Semiautomatic Cardiac CT Scan Classification with a Semantic User Interface

Andreas Elsner[1], Dženan Zukić[2], Gitta Domik[1], Zikrija Avdagić[2], Wilhelm Schäfer[1], Eva Fricke[3], Dušanka Bošković[2]

[1]Department of Computer Science, University of Paderborn, Paderborn, Germany
[2]Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia & Herzegovina
[3]Heart and Diabetes Center NRW, Bad Oeynhausen, Germany

Abstract

In medical volume visualization, one of the main goals is to reveal clinically relevant details from the CT study by classification of the data, i.e. the coronary arteries, without obscuring them with less significant parts. Usually, the classification is carried out by defining multi-dimensional transfer functions which assign specific visual attributes to the voxels which express the features of interest. Unfortunately, this can become a fairly complex task, generally accomplished by trial and error even for the experienced user. Many sophisticated semi-automatic and automatic approaches for volume classification have been published in the past, which rely either on the overall quality of the rendered image or on a general boundary detection between different materials rather than on an insight as to what makes the transfer function appropriate for a specific feature in the dataset. This paper presents an efficient way for automatic transfer function generation based on neural networks. We describe how to use neural networks to detect distinctive features of the volume data and how this information can be used to provide the user with a semantic view on the automatic data classification.

1. Introduction

Direct volume rendering is a powerful technique for visualization of CT data as it has the potential to show the three dimensional structure of a feature of interest, rather than just a small part of the data by a cutting plane. It helps the viewer to get a better insight into the relative 3D positions of the object components and makes it easier to detect and understand complex phenomena like coronary stenosis for diagnostic and operation planning. The latest advances in consumer graphic cards has made this approach even more attractive, as high quality interactive volume rendering is now possible on common low cost hardware.

However, in clinical practice, volume rendering is seldom used. This is due to the fact that a proper classification of the data, which is mandatory to extract the features of interest while suppressing insignificant parts, can become a fairly complex task. CT scans contain a combination of different materials and tissue types with overlapping boundaries. Depending on the actual diagnostic target, some of those features of the dataset must be highlighted, some must be shown transparent to retain the anatomical context while others should become completely invisible to prevent obscuring the more important parts. Without such a proper classification, it is impossible to obtain a successful visualization which helps the user to gain a good insight into the dataset. Furthermore, the user should be able to change the classification interactively according to his/her needs. This should be possible in a semantic way, where concrete features of interest can be selected directly by their name instead of working with a non-intuitive data driven approach.

For volume classification, transfer functions are typically used. Their role is basically to assign specific visual attributes like color and opacity to the features of interest. The simplest approach is to use a one-dimensional transfer function, where the voxel data is the only variable to which these visual attributes are assigned. But this method fails in most cases to omit interference of other anatomical structures, since these may contain the same range of scalar values making it impossible to differentiate between them.

Two-dimensional transfer functions solve this problem by adding an additional dimension to the transfer function domain. By using a classification based on a combination of properties, different features can then be visualized separately from each other. Therefore two-dimensional transfer functions are of great importance for the classification process of medical data.

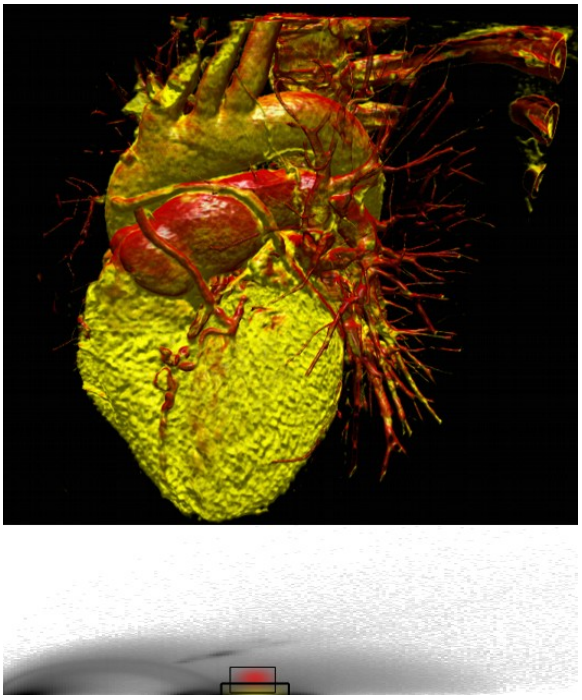

Figure 1: *(a) above: Volume rendering of a cardiac CT scan classified with a 2D transfer function. (b) below: 2D histogram based on attenuation coefficient and gradient with filters.*

