Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 320

Fast Retinal Vessel Analysis

Michael Krause, Ralph Maria Alles,
Bernhard Burgeth and Joachim Weickert

Saarbrücken 2012

Fachrichtung 6.1 – Mathematik Preprint No. 320
Universität des Saarlandes submitted: Dec 18, 2012, update: Feb 28, 2013

Fast Retinal Vessel Analysis

Michael Krause
Institute of Microelectronics
Building A5.1
Saarland University
66041 Saarbrücken
Germany
michael.krause@lme.uni-saarland.de

Ralph Maria Alles
Augenärzte Bies-Alles-Mély-Hadavi
66740 Saarlouis
Germany
email2005@dr-alles.de

Bernhard Burgeth
Dept. of Mathematics
Faculty of Mathematics and Computer Science
Building E2.4
Saarland University
66041 Saarbrücken
Germany
burgeth@math.uni-sb.de

Joachim Weickert
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science
Building E1.7
Saarland University
66041 Saarbrücken
Germany
weickert@mia.uni-saarland.de

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax:	+ 49 681 302 4443
e-Mail:	preprint@math.uni-sb.de
WWW:	http://www.math.uni-sb.de/

Abstract

We introduce a fast image processing system that allows to analyse digital databases of retinal images in a short time, and to process the image in situ while the patient is examined. While it achieves a comparable quality as state-of-the-art methods, it differs from most of them by the fact that it is extremely fast.

Retinal blood vessels are enhanced via convolution with the second derivative of the local Radon kernel. It is rotated by different angles, and it adapts itself via a maximisation procedure to the vessel directions. We combine smoothing along vessel directions with contrast enhancement across them. We detect vessels as connected structures with very few interruptions. A subsequent skeletonisation allows a higher-level description of the vessel tree.

To end up with a very fast system, we combine efficient algorithms for numerical integration, differentiation and interpolation, and we propose an automatic parameter selection strategy. Our convolution kernels are precomputed and stored into cached constant memory. All essential subroutines are intrinsically parallel, and the resulting system is implemented on GPUs using CUDA.

Our qualitative evaluations with the DRIVE database and our own database shows that the system achieves competitive performance. It is possible to process images of size 4288×2848 pixels in 1.2 seconds on an NVIDIA Geforce GTX680. Compared to our sequential implemention, this amounts to a speed-up by two orders of magnitude.

1 Introduction

The inspection of retinal vessels is a well established and scientifically evaluated method for the screening of important vascular diseases. Widely available ”Non-Mydriatic” cameras allow ophthalmologists to create considerable databases that prepare the path for unprecedented numerical and statistical analysis. Thus, it would be attractive to have an efficient image processing system for the analysis of huge databases of retinal images. In this way, it also becomes possible to test the quality of a retinal image in interactive time while the patient sits before the camera, and to take another image, if necessary. Unfortunately, most systems that have been proposed in the research literature and give results of high quality are too slow for these tasks.
The goal of the present paper is to address these problems by introducing a system for retinal image analysis that combines high quality with high computational efficiency. It combines a number of very useful concepts:

- A local Radon transform [7] is used for smoothing along vessel directions.

- Perpendicular to these directions, contrast enhancement is achieved by a second-order differential operator.
- The resulting convolution kernel is precomputed and stored to constant memory.
- We propose a novel strategy for the automatic adaptation of a threshold parameter that has a large influence on the quality of the output. This allows to process an entire database without manual interactions.
- All components of our system have been selected on the basis of their intrinsic parallelism.
- This parallelism is exploited in a highly efficient CUDA-based implementation on general purpose graphics processing units (GPUs).

We end up with a system that achieves a qualitative performance with state-of-the-art methods, but outperforms most of them with respect to its computational efficiency.

Organisation of the paper. The structure of our paper is as follows: As the understanding of the underlying model is necessary for understanding its efficient parallel implementation, the modelling aspects are discussed first. To do this, in Section 2 the preprocessing as well as the use of second order derivative filters for vessel detection are explained. In Section 3 the use of our main tool, the second derivative of the local Radon transform, is explained. We further apply a skeletonisation algorithm and detect the branching points for better visualisation of our results in Section 4. In Section 5, fast parallel implementation aspects are addressed. We report on experiments with a public database and our own clinical database in Section 6. The paper is concluded with a summary in Section 7.

