pattern matching, simulator
transoesophageal echocardiography,
computed tomography

Grzegorz BERNADY, Andrzej GACKOWSKI[1], Aleksander KEMPNY[2],
Adam PIÓRKOWSKI[3],

PATTERN MATCHING ALGORITHMS IN PREPARATION OF A 3D HEART MODELS

The CT2TEE is an online transoesophageal echocardiography (TEE) simulator [2] developed on the AGH University of Science and Technology. As a basis for the displayed real-time simulation it uses 3D models created by hand from the CT images.

The aim of this paper is to present algorithms developed specifically for efficient editing and preparation of three-dimensional models of the heart for use in the CT2TEE simulator. A detailed description of proposed algorithms, their advantages and limitations, is provided.

1. INTRODUCTION

Echocardiography is usually performed in a transthoracic (TTE) or transoesophageal (TEE) form. Transthoracic method is non-invasive and can be trained without any harm to the patient. The transoesophageal approach is relatively unpleasant for patients and can cause complications if not done correctly. Difficult training of doctors has resulted in the development of a number of TEE simulators [4,7,9]. In order to popularize this method of training a free online simulator was created - the CT2TEE [6].

This simulator uses data collected during a computed tomography examination to create a three-dimensional model of the heart, which is then used to generate an ultrasound image. In addition, the simulator allows the use of an educational colour layer [8] to visualize selected objects. Development of appropriate model of heart with indexed objects is tedious and time consuming therefore the concept of accelerating this work was born. The authors developed techniques and algorithms that enable efficient editing and preparation of the model which can then be loaded into the CT2TEE software.

2. INPUT DATA

The data used to create 3D models comes from the CT imaging technique. The biggest problem in working with this data set is its large size. Four-dimensional model (describing motion in 3D space) consists of several three-dimensional data blocks. Each block consists of about 250-300 two-dimensional, grayscale images of the heart (Fig. 1). Each image is 512 by 512 pixels in size and has a 12 bit resolution. One 3D image set (data block) takes about 120MB of memory, and a complete 4D model of the heart more than 1GB.

[1] Jagiellonian University, Medical College, Dept. of Coronary Disease, John Paul II Hospital, Kraków, Poland.
[2] Adult Congenital and Valvular Heart Disease Center, University of Muenster, Muenster, Germany.
[3] AGH University of Science and Technology, Department of Geoinfomatics and Applied Computer Science, Cracow, Poland, email: pioro@agh.edu.pl.

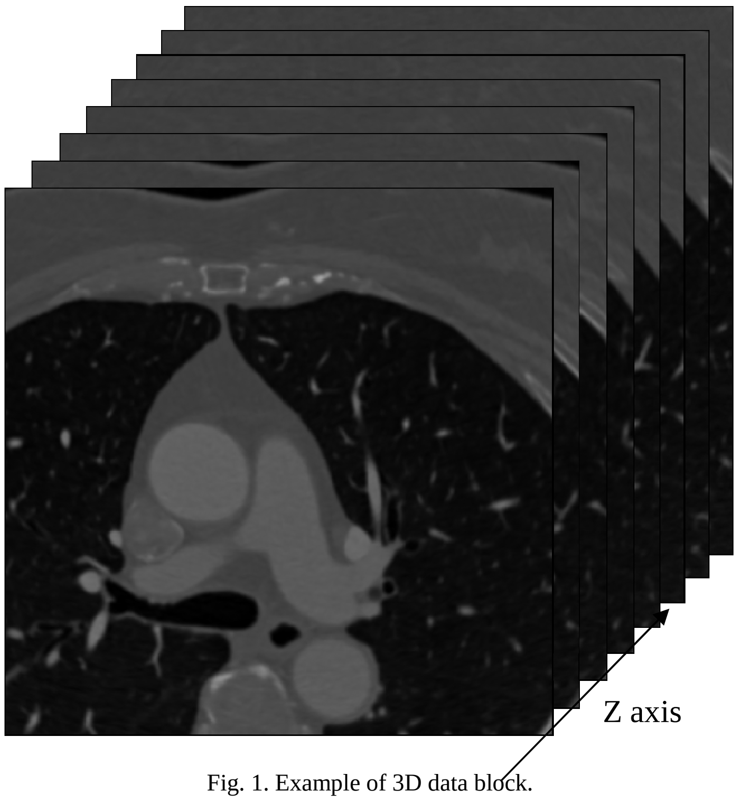

Fig. 1. Example of 3D data block.