Unfortunately, specifying two-dimensional transfer functions can become a fairly complex and time-consuming task even for experienced users like visualization experts. First, the user must handle additional parameters for the second dimension, thus making the editing process more difficult. Second, the user must set the transfer function in the rather abstract data analysis domain while observing the result in the rendered image (visualization domain). An example of a volume rendering of a cardiac CT scan classified with a 2D transfer function is shown in Figure 1. The situation is made worse by the fact that small changes to these parameters tend sometimes to cause large and unpredictable changes to the visualization. Thus, to avoid editing based completely on trial and error, it is necessary to identify features of interest in the data analysis domain first, requiring a good understanding of the data values and their relevance for a certain feature. Additionally, the manual assignment of optical properties might be non-standardized but rather influenced by the personal taste of the user. This can cause misinterpretation of the data if the visualization is later on reviewed by others. Also, this often leads to results which are hardly reproducible and hence not suitable for clinical practice. Finally, radiologists have a limited time allowed for diagnostics per patient, which makes two-dimensional transfer functions impractical to use without proper guidance through the process or a semi-automatic transfer function generation.

Besides manual transfer function creation, semi-automatic and automatic methods have been proposed in the past as we will discuss below in more detail. These can be categorized into two kinds of approaches: (1) Image-driven methods focus on the overall quality of the rendered image and rely often on user interaction to decide which transfer function setup is the most appropriate of a given, pre-generated set. (2) Data-driven approaches try to detect certain features and their boundaries automatically by analyzing the volume data with different algorithmic methods. However, both approaches have in common that they are not purposive enough when a specific feature of the dataset should be visualized, as it is required for clinical routine.

In our work we present a new approach for two-dimensional transfer function generation based on neural networks. We describe how to train neural networks to detect distinctive features of the volume data and how this information can be used to provide the user a semantic view on a semi-automatic data classification. Because histograms of same scan type (e.g. cardiac scans) have similar structures, neural networks can be trained to position filters on features of interest according to the diagnostic target. Although this technique is flexible enough for classification of different types of CT datasets, in this paper we focus on heart scan visualization to detect coronary diseases.

2. Related Work

Different approaches for semi-automatic or automatic transfer function generation have been proposed recently. He et al. [HHKP96] describe a method based on stochastic search techniques, where the user selects transfer functions from thumbnail images. For each selection after the first one, which is either generated randomly or given as a preset, a heuristic search is performed until the selection converges with the search results. Another interactive approach is presented by Marks et al. [MAB*97], which uses an interactive user interface called Design Gallery for setting the visual attributes. The gallery consists of different renderings created by a broad variation of the transfer functions parameters. From this selection the user can choose the one with the best visual appearance. Both techniques rely on the quality of the rendered image rather than on an insight as to what makes the transfer function appropriate for a specific dataset. Also, both approaches support only one-dimensional transfer functions, while multi-dimensional approaches have proven superior for effective classification.

Therefore, Kindlmann et al. [KD98] use the first and second derivative of the data value to create a three dimensional volumetric histogram for a mathematical analysis to perform an automatic boundary detection. But this approach is less feasible for clinical practice, since no information about the features of interest is used to generate the transfer functions. Salama et al. [SKK06] also proposed a technique which introduces a semantic model to the trans-

fer function setup. In this case, the authors use principal component analysis to decide which parameters of a transfer function show the biggest variance between different datasets of the same type. The first principal component is then used to adjust the transfer function to its individual dataset, while additional properties (like color, visibility) can be changed to increase the overall visual quality. To create a semantic view, the principal component analysis is performed for every feature of interest. This information is then incorporated in a user interface, grouping the different parameters regarding to the features name. In our approach, we borrow the idea of semantic categorization of transfer functions, while replacing the manual adjustment of parameters with an automatic approach for transfer function generation.

Hence, closer to the method presented in the current paper is the technique described by Tzeng et al. [TLM03]. The system described in their paper generates high-dimensional transfer functions interactively as the user selects relevant and irrelevant parts by "painting" on a set of slices from the volume dataset. A neuronal network uses the painted regions as training data to learn which voxels have characteristics that might be of interest. Based on this information, opacity functions are created to classify the whole dataset. As input, this approach uses the voxel's value, gradient, values of its neighbours and its location. While this method focuses on evaluating the properties of each voxel in one dataset, the importance of the current paper is that while these values vary between different datasets, the two-dimensional histogram contains structures that can be still used to identify features of interest by neural networks.