Related work. Retinal vessel segmentation algorithms can be grouped into various categories. Some algorithms – including ours – rely on matched filter response techniques, see [1, 2, 5, 9, 13, 22]. A shape model that resembles the vessel cross-section is convolved with the image and rotated by several angles in the search for an optimal fit. The filter exists in one or two-dimensional versions [13].
The Laplacian of a Gaussian (LoG) is used by Vermeer et al. [25] to extract vessels. The LoG is a second order derivative filter, such as the second derivative of the local Radon transform used by us. Vermeer et al. derive optimal values for the standard deviation of the Gaussian and for the threshold value in order to extract bar-shaped vessels. Separable kernels composed from a second order Gaussian derivative and a window function are used by Gang et al. [5] and Sofka and Stewart [22]. Gang et al. employ a box function, while Sofka and Stewart utilise a

Gaussian window function. The algorithms proposed in [14] use morphological operators for preprocessing, as we also do.
In [9] a piecewise threshold is used to separate vessel pixels from non-vessel pixels. An alternative way to distinguish between vessels and non-vessels is supervised classification based on a training set: to each pixel a feature vector is assigned containing different properties, for example greyscale, matched filter response to a wavelet filter [21], line operators [18], directional derivatives and more [24]. Vessel segementation with suitable filters has been done already 1998 by Frangi et al. [4].
A recent article dealing with the implementation of a matched filter technique for general medical imaging on GPUs and FPGAs has been written by Savrarimuthu et al. [20]. It analyses vessels in the human forearm, and it concentrates on achieving video resolution (640×480) in real-time.

2 Vessel segmentation using second order derivatives

2.1 The cross section of a vessel

Vermeer et al. [25] consider a vessel as a ”brightness gap” with step size h and width w. The grey-value of the vessel is by the value h lower than the surrounding retina, and its width is w. In order to segment vessels from background, Vermeer et al. convolve the cross section of the vessel with the second derivative of a Gaussian with standard deviation σ. As the vessel profile is modelled as a brightness gap and the vessel is darker than the background, the second derivative of the vessel profile contains two positive peaks showing to the interior of the vessel and two negative peaks outside the vessel. Due to the Gaussian convolution, these positive peaks are propagated to the interior of the vessel, thus allowing to detect the interior of the vessel via thresholding. Thus, Vermeer et al. determine the interior of the vessel via a carefully chosen threshold depending on the relation of the Gaussian standard deviation σ and the vessel width w.
Gao et al. [6] and Chapman et al. [2] use a Gaussian shaped model of a vessel. The vessel profile of Gao et al. assumes the vessel profile to be convex in the interior of the vessel, this way the second derivative in direction of the vessel profile to be greater than a threshold $T > 0$.

2.2 Vessel detection using separable convolution kernels

The method proposed in this article belongs to the class of matched filter approaches. As we have seen so far, vessels are valley-like structures, see Vermeer et al. [25]. Such structures can be detected using convolution kernels that are composed by a low-pass filter in direction of the vessel run and a second-order derivative in direction of the cross-section. The realisation of such kernels is performed by the tensor product of the components. In fact, the convolution kernel used in this article is composed by a second order central difference quotient tensored with a Gaussian.

Kernels that can detect lines in many directions and use a large neighbourhood are proposed by Gang et al. [5], and by Sofka and Stewart [22]. Gang et al. calculate a Gaussian derivative perpendicular to the vessel direction multiplied with a box function in tangential direction. Instead, Sofka and Stewart use a Gaussian convolution in tangential direction as well.

In contrast to these approaches, our matched filter computes finite difference approximations of the second order derivative combined with true Gaussian weighted line integrals that are obtained by interpolation; see next sections. This enhances the localisation of our kernel and lowers the computational load in comparison with Sofka and Stewart [22] and Gang et al. [5].

3 Vessel detection using the local Radon transform

3.1 Techniques for Preprocessing

In our early experiments it became apparent that the brightness of the optic disc was responsible for erroneous results at its boundary. These errors can be diminished with a classical morphological operator, namely the black top hat, followed by an image inversion. Detailed information about morphological operators are provided, for example, in [23]. Note that the black top hat only effects the boundary of the optic disc which would lead to a large number of false positives. Another preprocessing step is a convolution of the input image f with a Gaussian K_σ,

$$f_\sigma = K_\sigma * f \ . \tag{1}$$

This convolution is the main regularisation of our algorithm. Alternatively we have also tried anisotropic smoothing techniques, but they were more noise sensitive than isotropic Gaussian smoothing and did not lead to a substantial improvement of the results.

3.2 Coordinates

One of the main aspects of our vessel detection algorithm is the adaptivity to the actual vessel profile. This is achieved by convolving the image with a strongly anisotropic kernel, whose orientation is determined by a rotation angle θ.

