Automatic Hepatic Vessel Segmentation Using Graphics Hardware

Marius Erdt[1], Matthias Raspe[2], and Michael Suehling[3]

[1] Fraunhofer Institute for Computer Graphics, Cognitive Computing & Medical Imaging, Fraunhoferstrasse 5, 64283 Darmstadt, Germany
marius.erdt@igd.fhg.de
[2] University of Koblenz-Landau, Institute of Computational Visualistics, Universitaetsstrasse 1, 56070 Koblenz, Germany
[3] Siemens Medical Solutions, Computed Tomography: Physics & Applications, Siemensstrasse 1, 91301 Forchheim, Germany

Abstract. The accurate segmentation of liver vessels is an important prerequisite for creating oncologic surgery planning tools as well as medical visualization applications. In this paper, a fully automatic approach is presented to quickly enhance and extract the vascular system of the liver from CT datasets. Our framework consists of three basic modules: vessel enhancement on the graphics processing unit (GPU), automatic vessel segmentation in the enhanced images and an option to verify and refine the obtained results. Tests on 20 clinical datasets of varying contrast quality and acquisition phase were carried out to evaluate the robustness of the automatic segmentation. In addition the presented GPU based method was tested against a CPU implementation to demonstrate the performance gain of using modern graphics hardware. Automatic segmentation using graphics hardware allows reliable and fast extraction of the hepatic vascular system and therefore has the potential to save time for oncologic surgery planning.

Keywords: Segmentation, Automation, Computed Tomography, Graphics Hardware, Hepatic Vessels.

1 Introduction

Shape and location of the intrahepatic vessels are of significant importance to liver surgery. Modern minimal invasive operation methods like laser surgery as well as established surgical intervention techniques require the detection of vessels with a diameter down to 2 mm to decide whether an operation can be realised or not.

In order to develop oncologic operation planning tools it is necessary to segment the vessel systems of the liver in a pre-computing step. Therefore, contrast agents are injected in the bloodstream to raise the opacity of those structures and make them appear bright in the CT scan. However, the distribution of the agent and hence the quality of the contrast between the vessels and the liver-tissue depends on the point of time the scan is started. This leads to heavily varying results regarding the amount of contrast in the image. Usually filter based methods are used to enhance the quality of CT images since they can be applied as soon as the image comes out of the machine and require no user interaction. In [1] gaussian/median filters are used to intensify the liver vessels. However, those filters are insufficient to enhance low contrasted CT images since image details and therefore small vessels may get lost. A common approach is to model the vessels locally as tube like objects and applying hessian based eigenanalysis to find those structures in the images [2,3,4,5,6]. Such methods have proven to yield good results on the extraction of vessel systems like pulmonary vessels as well as coronary and retinal arteries. However, often several parameters need to be set that do not have a direct geometrical meaning and have to be statistically determined by testing a large amount of representative datasets. Our filter is based on a reduction of the parameter space by creating a vesselness function that is directly calculated from a pre-defined vessel model, thus supporting a fully automatic solution.

There are few fully automatic approaches for vessel segmentation from hessian based filter outputs. Often the result is thresholded based on either semi-automatically or statistically found values. Selle et al. [1] use an automatic and iterative algorithm after applying their gaussian and laplace like filters. However, hessian based filter outputs fundamentally differ from the results of pure denoising algorithms. Our method is therefore based on two principles: We first calculate a threshold based on a pre-defined ideal vessel model. Then an iterative region growing is used directly on the filter output to ensure that only tube like structures are segmented.

As described earlier, clinical datasets often considerably vary in their contrast quality which can make it impossible to segment the whole vessel tree without any artefacts attached. Therefore we implemented a real time preview function that allows the user to visually verify and refine the segmentation before rendering the result.

Recent advances in the development of graphics processors (GPUs) have increased their programmability and performance, making them interesting for non-graphical applications especially in the field of medical imaging. Owens et al. [7] give an exhaustive overview of current "General Purpose GPU" research and applications. Especially filtering processes are highly suited for a GPU implementation since GPUs are by their architecture massively parallel processors, as shown for example in Langs et al. [8]. In order to evaluate the benefit of using graphics hardware we present a CPU-GPU performance comparison of components of our filter implementation.

In what follows, we outline our approach towards segmenting liver vessels from CT datasets. Section 2 shows the methods used to enhance and automatically extract the vessels as well as the previewing option to verify and manually select a segmentation in low contrast CT images. Section 3 shows the results regarding performance on CPU and GPU as well as robustness of the extraction process on clinical datasets. In 4 we discuss the obtained results and give an outlook of future work on the proposed approach.