This data set differs significantly from the grayscale images seen in TEE, therefore it needs editing. It is necessary to change the brightness of some parts of the model to mimic a real ultrasound data. Furthermore, in order to construct the educational colour layer areas selected by a doctor during editing process must be filled with solid colours. Painted areas can then be indexed and displayed in the CT2TEE simulator. The most important thing here is the ability to select any structure in the model. Because some parts of the image are defined by the same texture with the same brightness, the algorithms must operate interactively thus allowing a very precise selection.

3. 3D INDEXING ALGORITHMS

Selection of tissues in the three-dimensional CT model can be based on the known 3D indexing algorithms. There are many algorithms for indexing 3D [5], some of them are used for visualization of medical examinations [1,2]. To edit a 3D model based on CT scans these algorithms seem not to be useful. In the case of colouring cardiac structures 3D indexing algorithms may not be appropriate. Physician may want to select only the desired portion of the tissue. Another problem might be the concentration of the contrast, where brightness of the blood is the same as the brightness of the bone. Those cases require an interaction with the user. The solution that solves described problem is to create special algorithms dedicated to this issue.

4. DEDICATED ALGORITHMS

The CT2TEE simulator reads data as a series of two-dimensional grayscale images in the DICOM format. Therefore, the developed, dedicated pattern matching algorithm was divided into two independent parts. The first part selects structures on a single 2D slice, and the second automatically finds the desired structure on the following slices in the same data block. This allows the user to quickly choose an interesting object in 2D space and move this selection to a 3D space.

4.1. 2D SELECTION ALGORITHM

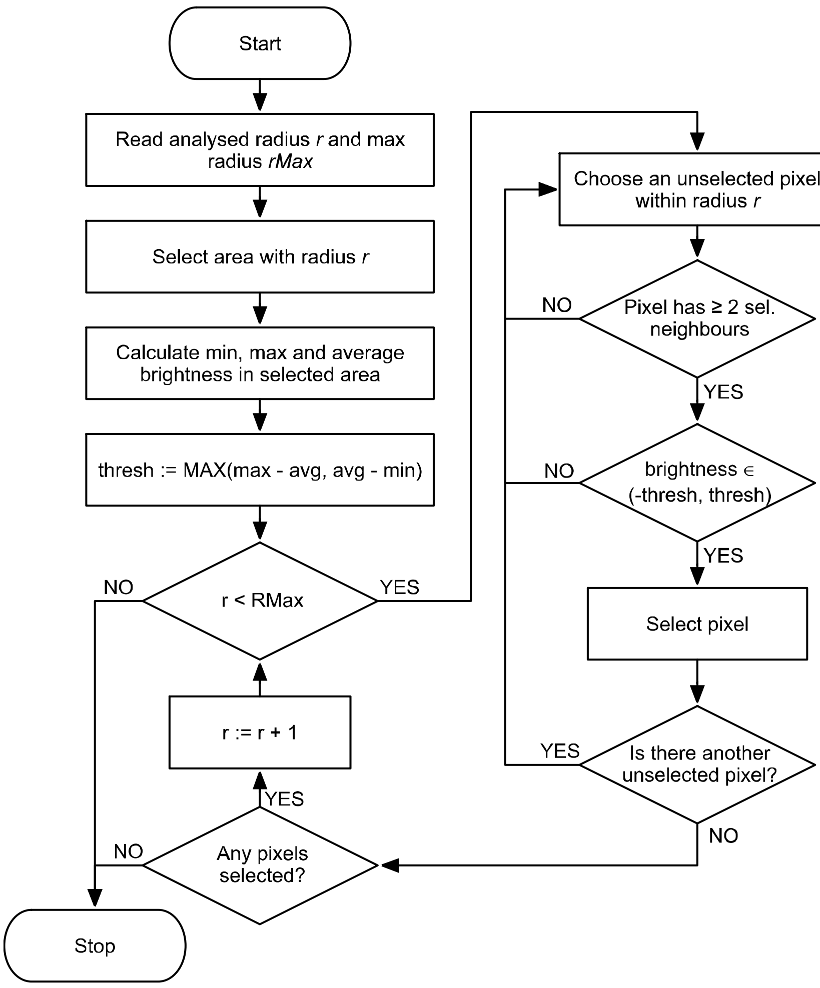
Start
Read analysed radius *r* and max radius *rMax*
Select area with radius *r*
Calculate min, max and average brightness in selected area
thresh := MAX(max - avg, avg - min)
r < RMax
NO
YES
r := r + 1
YES
Any pixels selected?
NO
Stop
Choose an unselected pixel within radius *r*
Pixel has ≥ 2 sel. neighbours
NO
YES
brightness ∊ (-thresh, thresh)
NO
YES
Select pixel
Is there another unselected pixel?
YES
NO