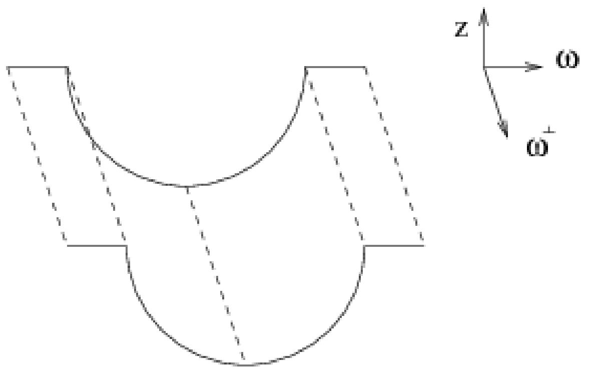


Figure 1: Vessel appearance and local coordinates, ω and $\omega^{\perp}$ span the image plane, z-coordinate shows the grey value

We introduce local coordinates as shown in Fig. 1 where we depict a three-dimensional vessel model in its representation as a grey-scale image. The image plane is spanned by two axes, one in direction ω and one in direction $\omega^{\perp}$ perpendicular to it,

$$\omega = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \; \omega^{\perp} = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}. \tag{2}$$

The direction $\omega^{\perp}$ corresponds to the vessel run, while the grey-value representation of the vessel profile is represented in the ω-z plane. Later on, we will differentiate in direction ω and integrate in direction $\omega^{\perp}$. The grey-scale intensity f at a point $p = s\omega + t\omega^{\perp}$ in the image plane is represented by a z-value, hence the darker inner part of the vessel attains a smaller z-value than the surrounding retina.
We have seen that the interior of the vessel can be detected via the convolution of f_σ with a kernel elongated in the direction $\omega^{\perp}$ of the vessel. In order to treat each point $x \in \mathbb{R}^2$ as the centre of the ω-$\omega^{\perp}$ coordinate system, we slightly modify the coordinate system by introducing the centre point x. Thus, each point p can be written as $p = x + s\omega + t\omega^{\perp}$ for a fixed x.

3.3 The second derivative of the local Radon transform

The coordinate system from subsection 3.2 is used to define the local Radon transform of f_σ [7]:

$$R_\rho f_\sigma(x, \omega, s) = \int_{-\infty}^{\infty} K_\rho(t) f_\sigma(x + s\omega + t\omega^{\perp})\, dt, \tag{3}$$

R_ρ produces a regularised version of f_σ. The smoothing is performed in the direction of the vessel run, which is an advantage of using the local Radon transform.

Remarkably, the convexity of the cross sections carries over to the second derivative of the local Radon transform:

$$\begin{aligned}(R_\rho f_\sigma)''(x,\omega,s) &:= \int_{-\infty}^{+\infty} K_\rho(t)\frac{\partial^2}{\partial s^2} f_\sigma(x+s\omega+t\omega^\perp)dt \\ &\geq T\int_{-\infty}^{+\infty} K_\rho(t)\,dt = T > 0\,, \qquad (4)\end{aligned}$$

where we have used the assumption $\frac{\partial^2}{\partial s^2} f_\sigma(x+s\omega+t\omega^\perp) \geq T > 0$ in the inner part of the vessel and the normalisation $\int_{-\infty}^{+\infty} K_\rho(t)\,dt = 1$. The second-order differentiation perpendicular to the vessel orientation is an additional contrast-enhancement between vessels and surrounding retina. In [12] the local Radon transform has been used to detect edges in digital images.

3.4 Criterion for vessel segmentation

In order to use the local Radon transform and its derivative to decide whether a point x is a vessel point or not, we consider $(R_\rho f)''(x,\omega,0)$ as a function of ω alone for a fixed point x. If a direction ω_x exists for which

$$(R_\rho f_\sigma)''(x,\omega_x,0) > T > 0 \qquad (5)$$

holds, we recognise x as a vessel point. It is important to remark that for the classification of the vessel points the accurate estimation of the vessel direction is not crucial due to the employed threshold technique (5). We compute the second derivative via a central difference with step size $h = 1$, as we assume the image to be sampled with that grid size. The resulting approximation $H(x,\omega)$ to $(R_\rho f_\sigma)''(x,\omega,0)$ is given by

$$H(x,\omega) = R_\rho f_\sigma(x,\omega,-1) - 2R_\rho f_\sigma(x,\omega,0) + R_\rho f_\sigma(x,\omega,1)\,. \qquad (6)$$

In our basic assumptions vessels are always convex. Hence, points with $H(x,\omega) < 0$ for all ω are of no interest. Therefore, we can restrict our attention to the function

$$\operatorname{res}(x) := \max(0, \max_\omega H(x,\omega))\,. \qquad (7)$$

Vessel points x are characterised by $\operatorname{res}(x) > T$.