3. State-of-the-Art

3.1. Volume Rendering

A lot of progress has been made in the development of consumer graphics hardware, which – mainly designed for computer video games – also makes it possible to accelerate volume rendering to achieve interactive frame rates. Among the different approaches for direct volume rendering including ray-casting, the shear-warp algorithm and texture based methods, the latter one has proved superior for CT data visualizing. It combines high quality images and most efficient use of hardware acceleration techniques [EHKR06a]. These approaches take advantage of the graphic cards support of bilinear and trilinear interpolation, required for the resampling step during volume rendering.

In the field of texture-based techniques, two methods exist. The 2D texture-based approach uses three copies of the volume data which reside in texture memory. Each copy consists of a fixed number of slices along a major axis of the dataset which will be addressed depending on the current view direction. After bilinear interpolation, the values of the slices will then be classified through a lookup table, rendered as a planar polygon and blended into the image plane. This method often suffers from artefacts caused by the fixed number of slices and their static alignment along the major axes. Alternatively, hardware interpolation can be used to generate intermediate slices along the slice axis to achieve better visual quality. But this would, in addition to the redundant data storage, further increase the amount of required graphics memory. Modern graphic cards support 3D texture mapping which allows storing the whole dataset in one volumetric texture object. It is then possible to sample view-aligned slices using trilinear interpolation. Thereby the artefacts which occur when 2D texture-based techniques switch between the orthogonal slice stacks can be avoided and an arbitrary sample rate can be chosen, which results in an overall better image quality. Since the 3D texture consists of a RGBA quadruplet for every texel, also in addition to the scalar value the pre-calculated gradients for multi-dimensional transfer functions can be stored without additional memory consumption.

Therefore in our approach, 3D texture mapping is used. However, the method described in this paper is independent of the technique used for volume rendering, as both 2D and 3D methods use transfer functions to map scalar values to visual attributes.

3.2. Classification

Besides the visual quality achieved with the rendering process itself, the most important task for a successful visualization is to find a good classification technique that captures the features of interest while suppressing insignificant parts. This step is typically accomplished by transfer functions, which assign renderable optical properties like color and opacity to each voxel of the dataset. A straightforward approach to achieve this goal would be to use 1D transfer functions, which consider only the voxel's scalar value to handle this task. For medical datasets, this approach is in most cases of limited effectiveness, because the materials and tissue types which are to be separated might have overlapping intervals of scalar values, making 1D transfer functions unable to render them in isolation. When a visual attribute is assigned to such an interval, all corresponding voxels are equally visualized, regardless to which anatomical structure they belong.

Multi-dimensional transfer functions classify the volume not just on the data values but on a combination of different properties. This additional information makes it then possible to differentiate between the features of the dataset. Kniss et al. presented a method for manual multi-dimensional transfer function generation based on the data values and its derivatives [KKH02]. The gradient is useful as an additional criterion for classification since it discriminates between homogenous regions inside a structure and regions of change at the boundaries. Also, the gradient can be used to apply illumination to the volume visualization which improves depth perception. In the past, we demonstrated the importance of two-dimensional transfer functions which, in addition to the attenuation coefficient, also take the corresponding gradient magnitude into account for visualization of cardiac CT studies in combina-

tion with PET. This approach performed well to show myocardial perfusion and location of stenosis of the coronary arteries [FSW*07]. For this purpose, the software "VolumeStudio" was developed, capable of visualizing both modalities, PET and CT.

The manual transfer function generation can be performed in a visual editor, which illustrates the distribution of tuples of attenuation coefficient and gradient magnitude of the dataset in a logarithmic scaled joint histogram. The attenuation coefficient is shown on the x-axis, the gradient magnitude on the y-axis. An example is given in Figure 1.

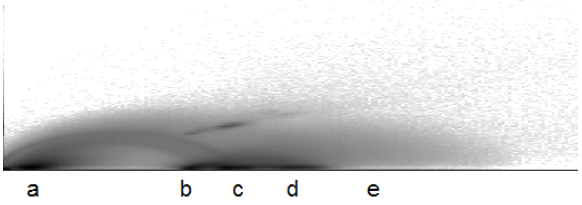

Figure 2: *2D joint histogram of attenuation coefficient versus gradient magnitude. Regions a) – e) identify different materials.*

Different materials and boundaries can be identified in the histogram. Homogeneous regions appear as circular spots at the bottom, as their gradient magnitude is very low. Five basic regions are shown: a) air, b) soft tissue like fat and skin, c) muscles, d) blood (with contrast agent) and e) bones. Arches between these regions represent the boundaries of different materials, where the gradient magnitude reaches its highest values. By using a two-dimensional transfer function, which classifies the data based on attenuation coefficient and gradient magnitude as shown in the histogram, each of these features can be visualized separately from each other.