2 Methods

2.1 Tube Detection

First, the vessels in the CT images have to be enhanced in order to handle low contrast datasets. For this purpose, a hessian based filter is developed since this kind of filters have proven to yield good results in terms of 3D vessel detection. Following [2] we compose the filter $h(\boldsymbol{r})$ as a linear combination of second order derivatives of a gaussian $g(\boldsymbol{r})(\boldsymbol{r} = (x, y, z))$:

$$h(\boldsymbol{r}) = \alpha_1 g_{xx}(\boldsymbol{r}) + \alpha_2 g_{xy}(\boldsymbol{r}) + \alpha_3 g_{xz}(\boldsymbol{r}) + \alpha_4 g_{yy}(\boldsymbol{r}) + \alpha_5 g_{yz}(\boldsymbol{r}) + \alpha_6 g_{zz}(\boldsymbol{r}), \quad (1)$$

i.e. the filter consists of 6 basis filters given by $h_1 = g_{xx}, .., h_6 = g_{zz}$.

The goal is now to tune the filter to a certain vessel model in order to design a vesselness function that can be applied without setting any parameters in advance. Therefore we model the vessel $v(\boldsymbol{r})$ locally by a cylinder along the x-axis with a radial Gaussian intensity profile:

$$v(\boldsymbol{r}) = b + (a - b)e^{-\frac{y^2+z^2}{2\sigma_v^2}}, \quad (2)$$

with a and b denoting the intensity at the vessel center and boundary respectively. The standard deviation σ_v is chosen to be the same as the standard deviation of the gaussian basis filters $h_1, .., h_6$. The coefficients α_n can now be determined analytically solving the optimization problem given by the maximization of the convolution of filter and vessel signal:

$$S = v * h = \int_{\Re^3} v(\boldsymbol{r})h(\boldsymbol{r})d\boldsymbol{r}, \quad (3)$$

The solution of S using the mathematical framework of Lagrange multipliers yields:

$$h(\boldsymbol{r}) = c \cdot \left(\frac{2}{3}g_{xx}(\boldsymbol{r}) - g_{yy}(\boldsymbol{r}) - g_{zz}(\boldsymbol{r})\right), \quad (4)$$

where $c = \sqrt{\frac{3}{5\pi^{\frac{3}{2}}}} \cdot \sqrt{\sigma_v}$.

To attain a maximum response, the filter has to be oriented along the vessel. This direction can be obtained by eigenvalue analysis of the Hessian H since the eigenvector corresponding to the largest eigenvalue $\lambda_1 = \max(\lambda_1, \lambda_2, \lambda_3)$ points in direction of the biggest local change of grey values (i.e. from vessel center to the vessel boundary). Applied to (4) the optimal filter output can be computed as:

$$S_{opt} = v * h_{rot} = \frac{2}{3}\lambda_1 - \lambda_2 - \lambda_3. \quad (5)$$

In order to avoid the detection of very bright, plate-like structures, i.e., regions where the cross-sectional intensity decreases rapidly in one direction but remains almost constant in the orthogonal one, we multiply (5) by an isotropy factor

$$\kappa = 1 - \frac{||\lambda_2| - |\lambda_3||}{|\lambda_2| + |\lambda_3|} \qquad \in [0, 1]. \quad (6)$$

The factor κ approaches 1 if $|\lambda_2| \approx |\lambda_3|$, i.e. the intensity decreases uniformly with increasing radial distance from the center, which is typically the case for vessels.

The filter output is maximal if the size of the filter mask matches the vessel thickness (i.e. have same standard deviations). Multiscale results are therefore obtained by selecting the maximum response over the range of all scales. In our tests 4 different ($\sigma_v = 1$–4) scales turned out to be sufficient for covering most liver vessels. Fig. 1(a) and (b) show the result of the 3D filtering procedure.

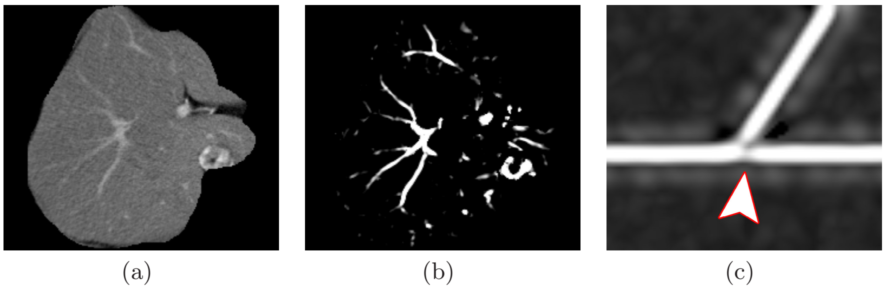


Fig. 1. Contrasted CT dataset of the liver. (a) Original. (b) Filter result. (c) Lower filter response on branches due to deviation from the tube-shaped vessel model (synthetic dataset).