3.5 Efficient implementation

We can implement the calculation of H as correlations with pre-computed kernels g_ω, in the ω-$\omega^\perp$-coordinate system. Exploiting the symmetry of both the Radon kernel and the Gaussian kernel, the correlations are in fact convolutions. To see this we assume x to be 0, and we approximate $R_\rho f_\sigma(0,\omega,l)$ with a trapezoid rule,

$$R_\rho f_\sigma(0,\omega,l) \approx \sum_{k\in\mathbb{Z}} K_\rho(k) f_\sigma(l\omega + k\omega^\perp) \ . \tag{8}$$

The points where f_σ has to be evaluated are approximated by bilinear interpolation which leads to a formula of the form

$$R_\rho f_\sigma(0,\omega,l) \approx \sum_{m\in\mathbb{Z}^2} \alpha_{\omega,l}(m) f_\sigma(m) \tag{9}$$

with suitable weights $\alpha_{\omega,l}(m)$. From the central difference quotient in (6) we obtain

$$H(0,\omega) \approx \sum_{m\in\mathbb{Z}^2} g_\omega(m) f_\sigma(m) \tag{10}$$

for some g_ω. The approximation of $H(x,\omega)$ at an arbitrary point x can be obtained by a correlation with the symmetric kernels g_ω:

$$H(x,\omega) \approx \sum_{m\in\mathbb{Z}^2} g_\omega(m) f_\sigma(x+m) \ . \tag{11}$$

Since g_ω have a small support, the correlations with the pre-computed kernels g_ω are highly efficient. Our filters g_ω consist of about 300 nonzero points per direction requiring $600+600+1200 = 2400$ bytes, each for x-coordinate, y-coordinate and float value. These 2400 bytes fit to constant memory cache and can be accessed in a very fast manner.
The filters act like the second derivative operator $(1,-2,1)$ in ω-direction and a Gaussian in $\omega^\perp$-direction. As the different g_ω are obtained from the local Radon transform via discretising the involved integral and differentiation operators, they can be seen as a discretised version of the local Radon transform. Note that the g_ω filters are separable in the ω–$\omega^\perp$ coordinate system, but not in the standard coordinate system. The kernels g_ω represent the precomputation steps that involve interpolation, integration with the trapezoidal rule and differentiation. Precomputing these kernels increases the efficiency.

3.6 Eliminating small connected components

To address the problem that our algorithm might produce false positives in addition to real vessels, we have to ensure that only relevant components are recognised as vessels. Small connected components are dismissed as non-vessels. The components and their size are computed via a Union-Find algorithm [3].

The central light reflex in the middle of larger vessels establishes a relatively small non-vessel component which is completely surrounded by an already detected vessel. As before these small components are identified with the help of a Union-Find algorithm and then turned into vessel points.

3.7 Automatic selection of the threshold

In our system the threshold parameter T has a large impact on the quality of the segmentation. Unfortunately, it does not give good results to use an identical threshold for all images. On the other hand, a manual optimisation of T to each individual image is prohibitive for a large database. Therefore, we have designed an automatic procedure that adapts T to each image without user interaction.

We have performed a lot of investigations in this direction to come up with a method that behaves in a stable way and gives good results. In a nutshell the main steps are as follows.

1. Compute the histogram of the local Radon result.
2. Smooth the histogram by convolving with a Gaussian.
3. Compute the third derivative of the smoothed histogram.
4. Choose the highest point T at which the graph of the this derivative leaves a small corridor of diameter T_h around the x-axis.

In this way, a relatively sensitive threshold T has been replaced by an insensitive threshold T_h. Note that T_h has to be set only once per image *type* and not again for every *single* image. Although at first glance it may seem to be unstable to use the third derivative of the histogram for threshold selection, practical experiments show that it is in fact stable. The used technique is shown in Fig. 2, where the threshold is taken at the point where the graph of the third derivative of the histogram of the second derivative of the local Radon transform first leaves a small corridor from -35 to 35 around the x-axis. For better visualisation, the graph starts with $x = 10$.

4 Skeletonisation and detection of branching points

In order to obtain a more abstract representation of the vessels we construct a graph-like structure, the so called 1-pixel wide vascular tree [1]. This representation facilitates the analysis of the vessel. To this end we apply a skeletonisation algorithm as described, for example, in [8]. The employed procedure is a prairie-fire algorithm, avoiding the well-known failures of other thinning-line algorithms

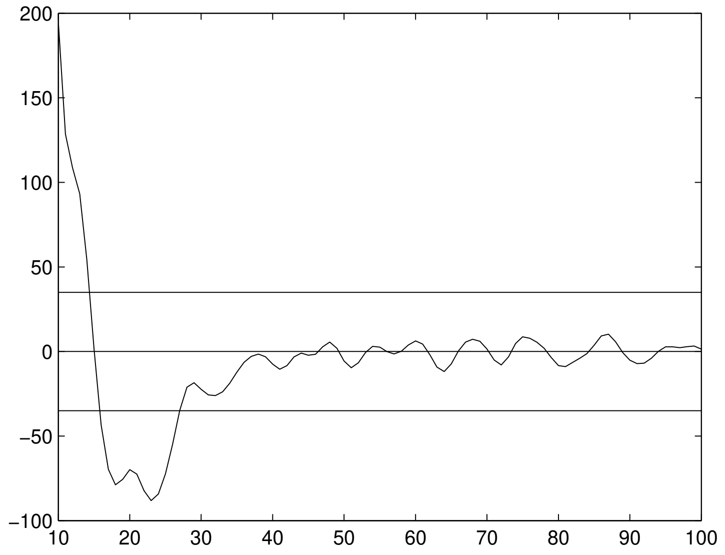


Figure 2: Using the third derivative of the histogram for threshold probing. The horizontal lines show the interval $[-35, 35]$ around the x-axis.