Fig. 2. Flowchart of the 2D selection algorithm.

The proposed algorithm (Fig. 2) [3] as the initial condition requires a starting point, the analysed area radius and the maximum selection radius. The starting point defines the structure to be selected. It also specifies the centre of the areas defined by the radiuses. The analysed area is used to calculate the brightness threshold by the following formula (1).

$$\text{threshold} = \text{MAX}\,(\text{maximum} - \text{average},\ \text{average} - \text{minimum}) \qquad (1)$$

The maximum radius defines the only boundary condition of algorithm. This way the search area is limited to a part of the image thus accelerating calculations. After determining the initial parameters the algorithm selects all the pixels inside the analysed area and calculates the threshold value. In the next step the main loop of the algorithm is run. In subsequent iterations the *r* value is incremented and all pixels within a circle of radius *r* satisfying the required conditions are selected. If in a given iteration no pixels are selected, the algorithm is terminated.

Pixel selection procedure works as follows. In a loop all pixels inside a circle of radius *r* are checked for 8-neighborhood. Candidates for selection are only the pixels with at least two of the eight possible neighbours. Then the algorithm selects those candidates, whose brightness does not exceed the maximum deviation defined by the threshold value. An example of selection process is shown on Figure 3.

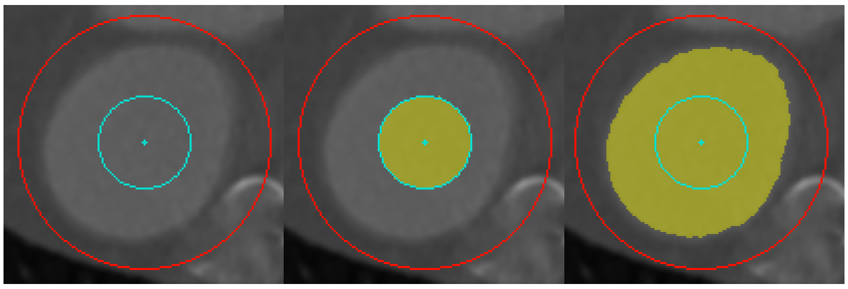

Fig. 3. Example of the selection process. The left image shows a structure with selection cursor over it. On the middle image only analysed area is selected. Finally on the right image whole object is selected. Red circle represents maximum selection radius and turquoise circle represents the analysed area.

The algorithm has several limitations, some of which depend on starting conditions. The first of these limitations is shown in Figure 4. In some cases, part of the object is not selected, because it contains pixels with brightness outside the threshold. It is caused by the fact that only a portion of the texture of the object is analysed when determining the threshold value. It can be easily fixed by setting the threshold manually or increasing the size of the analysed area.

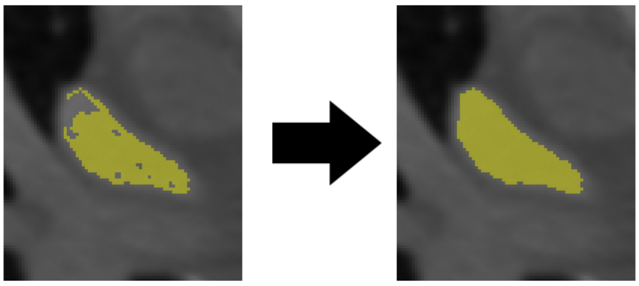

Fig. 4. Problem with holes in the selection (left) and fixed selection (right).