2.2 GPU Implementation

As mentioned in the introduction, current commodity graphics hardware is able to outperform software implementations by several orders of magnitude due to their parallel architecture. In order to investigate how advanced filtering of volume data can be accelerated by hardware we have implemented the filtering steps described above in shader programs. These small programs are executed for every primitive (vertex, geometry, or fragment), depending on the type of program. In our framework, these shaders perform computations on special 2D textures to exploit the two-dimensional memory layout of commodity graphics hardware also for volumetric data. To utilize also the vector-processing capabilities (SIMD) we employ an additional packing scheme that combines four subsequent volume slices in one RGBA texture slice (i.e., tile in our representation).

Although our framework is designed for building complete workflows to process medical volume data while visualizing the (intermediate) results, we have implemented the aforementioned computations in one shader. This fragment program is executed for every voxel and writes the results into an equally sized texture. Depending on the available hardware multiple execution units (pipelines) perform the computations in parallel. However, using the GPU requires to transfer the data to the video memory and, if successive computations are not also performed in hardware, load the results back to the host. In table 1 we have therefore listed the timings for computations and transfer seperately. Although current CPUs provide facilities for parallel computation like SIMD units or multiple cores as well, their design is not as data-processing oriented as graphics hardware. In addition, the price/performance ratio for GPU-based implementations is clearly superior to multi-CPU systems.

The algorithm first fetches the voxels from the texture depending on the current neighborhood size. Then, all partial derivatives are computed according to equation (1) by convolving the voxels with 1D gaussian kernels. In order to compute the filter response, the eigenvalues are determined by using the closed solution for symmetric 3×3 matrices. After the last computations according to the equations above the results are finally written into the target memory.

2.3 Vessel Segmentation

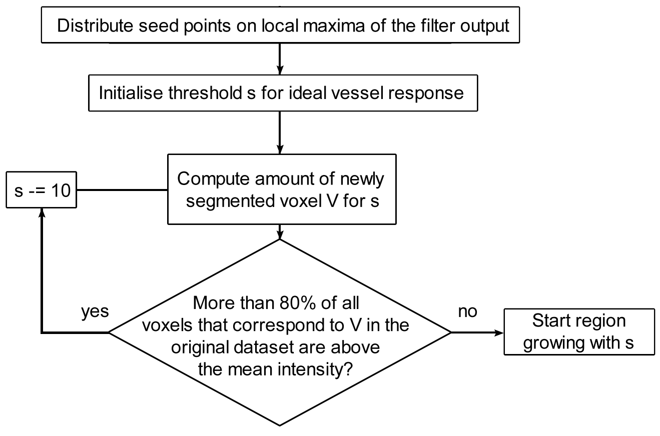
The enhanced vessels are now extracted by applying an automatic region growing algorithm directly on the filter output. The idea behind this is to guarantee that all segmented vessels are connected to each other. Furthermore no post processing is required to remove outlying artefacts that usually appear on tissue borders with high gradients. To operate on the filter output only implicates problems when the vessel splits into two or more branches. In those areas the vessel deviates from the ideal tube-like model and forms a blob-like structure (compare Fig. 1(c)). A possibility to alleviate that problem is to adapt the vesselness function to detect blob-like structures as well as tubes [9]. However, we found that such an approach amplifies the false positive rate of the segmentation procedure in noisy regions with small vessels too heavily. Therefore we rely on the described tube vesselness function alone. As a consequence, the region growing thresholds have to be automatically determined in a way that all vessels and branches are sufficiently segmented while artefacts should be avoided as much as possible.

Fig. 2(a) shows the automatic region growing algorithm used to segment the vessels. First, the volume is divided into layers and seed points are automatically placed on local maxima of the filter output in order to evenly distribute the seeds on the data. Then, the thresholds are initialized with the response value for the ideal vessel model defined in (2). That response can be computed analytically as

$$v * h = (a - b)\kappa\sqrt{\frac{6}{5}}\pi^{\frac{3}{4}}\sigma_h^{\frac{3}{2}}, \tag{7}$$

where σ_h denotes the standard deviation of the filter analysis window.