[26]. As a result of the algorithm connectivity patterns of the vessel structure are represented correctly by the obtained skeleton. After the skeleton is computed, the branching points are detected by regarding 0-1 transitions of the points surrounding the candidates for branching points. Note that endpoints p of the skeleton are characterised by having only one neighbour in the skeleton. The skeletonisation can be used for further analysis of the vascular tree, such as arteria-vein decomposition.

5 Fast implementation using NVIDIA CUDA

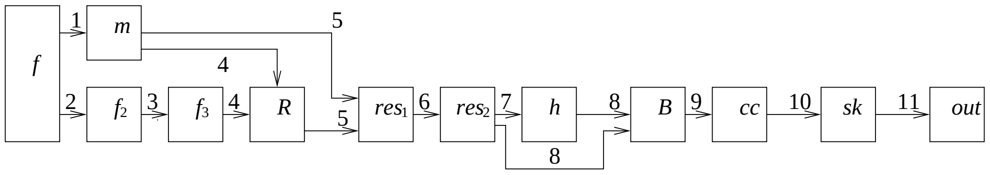


Figure 3: Flow diagram of our algorithm

Now that we have discussed all components of our system, let us focus on its implementation aspects.

Table 1: Runtime on a NVIDIA Geforce GTX680 for an image of size 4288×2848.

Step	Function	Runtime [s]	result
1	create mask	0.0173	m
2	inverted black-top-hat	0.0331	f_2
3	Gaussian convolution	0.0110	f_3
4	correlation with g_ω	0.1530	R
5	apply mask	0.0011	res_1
6	linear rescale	0.0014	res_2
7	create histogram	0.0020	h
8	create binary image	0.0033	B
9	analyse connected components	0.0681	cc
10	create skeletonised image	0.0506	sk
11	mark crossing points	0.0047	out
	sum	0.3748	

Apart from reading from and writing to hard disk, all parts of our system are implemented in the CUDA environment. CUDA is a programming language extension for languages such as C used for general-purpose programming on NVIDIA graphics cards; see [17]. Basic introduction to programming in CUDA is given by [19, 11]. For a more detailed introduction to CUDA see [17]. Note that the pre-computation of the kernels g_ω (11) does not need to be implemented parallel as the pre-computation is fast enough.
The individual steps of our implementation are listed in Table 1. Note that only one copy of the input image f from host (CPU) to device (GPU) has to be performed, and that only those intermediate images have to be copied from device to host which must be stored to hard drive. Figure 3 shows the flow of our algorithm. The names in the squares indicate the computed intermediate result, while the numbers above the arrows indicate the current step in the algorithm; see Table 1. Let us now have a more detailed look at the 11 steps within our system.

1. *Create Mask:* The first part of the implementation is the creation of a mask image m describing the Region of Interest. It is created from the input image f by first determining the black parts of f via thresholding and can be implemented fully in parallel. An erosion with a separable square structuring element [23] is applied to reduce boundary effects which had been created while computing the second derivative of the local Radon transform.

2. *Inverted black-top-hat*: An inverted black top-hat [23] is applied in order to reduce the effects created by the boundary of the optic disc. Due to the

local behaviour of erosion and dilation the latter algorithms can be implemented in parallel. The structuring element was a separable square, thus being implemented in two kernels, one for the x-direction and one for the y-direction. The resulting image is called $f_2 =: \tilde{f}$, where $\tilde{f}$ is the applied erosion to the image f.

3. *Gaussian convolution:* The main regularization is achieved by a Gaussian convolution to compute $f_3 := \tilde{f}_\sigma$. Its parallel implementation uses texture memory for the input image $f_2 = \tilde{f}$ in order to enable local caching [19]. The Gaussian kernel is accessed using constant memory, thus allowing caching.

4. *Correlation with g_ω:* The computation of the second derivative of the local Radon transform R_ρ is approximated by a convolution with the kernels g_ω. As these kernels are sparse, their coordinates in $\mathbb{Z}^2$ as well as their values fit to constant memory. This allows caching and fast access. The input image $f_3 = \tilde{f}_\sigma$ is accessed using read-only cached texture memory. The output is denoted by R. Note that it is only necessary to compute the correlation (which is in fact a convolution) in the pre-computed region of interest; see the arrow in Figure 3 from m to R.