To create such a transfer function, the user places filters inside the histogram, shown as rectangular areas. Each filter assigns color and opacity values to the voxels of the dataset which are represented by the tuples of attenuation coefficient and gradient in the histogram inside the defined area. The filter size and position can be changed, also its color and opacity distribution. Additionally, the shape of the filter kernel can be altered between different types like Gauss or sine. If multiple filters are set, their color and opacity values are blended together. The above editing can be done interactively in the data analysis domain, while the user can decide by observing the visualization in the visualization domain if the current setup is appropriate or not. This feedback is especially useful for the initial exploration of the dataset to get an overview of which parts in the histogram correspond with the features to visualize. Kniss et al. also presented other kernel shapes for transfer function generation, our experience, however, has shown that the two described above are sufficient for the visualization of cardiac CT scans.

Figure 1 shows a volume rendering of a cardiac CT scan and the transfer functions used. It consists of two gauss filters: The first one colored in yellow is located between the regions c) and d) (compare Figure 2) to visualize the myocardial muscle (heart muscle) and the coronaries (by contrast agent). The second one resides at the top of the first filter, enhancing the contrast between myocardium and coronaries by coloring the properties that represent the boundaries of the contrast agent in red. The coloring was chosen to get a high contrast between the myocardium and the coronaries.

For an experienced user, the distinctive features of the distribution shown in the histogram provide useful information about the features metrics, thereby guiding the transfer function generation. But even with these hints, this is a time-consuming iterative process. The user has to explore the dataset by defining filters and move them to possible interesting locations on the histogram. Once a feature of interest is identified, the parameters for the filter size, location, filter kernel shape opacity and color have to be optimized to match the user's needs until all features of interest are made visible.

3.3. Neural networks

An artificial neural network is a software tool to estimate relationships in data and can be used for function approximation, classification, estimation, pattern recognition and simulation purposes. Unlike rule-based algorithms, neural networks learn by being repetitively trained with examples of the data to be differentiated. During a training phase, it gains knowledge about the relationship between data given as input and the desired output. After the training is completed, the neuronal network applies its knowledge to estimate the solution for problems of similar kind. More generally, it has the ability to generalize solutions about imprecise input data.

The design used for neural networks has its roots in our understanding of the human central nervous system and its way of processing information. Therefore, it relies largely on parallel processing and weighting of information. Basically, a neural network consists of neurons which are grouped into different layers and are capable to process multiple inputs, to a single output (see Figure 3). The different layers are interconnected to each other where each input of a neuron in the current layer is connected to the outputs of the previous. Each connection has a weight which describes the relevance of the information across this connection, sometimes also referred to as synaptic strength. For each neuron, these weighted inputs are summed and then transformed via a nonlinear activation function to form the output. Training is performed by modifying the weights until the output matches the desired value for a given input. Hence, to train the network a so called training set of matching input and output values is required, which represents the relationship the neural network should later be able to approximate. Starting with random weights, training is performed iteratively until the output error is minimized. Once training is completed, the

network can be used to process data beyond the training set.

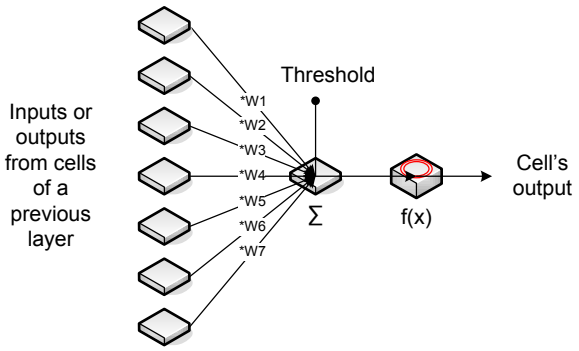

Figure 3: *Neural cell - neuron*

For our approach, we use the feed-forward back-propagation network architecture developed in the early seventies by Werbos and later redeveloped by Rummelhart and others [RHW86]. While advantageous as they are fast to execute, the major disadvantage of feed-forward back-propagation networks is that no fast training algorithm has been developed down to the present day and therefore can be extremely slow to train. However, as training is performed only once in most cases and because of its good performance as universal function approximator, it is the most common and widely used type of artificial neural networks for data classification and prediction. Therefore, we have isolated this specific network type as the neural network we have used for our approach.