Another limitation is the problem of recognizing the boundaries of objects with similar brightness range, but different shape of the histogram. In this case, selection "leaks" through the observed border between the two objects (Fig. 5). It can be fixed by precisely defining maximum radius value so that the selection area ends exactly on the object's border. Sometimes it is mandatory to divide the selection into several smaller ones that better reflect the shape of the object’s border.

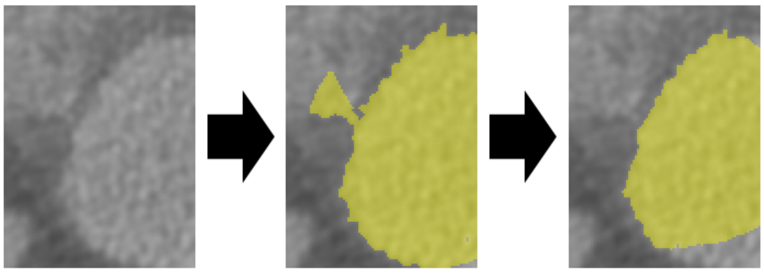

Fig. 5. Problem with selection “leaking” through the object’s border (middle). Structure before selection is shown on the left image and properly selected object on the right image.

4.2. TRACKING ALGORITHMS

To transfer selection from one 2D slice into other slices in the same 3D data block two algorithms were developed [3]. They both operate on a pixel selection array but with different approach to the problem.

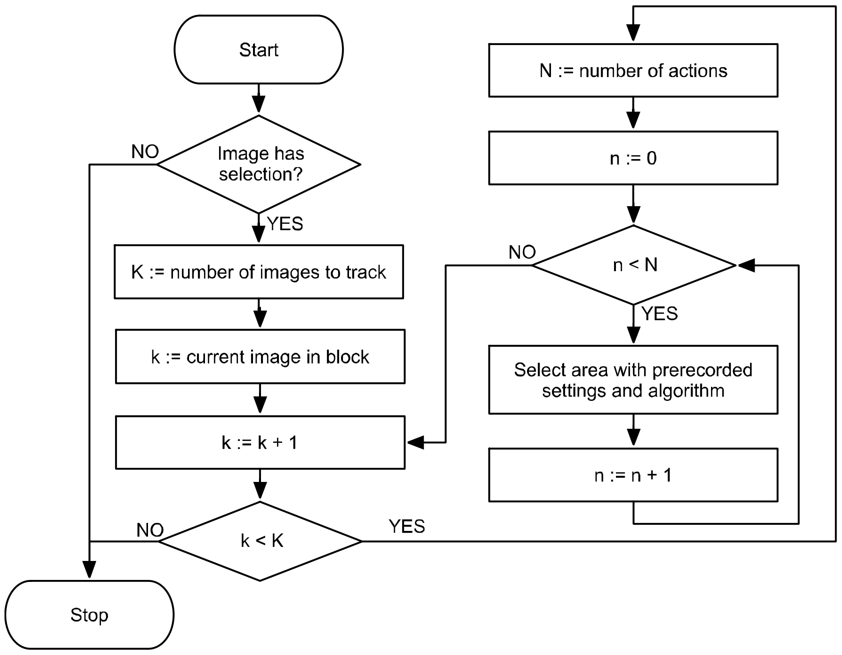
Start
Image has selection?
NO
YES
K := number of images to track
k := current image in block
k := k + 1
k < K
NO
YES
Stop
N := number of actions
n := 0
n < N
NO
YES
Select area with prerecorded settings and algorithm
n := n + 1

Fig. 6. Flowchart of the first tracking algorithm.

The Figure 6 shows the block diagram of the first of the proposed algorithms. The principle of this algorithm is based on recording selection actions on one image and replaying them on the following images in the data block. Therefore it is necessary to remember the initial conditions of each selection made by the user on every image. Additionally the depth of tracking must be provided. It specifies the number of images in which the recorded pattern will be applied.

Implementation of this algorithm consists of two nested loops. The outer loop iterates over the images in data block and the internal loop over the actions recorded in each image.