5. *Apply mask:* The application of the mask is just a multiplication with 1 or 0. It can be implemented fully in parallel. The result is called res_1 and it holds $res_1 \geq 0$, see (7).

6. *Linear rescale:* The linear rescaling to $[0, 255]$ is a parallel reduction [19] to compute the maximum value of the second derivative of the local Radon transform image followed by a point-wise multiplication. The resulting image is called res_2.

7. *Create histogram:* The histogram h was created as described in [19]. The image is decomposed into blocks, each block storing the local histogram in shared memory of size 256. The local histograms are computed using shared memory atomics, thus requiring Compute Capability 1.3 of the GPU for running our program. Then the local histograms are combined to a global histogram by using global atomics, which are already available for Compute Capability 1.1.

8. *Create binary image:* The creation of the binary image B from res_2 is just a point-wise operation, assigning each point the values 0 or 255, respectively. As indicated in Figure 3, the threshold depends on the histogram h of res_2.

9. *Analyse connected components (ccl):* The connected component analysis resulting in the image *cc* was performed via a Union-Find algorithm, see [10, chapter 35] for details. The Union-Find algorithm used in [10] works as follows. Each point is assigned its father in the Union-Find forest as an integer containing the father's number. A point which is assigned its own number is the representative of its set. The Union function merges two sets together, and the Find function finds the set a point is assigned to. The crucial step in the Union function is an atomicMin called when two sets are merged together and the new label of the set is smaller than the old label, see [10, Figure 35.8]. The algorithm starts in *fast* shared memory working on a single tile of the input image. The next steps are merging the points at the boundary of tiles with border length increasing by the factor 2 in each step, until the whole image is processed. The last step is an optional flattening of the Union-Find forest, reducing future Find queries. For a very detailed description, see [10].

10. *Create skeletonised image:* the skeleton *sk* is computed in the way described in [8]. The algorithm consists of a prairie-fire-like algorithm. As we assume that the diameter of all vessels is smaller than 40, the skeletonization algorithm is implemented by a fixed number of 20 steps. In each step, the boundary pixels, which are not needed to be part of the skeleton are marked. After all boundary pixels have been marked, they are removed. As soon as the 20 iterations have been performed, only the real skeleton points remain, see [8] for details. In the sequential setting, a loop would pass over all pixels in the image, marking most of the boundary for deletion and delete these pixels after the marking step is completed. A corresponding parallelization is essentially a parallel unrolling of the loop.

11. *Mark crossing points:* The crossing points are easily determined by analysing the 8-point vicinity of each point of the skeleton, leading to the final image *out*. More detailed, order the 3×3-vicinity of a point P_1 as a sequence $P_2, P_3, \ldots, P_9, P_2$, such as in Fig. 4. If the number of 0-1 transitions in the above sequence is greater than or equal 3, we have a crossing point. Note that endpoints have only one 0-1 transition.

P_2	P_3	P_4
P_9	P_1	P_5
P_8	P_7	P_6

Figure 4: 8-point vicinity of a point P_1.

6 Experiments

6.1 Fast parallel implementation using NVIDIA CUDA

As already described earlier, a black top hat followed by a Gaussian convolution are performed as preprocessing steps. The chosen number of directions ω was $p = 6$, so 6 different convolutions have been performed in the convolution steps. The chosen directions are distributed equidistantly in the interval $[0, 5\pi/6]$. Data that are not directly accessible have been obtained by linear interpolation. Using 6 directions is a good compromise between angular resolution and efficiency. Table 2 shows the parameters of our settings.

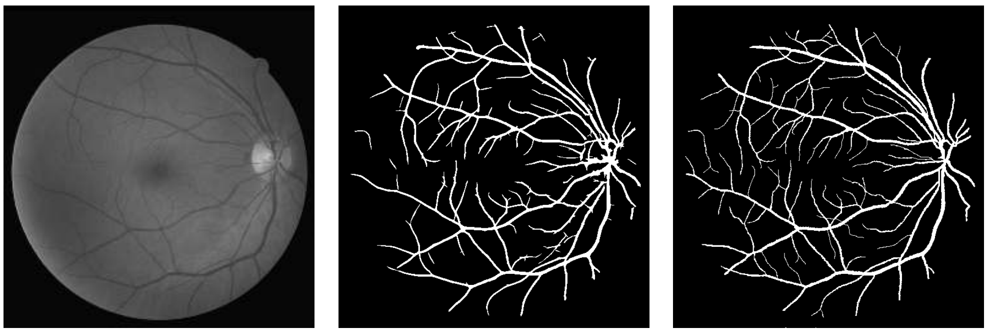
Table 2: Parameter Settings

Parameter	DRIVE	2048×1536	4288×2848
σ	1.75	2	4
ρ	4.5	6	12
size of conn. comp.	25	100	500