The topology of a feed-forward network usually consists of at least three layers in which the neurons are arranged. The first layer is the input layer, where each input of the data to be processed is assigned to one neuron. The output layer is the last layer of the neuronal network, which holds a number of neurons equally to the number of desired outputs. In between the input and the output layer resides one or more hidden layers, where each neurons input is connected to all outputs of the previous layer. Similarly every output is fully connected to the nodes of the next layer. All connections are made in a feed-forward fashion, hence there are no loops in the network and the information processed is only allowed to travel in one direction.

The output y of each neuron is computed by $y_j = f(\theta + \sum_i w_{ij} * y_i)$, where i is the previous layer, j is the current layer, f is the activation function, θ is the threshold (if implemented in network) and w is the weight. As no weight affects the input layer, we use therefore instead $y_i = f(\theta + x_i)$ were x denotes the input to the network.

In order to train the neural network by example, training sets of known input-output data points must be assembled. Training itself is then performed by the back-propagation algorithm, which adjusts the weight of the connections in the network to minimize square error between actual output and desired output. For a previously untrained network, first all weights are set at random. Then the algorithm propagates the input of the training set through the network and compares the actual output of with the desired output of the training set. The difference is called output error and is calculated for each output in the output layer as error delta $\delta_f = y'_f * (desired_f - y_f)$, where y'_f is first derivative of the activation function in that output neuron for the point of its output. The need to calculate derivatives in the process of back-propagation explains the typical choice of the activation function as

logistic ($\frac{1}{1+e^{-x}}$) or hyperbolic tangent ($\frac{e^{2x}-1}{e^{2x}+1}$).

The main reason is that derivatives of these functions can be calculated using the value of the function. For logistic function shown in Figure 4, which was used in our neural network, equality holds: $y' = y * (1 - y)$. Activation functions have to be differentiable in order to apply back-propagation.

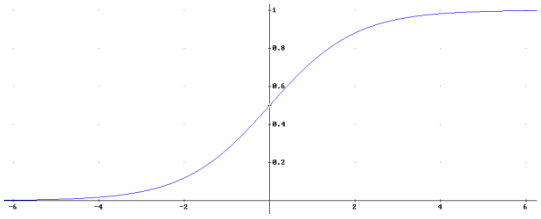

Figure 4: *Logistic activation function*

After the error deltas in the output layer are computed, we can update weights leading into it, using the following rule: $w_{ij} = w_{ij} + \eta * \delta_j * y_i$, where η is the learning rate (small value, usually between 0.05 and 0.3), δ_j is the error delta of the neuron weight is leading into and y_i denotes the output of the neuron weight is coming from (output produced in the feed-forward process). This rule is subsequently used for all weights. Weights can be updated individually (called on-line learning) or in groups (called batch learning). In batch learning, we calculate weight deltas for all samples, and then combine them in one single update. Batch update is usually better, as it lowers oscillations of the weights.

For all layers except output, error deltas are calculated using rule: $\delta_i = y'_i * \sum_j w_{ij} * \delta_j$, where i is current layer, j is next layer. We proceed backwards to the first layer calculating error deltas and updating weights. When the weights between 1st and 2nd layer are updated, back-propagation is finished. This process is repeated many epochs until satisfactory results are obtained. Training can stop when the mean square error (MSE) obtained is less than a certain limit, or we have reached some pre-set maximum number of training epochs.

During the training process over-training of the network should be avoided, as it lowers predictive abilities of the network. Over-training might occur if the network is trained on a training set that contains very similar datasets

or when the lower mean square error limit is not set properly. In this cases, the networks starts to focus its evaluation on irrelevant details of the training set, loosing its generalization ability. For example, a network for voice recognition can be over-trained with a training set consisting of sound samples spoken by only a few people. If the network is presented by a person with a different tone of voice or accent, it can perform poorly because it rather identifies the speaker than the sound. If this problem does occur, a lower mean square error limit should be set and a more varying training set should be used.

To test the neural network performance, it is usually tested against a validation set which is not part of the training set. As the network is unfamiliar with this data, it can be seen as a representative of the general case. Therefore, if the validation succeeds, it can be expected to perform well in all other cases, too. However, if validation fails and considering the training set is properly chosen, the networks topology like the size and number of the hidden layers should be changed in order to achieve better results.

