allows the doctors not only to take the general treatment decisions but also to establish the detailed aspects of the treatment.

6 Conclusions

In the dental field sometimes it is necessary to improve the image quality. The usefulness of the processed radiographic image does not end with the treatment applied, because the dentist can use it for the future controls in order to establish the evolution of the dental condition.

The benefits of digital radiography analysis using the developed software application are: it can be processed immediately (being available for viewing), it can be transmitted in real time, it can be improved with a number of processing techniques, thus obtaining a better contrast and brightness, etc. Through the interface that the application provides, the user can upload a processed and diagnosed image, the result of a previous examination, in order to easily track the evolution of dental caries, structures on the surface of the jaw teeth, the wisdom teeth, bone loss or any other problems in order to establish the efficacy of the treatment to which the patient is subjected.

Depending on the need for analysis and type of dental images, other models of software filters will be implemented.

References:

[1] D.A. Trochesset, R.B. Serchuk, D.C. Colosi, Generation of intra-oral-like images from cone beam computed tomography volumes for dental forensic image comparison, *Journal of forensic sciences*, Vol.59, no.2, 2014, pp. 510-513.

[2] M. Popescu, A. Naaji, Intelligent Image Analysis System for Position Control of People with Locomotor Disabilities*, Human Behaviour Analysis Using Intelligent Systems, Learning and Analytics in Intelligent Systems*, Springer, Vol. 6, 2020, pp. 93-119.

[3] N. Sharma, L.M. Aggarwal, Automated medical image segmentation techniques, *Journal of Medical Physics*, vol.35(1), 2010, pp. 3–14.

[4] J. Agarwal, S.S. Bedi, Implementation of hybrid image fusion technique for feature enhancement in medical diagnosis, *Human-centric Computing and Information Sciences*, vol.5, no.3, 2015.

[5] S. Pezelj-Ribaric, I. Anic, I. Brekalo, I. Miletic, M. Hasan, M. Simunovic-Soskic, Detection of tumor necrosis factor α in normal and inflamed human dental pulps, *Archives of Medical Research*, vol.33(5), 2002, pp. 482-484.

[6] R.B. Ali, R. Ejbali, M. Zaied, Detection and Classification of Dental Caries in X-ray Images Using Deep Neural Networks, *The Eleventh International Conference on Software Engineering Advances*, 2016, pp. 223-226.

[7] F. Gaye, M. Mbaye, B. Faye, B. Diallo, The place of radiography in endodontic treatment carried out in general practice in Dakar, *Tropical Dental Journal*, vol.25(97), 2002, pp. 52-56.

[8] R. Pauwels, J. Beinsberger, B. Collaert, C. Theodorakouc, J. Rogers, A. Walker, L. Cockmartinf, H. Bosmans, R. Jacobs, R. Bogaerts, K. Horner, Effective dose range for dental cone beam computed tomography scanners*, European Journal of Radiology*, Elsevier, 2012, pp. 1-5.

[9] D. Kaur, Y. Kaur, Various Image Segmentation Techniques: A Review, *International Journal of Computer Science and Mobile Computing*, vol.3(5), 2014, pp. 809 – 814.

[10] J. E. Oh, H. S. Cho, D. S. Kim, S. I. Choi, U. K. Je, Application of digital tomosynthesis (DTS) of optimal deblurring filters for dental X-ray imaging, *Journal of the Korean Physical Society*, vol.60, 2012, pp. 1161–1166.

[11] C. Wang, Z. Ye, Salt-and-pepper noise removal by adaptive median filter and TV inpainting, *Journal of University of Science and Technology of China*, vol.38 (3), 2008, pp.282-287.

[12] C.J. De Luca, L.D. Gilmore, M. Kuznetsov, S.H. Roy. Filtering the surface EMG signal: Movement artifact and baseline noise contamination, *Journal of biomechanics*, vol.43, 2010, pp. 1573–1579.

[13] M. Lyra, A. Ploussi. Filtering in SPECT Image Reconstruction, *International Journal of Biomedical Imaging*, 2011, pp. 1-14.

[14] R. Heilbronner**,** S. Barrett**,** *Image Analysis in Earth Sciences: Microstructures and Textures of Earth*, Springer 2014.

[15] F.V. Garrido, F.M. Munoz, *Advanced Techniques in Musculoskeletal Medicine &Physiotherapy. Using minimally invasive therapies in practice*, Elsevier Health Sciences, 2016.