The total runtime of our system on an NVIDIA Geforce GTX680 is less than 0.78 seconds for an image of mid resolution (2048×1536 pixels, $\sigma = 2$, and $\rho = 6$), including reading and writing to hard drive, For an image of high resolution (4288×2848 pixels, $\sigma = 4$, and $\rho = 12$), the runtime is less than 1.2 seconds. On an Intel Core i5 CPU with 2.67GHz, the corresponding runtime would amount to 26 seconds for the mid-resolution image and 127 seconds for the high-resolution image. Note that the main part of the runtime for the parallel implementation is dedicated to reading and writing to hard disc. On our GPU-based system. it is possible to evaluate 6500 images of size 4288×2848 or more than 20000 images of size 2048×1536 in about 3 hours.

Table 3: Performance of vessel segmentation methods (DRIVE database) The results for other techniques are taken from [14].

Method	Accuracy	True pos.	False pos.
Human observer [15]	**0.9473**	**0.7761**	0.0275
Our method	0.9468	0.7517	0.0259
Mendonça [14] (grey)	0.9463	0.7315	**0.0219**
Mendonça [14] (green)	0.9452	0.7344	0.0236
Niemeijer, [15], Staal [24]	0.9442	0.7194	0.0227
Niemeiijer, [16], [15]	0.9417	0.6898	0.0304



(a) Original (b) Segmentation, cropped at boundary (c) Ground truth

Figure 5: Results of our algorithm on DRIVE-database

6.2 Evaluation on DRIVE database

As the segmentation steps 1-9 are the most important steps of our method, they are evaluated using a publicly available database. When applying our method to the DRIVE database, we follow the standard rules described in detail in [14] and [24]. Table 2 shows the parameter settings of our experiments. The noise scale σ refers to the standard deviation of the Gaussian presmoothing that was performed for regularisation. The integration scale ρ determines the size of the applied local Radon kernel. The numbers in the last line in Table 2 indicate the minimal size (in terms of numbers of pixels) of the connected components that are displayed in the final result. Connected components that have less pixels than the user defined number have been eliminated. The specific results of our experiments are displayed in Table 3. For comparison and examples, see Fig. 5. The bold letters indicate the best results in the following categories:

$$\text{accuracy} = \frac{\text{number of correctly classified pixels}}{\text{total number of pixels}},$$

$$\text{true positives} = \frac{\text{number of detected vessel pixels}}{\text{total number of true vessel pixels}},$$

$$\text{false positives} = \frac{\text{number of falsely detected vessel pixels}}{\text{total number of non-vessel pixels}}.$$

We observe that our method leads to very good results that are competitive to modern approaches from the literature. Two advantages of our algorithm are the smoothness and connectedness of the detected vessels.