Since real vessels will deviate from the ideal model to a certain degree, the thresholds are iteratively refined. For this task, the original unfiltered data is comprised, because it is not feasible to decide from the filter output alone whether any detected structure is a real vessel or just appeared due to local noise. The reason for this lies in the aim of the Hessian based filters to detect tubular structures. Therefore the filter output will show many small tubular patterns, that cannot be directly classified as vessels. As a result, the region growing shows no clearly identifiable leaking from the vessels into the background like it is given on an ordinary dataset.


Distribute seed points on local maxima of the filter output
Initialise threshold s for ideal vessel response
Compute amount of newly segmented voxel V for s
s -= 10
yes
More than 80% of all voxels that correspond to V in the original dataset are above the mean intensity?
no
Start region growing with s
(a)

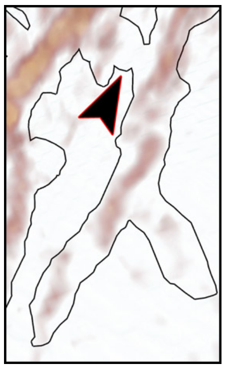


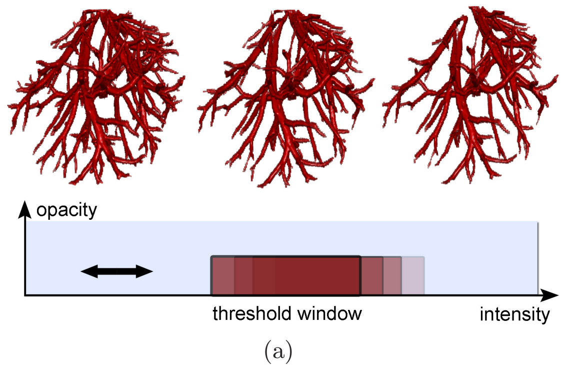
Fig. 2. (a) Automatic region growing algorithm on the filter output. (b) Continous segmentation on a low contrast dataset. A connection between two adjacent vessels needs to be refined by manual interaction (arrow).

The iterative refinement of the region growing thresholds using the original data works as follows: For the given threshold the mean value of the pixels in the original data corresponding to the detected pixels in the filter output is computed. This value is compared to the mean intensity of the dataset. If 80% of the pixels are above that value the threshold is lowered and the procedure is applied anew. Otherwise the iteration stops and the current threshold is taken as the final threshold. This automatic approach allows the continous segmentation of vessels that are disconnected in the original dataset while preventing the segmentation of noisy structures. However in the case of very low contrasted datasets, unwanted connections between vessels may appear Fig. 2(b).

2.4 Real Time Preview of Parameter Variations

As an enhancement of the automatic approach of the last section our framework gives the opportunity of a manual refinement of the segmentation result by implicitly adjusting the underlying parameters. Especially for people who do not have a computer science background, it is favorable to have a simple and intuitive dialog that gives the user the opportunity to manipulate the segmentation result without the need to manually adjust any parameters. By introducing a parameter free vesselness function in section 2.1 the threshold of the region growing is the only variable value in the segmentation process. In order to give the user a direct feedback of adjusting the threshold a real time preview of pre-computed segmentations is provided.

Starting with the automatically determined threshold, the region growing is applied with slightly varying values leading to different segmentation results. The



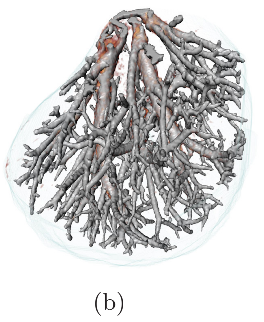


Fig. 3. Real time preview of manual parameter refinement. (a) A dynamic transfer function is used to visually seperate the pre-computed segmentation results. (b) The desired view is selected by the user and then rendered as a 3D mesh.

computed results are all stored in a single tagged volume separated by different values. This procedure can already be partly realized during the refinement process of 2.3 by storing the segmentation results of each iteration. The tagged volume is then visualized by direct volume rendering. In order to display all segmentations separately a dynamic transfer function is applied which only maps values to an opacity > 0 that correspond to the current threshold. Because the manipulation of the transfer function does not require to change or reload the dataset representation in memory it is very fast and therefore a real time preview of the different segmentation results is possible. Fig. 3(a) shows the results of adjusting the thresholds by moving a slider for the 3D case. After selecting the desired view the segmentation can be rendered as a 3D mesh using the marching cubes algorithm (Fig. 3(b)).