4. Semi-automatic classification with neural networks

As stated in Section 3, the 2D histogram showing the distribution of tuples of attenuation coefficient and gradient magnitude of a heart dataset contains distinctive features which can be used to guide the transfer function setup. These features consist of circular spots at the bottom of the histogram representing homogeneous materials and arches which define material boundaries. Hence, the position and size of a filter setup for a 2D transfer function depends on those patterns. Given as an input, the histogram can be used to train a neural network for pattern recognition. Therefore the user creates filter setups for a training set manually according to the diagnostic target. The network is then trained to associate outputs (filters) with input patterns in the histogram. This time consuming step has only to be performed once and can be done outside clinical practice by a visualization expert in cooperation with a radiologist to ensure the quality of the classification. Once the network is properly trained, it can be used to create an appropriate filter setup automatically.

The 2D histogram is basically a greyscale image with an initial resolution of 256*256. Without further preprocessing, an input of this size would require a significant amount of memory for storage of the neural network (16MB just for weights in case of 64 neurons in 2^{nd} layer). Also, training of a network of this size would be slow, and its generalization abilities would be presumably low.

Therefore, as a pre-processing step, the input to the neural network must be reduced to data that is relevant for the pattern recognition. First, we removed those parts of the histogram that contain no data at all. Since the size of the used part in the histogram varies from dataset to dataset, we estimated a maximum size based on the training set and use this as a fixed value for all datasets of the same type. For the cardiac CT scans used to evaluate our approach it is sufficient to remove the upper half of the histogram and only take the lower one into account. Second, we downscaled the remaining histogram averaging by a factor of 4 as shown in Figure 5. This reduces the number of inputs to the neural network to just 2048. Furthermore, the downscaling of the image smoothes out parts of the histogram which lie outside the distinctive features required for pattern recognition. Since these parts consists of tuples of attenuation coefficient and gradient magnitude which have only a few voxels of the dataset assigned to them, they appear to the neural network as noise. Also, these parts vary a lot between different datasets. As this affects the learning rate, with noise removed and image size reduced, the neural network will learn more easily and will have better generalization abilities.

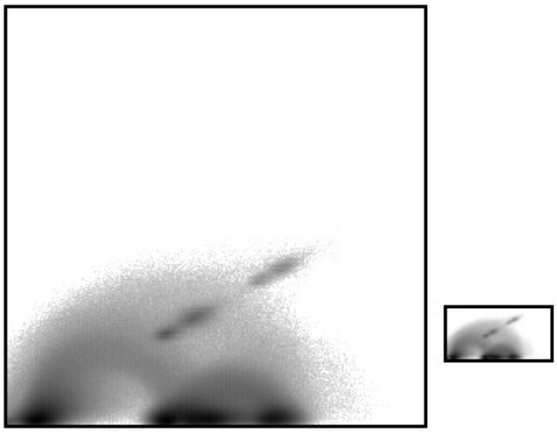

Figure 5: *Original and reduced histogram*

For visualization of the myocardium and coronaries, two Gauss filters are required for proper classification. Each filter consists of four varying parameters, its width, height and its x, y location. As these form the output of the network, its dimensionality is 2 x 4 = 8. Additional information set during the training process, like the color of the filters and its kernel type, is stored separately from the network, as these parameters don't change between different datasets of the same type.

Regarding the size of the hidden layer, we initially started with 64 neurons and then reduced the size step by step to 16. As we noticed no degradation in results, we kept this size for the hidden layer as it increased both learning and execution speed.

The approach described in this paper has been evaluated using real patient cardiac CT data sets recorded by a 16-slice-CT Siemens Somatom Sensation spiral scanner. The slice resolution was set to 256x256 with a varying number of slices to match the patient's cardiac region. To test our approach, we manually determined positions of filters for the heart and coronaries on 14 samples. Four of the samples were randomly marked as testing sets, the other 10 were used for training. We created 5 neural networks and trained the first one with 2 samples, the second one with 4 samples and the fifth one with 10 samples. On all of these

networks we used the testing set to compute the mean square error as shown on Figure 6.

For the network trained with 6 samples, the mean square error is lower on the testing set, than on the training set. More precisely, the training mean square error increases by 0.00075 between the networks trained with 4 and with 6 samples. After that, with further increased number of training samples, the MSE gradually decreases again. We assume that this is caused by the 5th sample training data, which histogram is substantial different from the others, so the network could not easily minimize the errors that its oddity produces. As the number of samples increase, relative influence of that sample is reduced and MSE is lowered again.