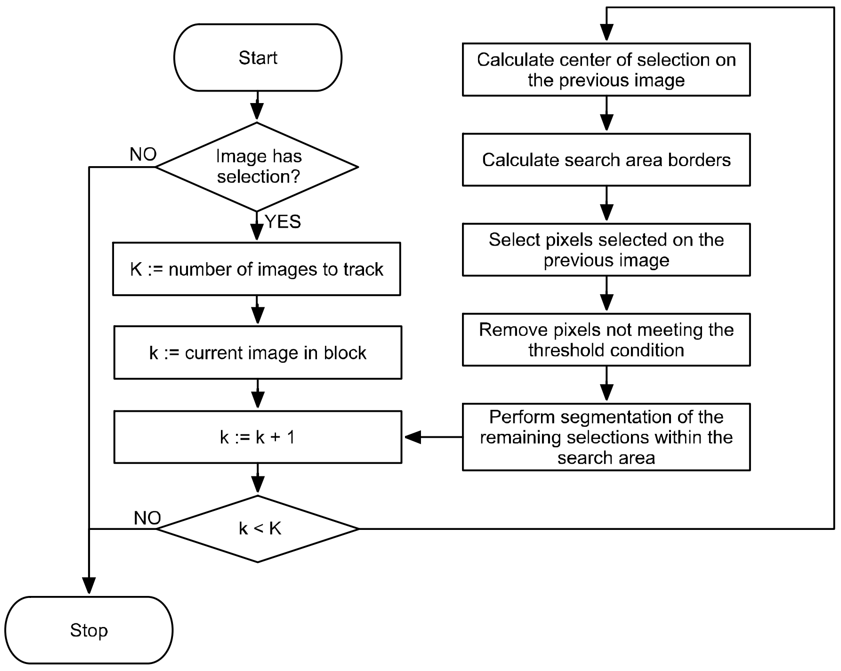
Start
Image has selection?
NO
YES
K := number of images to track
k := current image in block
k := k + 1
k < K
NO
Stop
Calculate center of selection on the previous image
Calculate search area borders
Select pixels selected on the previous image
Remove pixels not meeting the threshold condition
Perform segmentation of the remaining selections within the search area

Fig. 7. Flowchart of the second tracking algorithm.

The second proposed algorithm treats the selection as a whole, rather than a sequence of actions. Figure 7 shows a flowchart of this algorithm. Initial conditions include dimensions of the rectangular pattern search area, tracking depth and threshold defined by the formula (1). The centre of the search area is calculated as the centre of gravity of the selection made on previous image, thus it is determined individually for each image.

Main loop of the algorithm iterates over the images in the 3D data block. At the beginning of each iteration the centre of gravity of the previous image's selection is determined and the boundary conditions are calculated. This way, if the selected object is moving, the search area will be following it. In the next step selection from the previous image is transferred to current slice and all pixels that do not meet the threshold condition are removed. Finally the segmentation of the remaining area is performed filling in parts of the object, which were not selected in the third step.

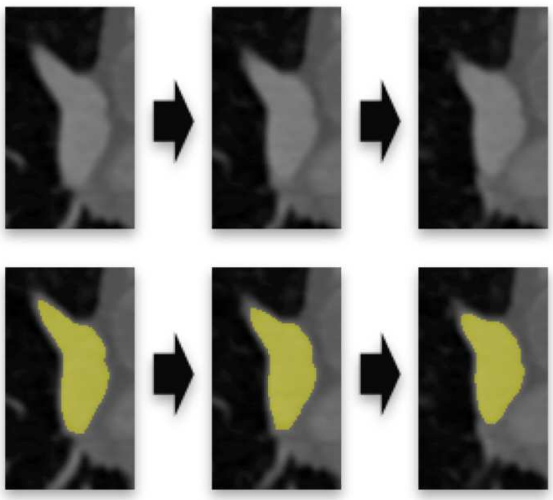

Fig. 8. Original subsequent images of a structure in the model (upper three pictures) and final selection made by any of the above tracking algorithms (lower three pictures).

The final effect of the proposed algorithms is shown on Figure 8. Both algorithms work very quickly and allow for easy selection of structures in the model for later editing. The first one (Fig. 6) is more suited for static objects and introduces a form of automation while retaining all advantages of manual selection. However a problem may arise when object changes shape or position rapidly between subsequent images causing the replayed actions not to land entirely on the searched texture. It can lead to a situation shown on Fig. 9. The problem may be fixed by manually correcting the selection on this image.