3 Results

We applied the segmentation method to 20 clinical CT datasets (14 for portal-venous phase, 6 for arterial phase) with an in-plane-resolution of 512^2 (0.613 to 0.84 mm voxel size) and an axial spacing of 1.0 to 2.5 mm. For the tests the

Table 1. Comparison of CPU and GPU performance. The first two columns show the computation time of the filter response in seconds on an AMD 1.8 GHz CPU using a $7 \times 7 \times 7$ kernel. The last two columns present the results of the GPU implementation performed on an Nvidia Geforce 8800 GTS with 640 MB of VRAM.

Dataset	$278 \times 271 \times 139$	$278 \times 308 \times 99$	$278 \times 271 \times 139$	$278 \times 308 \times 99$
Partial derivative ($\times 6$)	3.27(19.62)	2.72(16.32)	0.23(1.38)	0.19(1.14)
Filter response	11.93	9.87	0.11	0.09
Transfer to/from VRAM	$\times$	$\times$	0.28/0.20	0.21/0.16
Total	31.55	26.19	1.97	1.59

liver was manually segmented in a pre-processing step (resulting in a region of $300 \times 300 \times 120$ on average).

3.1 Comparison CPU – GPU

Table 1 shows a comparison between CPU (VisualStudio 6.0 compiler) and graphics hardware implementation of the filter. It can be seen that the GPU is approximately 15 times faster than the CPU concerning the pure filtering task and 100 times faster in the case of filter response calculation. This performance gain is partly attenuated by the time needed to transfer the data to the video RAM and back to the host. Although this transfer takes about 1/4 of total computation time the GPU implementation is still 15 times ahead.

3.2 Segmentation

A region growing (manual settings of seeds and thresholds) running on the unfiltered datasets was used for a visual comparison with the automatic approach. Hereby, the parameters of the region growing were chosen to prevent a leaking to the liver tissue while segmenting as much vessels as possible at the same time. Fig. 4(a)-(d) shows an exemplary result for the portal venous phase. With the

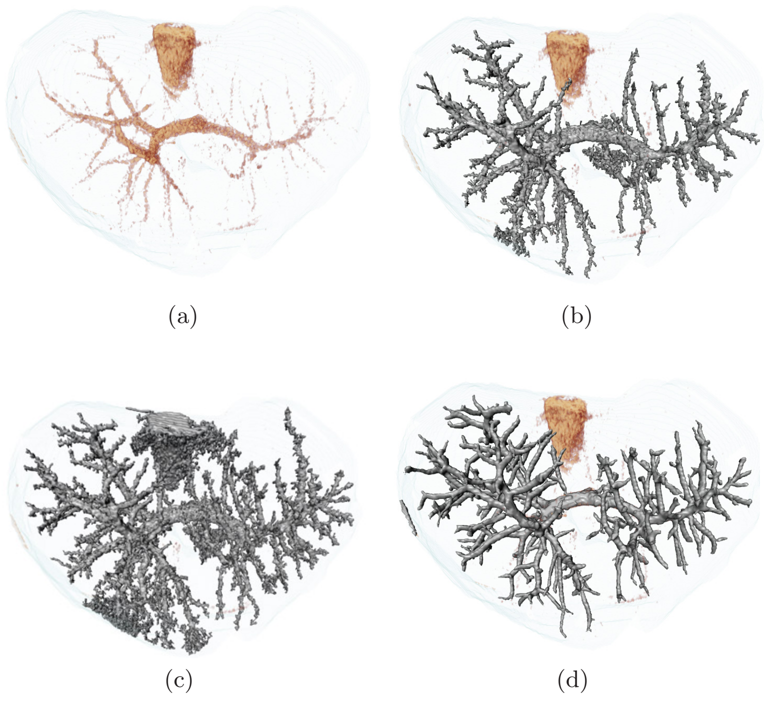


Fig. 4. Comparison of a manual region growing segmentation with the automatic method (portal venous phase). (a) Volume rendering of the original dataset. (b) and (c) Rendered masks of region growing with different thresholds. (d) Our approach.