The result of our tests demonstrates that even small number of training samples produce good results for automatic transfer function generation. In our measurements, networks trained on 8 and 10 samples provide nearly the same low mean square error as network trained with just 6 samples. This is due to the fact that we used a network with a small hidden layer that leads to fast learning capabilities and the histograms have a very typical pattern structure, so just 6 training samples suffice for good recognition results. Rather, additional knowledge gained to the network by additional training might be annihilated by over-training. Hence training the network beyond this point achieves very little effect and should be avoided.

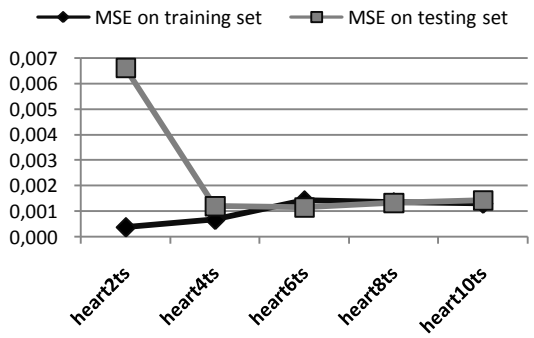

Figure 6: *Mean square error*

5. Semantic User Interface

User interfaces for manual transfer function generation are often based on a data-driven approach, where the user has to manipulate filters represented by simple shapes directly in the histogram to create the desired classification. During the exploration of the dataset, the classification has to be adopted to match the structures of interest shown by the visualization with the actual diagnostic target. This includes increasing or decreasing the visibility of a certain structure to make occluded features visible while retaining the anatomical context or highlighting parts to make them easier to distinguish from others. All these steps rely either on a fine-tuning of the transfer functions or even make it necessary to start the editing process out from scratch, which is very time-consuming and unintuitive for the end-user.

As the neural network is able to detect patterns in the histogram that are assigned to concrete features for the trained dataset, we are able to introduce a semantic level where this changes can be made by the name of the features instead of working with a non-intuitive data driven approach. Therefore, after determining all relevant features of the data in the training set and creating appropriate filters for each of them, we group the filters by the features name. Equally, the corresponding outputs of the neural network that calculate the parameters for those filters after the training phase are grouped together.

After the training phase, the result of the classification performed by the neural network is presented in a hierarchical view, from where the user can modify the visual properties of the different filter groups (see Figure 7). This includes the feature's opacity, color and its relevance as a focus or non-focus object. The relevance parameter affects color and opacity values for all filter groups based on a set of templates to optimize the visualization to match a certain diagnostic target. For example, to gain a better view on the coronaries of a cardiac CT dataset, lungs and bones are made invisible and the saturation and opacity of the myocardium is reduced to gain a better contrast while retaining the anatomical context. The visualization can therefore easily be adapted to the user's needs without knowledge on transfer-functions or data classification techniques.

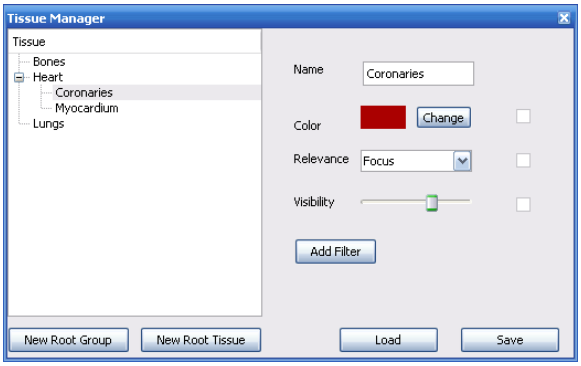

Figure 7: *User Interface for a classification of a cardiac CT scan.*

6. Conclusion and outlook

This paper presents an approach for automatic two-dimensional transfer function generation that incorporates a semantic view to show different features of interest regarding the diagnostic target. The classification itself is performed automatically based on neural networks trained to detect patterns in the two-dimensional histogram of the data. The result is presented on an intuitive user interface that allows the user to change the visualized features by selecting them by name from a hierarchical semantic view. Also, the user can easily switch between focus and non-