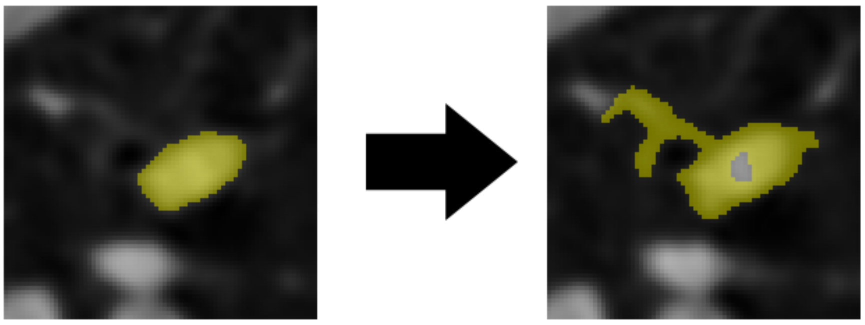

Fig. 9. Selection of pixels not belonging to the object in subsequent image.

The second algorithm (Fig. 7) was developed with moving objects in mind and the problem shown in Figure 9 is non-existent. Whenever a part of the selection hits the object it will be selected properly.

For fast moving objects it is important however to make the search rectangle big enough to cover the whole object in following images because, for performance reasons, all calculations are done within this search area.

4.3. EFFICIENCY OF THE ALGORITHMS

On a test machine (Intel Core i7 3.4GHz and 16GB RAM) both selection and tracking algorithms worked in real-time with CPU usage below 5%. On a virtual machine with single core 2.0GHz CPU and 1GB RAM they performed only slightly slower but CPU usage went to about 80-85%. Due to a very fast performance of the algorithms there was no action taken to parallelize either of them. When using byte array the selection layer takes 262KB of RAM for each image in 3D data block, which gives a memory usage of about 60-80MB for an entire image set.

5. FUTURE WORK

Physicians found that the colouring parts of the heart structures are important for educational purposes. Due to the described application existing input data sets in CT2TEE simulator [6] are improved. There are educational colour layers [8] prepared for all sets. Work is underway to create new sets of important heart defects. In the nearest future work involves an implementation of editing or tracking heart structures in motion. Further development of the project will take place in close cooperation with doctors from the John-Paul II Hospital in Cracow.

6. ACKNOWLEDGEMENT

This work was financed by the AGH University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection as a part of statutory project.

BIBLIOGRAPHY

[1] ANCUTA P. N., 3D Object Modelling and Visualisation Software. The Romanian Review Precision Mechanics, Optics & Mechatronics, Vol. 19, No. 36, 2009.

[2] ANCUTA P. N., 3D Object Modeling and Visualization Software for Surgery Preoperative Plan, 6th Workshop on European Scientific and Industrial Collaboration on promoting Advanced Technologies in Manufacturing, WESIC'08, 2008.

[3] BERNADY G., Pattern search algorithms in digital images, Master's thesis, AGH University of Science and Technology, Department of Geoinfomatics and Applied Computer Science, Cracow, Poland, 2011.

[4] BOSE R., MATYAL R., PANZICA P., et. al., Transesophageal echocardiography simulator: a new learning tool, J., Cardiothorac Vasc Anesth, 2009.

[5] DUGELAY J.L., BASKURT A., DAOUDI M., 3D Object Processing: Compression, Indexing and Watermarking, 2008 John Wiley and Sons.

[6] KEMPNY A., PIÓRKOWSKI A., CT2TEE - a Novel, Internet-Based Simulator of Transoesophageal Echocardiography in Congenital Heart Disease, Kardiol. Pol., Vol. 68, 2010, pp. 374–379.

[7] KUTTER O., SHAMS R., NAVAB N., Visualization and GPU-accelerated simulation of medical ultrasound from CT images. Computer Methods and Programs in Biomedicine, 2009.

[8] PIÓRKOWSKI A., Construction of Educational Color Layer for Echocardiography Simulator, Journal of Medical Informatics and Technologies, Vol.16, 2010, pp. 167-171.

[9] WEIDENBACH M., DRACHSLER H., WILD F., et. al., EchoComTEE - a simulator for transoesophageal echocardiography, Anaesthesia, Academic Press, London, 2007.