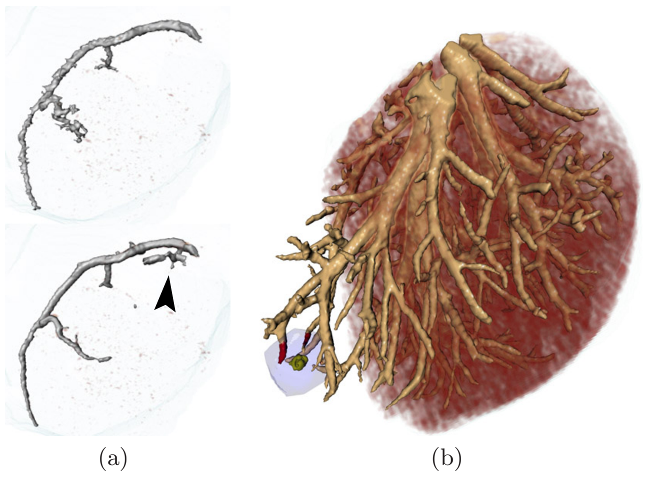


Fig. 5. (a) Arterial phase: manual region growing (top) and automatic segmentation with an artefact attached (bottom, arrow). (b) Illustration of an operation planning prototype using the proposed approach.

proposed approach more and better defined vessels can be segmented compared to the manual region growing. In fig. 5(a) an artefact appeared, because the automatic threshold was slightly too low. In such a case the preview function described in section 2.4 has to be used to refine the result. In order to detect falsely segmented structures, our operation planning prototype is providing a visual comparison with Maximum Intensity Projection, transparence and transfer function refinement. Fig. 5(b) illustrates a tumor resection scenario build with the proposed methods.

4 Discussion

We proposed an automatic hepatic vessel enhancement and segmentation approach together with a user friendly real time preview function to manually refine the resulting masks. A comparison with a manual region growing showed the potential of the method to save surgery planning time while providing accurate segmentations at the same time. An implementation on the GPU showed that a hardware implementation is able to perform the filter operations approximately 15 times faster, even for larger neighborhoods. The overall performance could be increased by the same factor.

Future work includes the isolation of portal and hepatic veins. For that task, knowledge about the tree structure of the vascular systems has to be incorporated. In addition, this procedure is promising to further reduce artefacts on low contrasted images. Also, we will further investigate on extending the use of graphics hardware to the other steps of the procedure.

References

1. Selle, D., Preim, B., Schenk, A., Peitgen, H.: Analysis of vasculature for liver surgical planning. IEEE Transactions on Medical Imaging 21, 1344–1357 (2002)
2. Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A.: Multiscale vessel enhancement filtering. In: Wells, W.M., Colchester, A.C.F., Delp, S.L. (eds.) MICCAI 1998. LNCS, vol. 1496. Springer, Heidelberg (1998)
3. Sato, Y., Nakajima, S., Atsumi, H., Koller, T., Gerig, G., Yoshida, S., Kikinis, R.: 3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. In: Troccaz, J., Mösges, R., Grimson, W.E.L. (eds.) CVRMed-MRCAS 1997. LNCS, vol. 1205, pp. 213–222. Springer, Heidelberg (1997)
4. Manniesing, R., Viergever, M.A., Niessen, W.J.: Vessel enhancing diffusion: A scale space representation of vessel structures. Medical Image Analysis 10, 815–825 (2006)
5. Koehler, H., Couprie, M., Bouattour, S., Paulus, D.: Extraction and analysis of coronary tree from single x-ray angiographies. In: Galloway Jr., R.L. (ed.) Medical Imaging 2004: Visualization, Image-Guided Procedures, and Display. Proceedings of the SPIE, May 2004, vol. 5367, pp. 810–819 (2004)
6. Langs, G., Radeva, P., Rotger, D.: Explorative building of 3d vessel tree models. In: Digital Imaging in Media and Education. 28th annual workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR) (2004)
7. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krueger, J., Lefohn, A.E., Purcell, T.J.: A Survey of General-Purpose Computation on Graphics Hardware. Computer Graphics Forum 26(1), 80–113 (2007)
8. Langs, A., Biedermann, M.: Filtering Video Volumes Using the Graphics Hardware. In: Ersbøll, B.K., Pedersen, K.S. (eds.) SCIA 2007. LNCS, vol. 4522, pp. 878–887. Springer, Heidelberg (2007)
9. Zhou, C., Chan, H.-P., Hadjiiski, L.M., Patel, S., Cascade, P.N., Sahiner, B., Wei, J., Ge, J., Kazerooni, E.A.: Automatic pulmonary vessel segmentation in 3D computed tomographic pulmonary angiographic (CTPA) images. In: Reinhardt, J.M., Pluim, J.P.W. (eds.) Medical Imaging 2006: Image Processing., March 2006, vol. 6144, pp. 1524–1530 (2006)