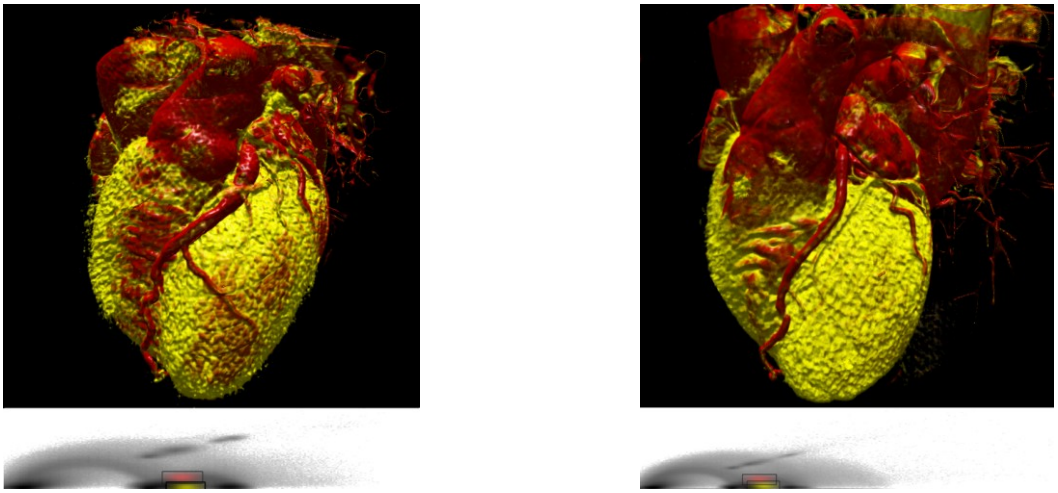

Figure 8: *Two cardiac CT datasets of the testing set. The 2D histograms below show the position of the transfer functions filters after classification with the neural network.*

focus features to retain the anatomical context or change the visual attributes used for visualization. This can be done without any knowledge on transfer-functions or data classification techniques. Thus, the time spent on transfer function creation has been cut down from 1-3 minutes needed even by experienced users to approximately 5-20 seconds for setting the desired features from the semantic view after automatic filter generation, giving doctors more time to analyze the data. Also, as this feature reduces the need for an in-depth understanding of the data analysis domain, it allows a wider range of doctors as end-users to medical software to experiment with patient's 3D data.

As future work, we would like to perform a deeper analysis how modular neuronal networks would even increase the overall classification performance. In contrast to fully connected networks, modular networks consist of fully connected groups of neurons that compute their result independent of the others. This further enhances the networks ability for generalization.

7. References

[Avd03] AVDAGIĆ Z.: Artificial Intelligence & Fuzzy-Neuro-Genetic, Grafoart Sarajevo, 2003.

[EHKR06] ENGEL K., HADWIGER M., KNISS J., REZK-SALAMA C.: Real-Time Volume Graphics. Eurographics Tutorial (2006).

[EHKR06a] ENGEL, K., HADWIGER, M., KNISS, J., REZK-SALAMA, C., WEISKOPF, D.: Real-Time Volume Graphics. AK Peters, Ltd, Wellesley, Massachusetts (2006).

[FSW*07] FRICKE H., SCHWIER M., WEISE R., ELSNER A., FRICKE E., DOMIK G., BURCHERT W.: Coregistration and visualization of cardiac CT studies and dynamic PET studies using scene-graphs, direct volume rendering and 2D transfer functions. In Proc. of the 54th SNM Annual Meeting, vol. 48, May 2007.

[HHKP96] HE T., HONG L., KAUFMANN A., PFISTER H.: Generation of Transfer Functions with Stochastic Search Techniques. In Proc. IEEE Visualization, 1996.

[KD98] KINDLMANN G., DURKIN J. W.: Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. ACM Symp. On Volume Visualization, 1998.

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multi-Dimensional Transfer Functions for Interactive Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, 8(3): 270-285, July, 2002.

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design Galleries: A general approach to setting parameters for computer graphics and animation. In Proc. of SIGGRAPH '97, 389-400.

[RHW86] RUMELHART D., HINTON G., WILLIAMS R.: Learning representations by back-propagating errors. Nature, vol. 323, pages 533 – 536, October 1986.

[SKK06] SALAMA C. R., KELLER M., KOHLMANN P.: High-Level User Interfaces for Transfer Function Design with Semantics. In IEEE Transactions on Visualization and Computer Graphics, vol 12, no. 5, 2006.

[TLM03] TZENG F., LUM E., MA K.: A Novel Interface for Higher-Dimensional Classification of Volume Data. In Proceedings IEEE Visualization 2003, pages 505-512, 2003.