6.3 Evaluation on our own database

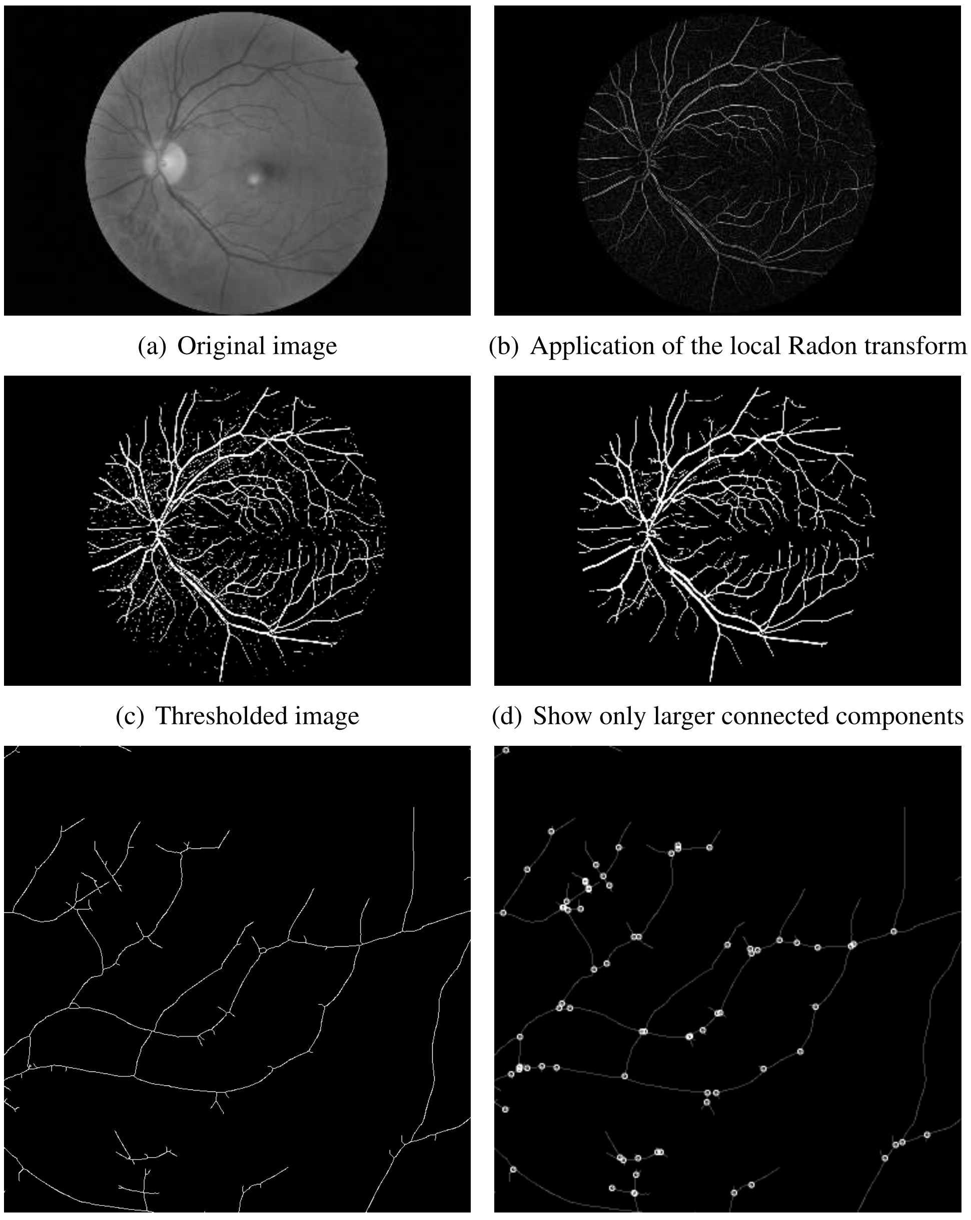
Fig. 6 shows the results on our own clinical database, high resolution. The figure on the top left is the original image, the figure on the top right is the result of the application of the second derivative of the local Radon transform. The figures in the middle are the segmentation step followed by a connected component analysis. The bottom row shows the skeleton and the branching points. A careful evaluation by an ophtalmologist has confirmed the high quality that can be achieved with our automised system.

7 Conclusion

We have shown that for analysing retinal images, high quality and high speed requirements do not exclude each other. To this end, we have designed a fully automatic system that is based on the local Radon transform and uses only algorithms that are well parallelisable on GPUs by means of CUDA. They enable the analysis of large databases in a short time span. For instance, one can analyse more than $20,000$ images of size 2048×1536 in about 3 hours on an NVIDIA Geforce GTX680. Moreover, interactive image analysis during the examination of the patient is no problem at all. Since the necessary graphics hardware is widely available for a few hundred dollars, our research has a good potential of becoming widely used within the ophtalmologic community.

Acknowledgements

We thank the authors of the DRIVE database for making their database available and thus allowing us to evaluate our results. Furthermore we like to thank Rüdiger Leilich for providing the tool ”Algo-Verifier” for better visualisation and documentation of the results in our specifically designed database.


(a) Original image (b) Application of the local Radon transform

(c) Thresholded image (d) Show only larger connected components

(e) Skeleton (f) Skeleton with branching points

Figure 6: Results of the skeletonisation algorithm using CUDA - own clinical database, resolution 4288×2848

References

[1] Chanwimaluang T, Fan G (2003) An efficient algorithm for extraction of anatomical structures in retinal images. In: Proc. IEEE International Conference on Image Processing, vol 1, pp 1093–1096

[2] Chapman N, Witt N, Bharat A, et al (2001) Computer algorithms for the automated measurement of retinal arteriolar diameters. British Journal of Ophtalmology 85:74–79

[3] Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to algorithms. MIT Press

[4] Frangi AF, Niessen WJ, Vincken KL, Viergever MA (1998) Multiscale vessel enhancement filtering. In: Medical Image Computing and Computer-Assisted Intervention- MICCAI'98, Lecture Notes in Computer Science, vol 1496, Springer, pp 130–137

[5] Gang L, Chutape O, Krishnan M (2002) Detection and measurement of vessels in fundus images using amplitude modified second-order Gaussian filter. IEEE Transactions on Biomedical Engineering 49(2):168–172

[6] Gao X, Bharat A, Hughes A, et al (1997) Towards retinal vessel parameterization. In: Medical Imaging 1997: Image Processing, SPIE Proc., vol 3034, pp 734–744

[7] van Ginkel M (2002) Image analysis using orientation space based on steerable filters. PhD thesis, Delft University of Technology, URL http://www.ph.tn.tudelft.nl/˜michael/publications.html

[8] Gonzalez R, Woods R (2002) Digital Image Processing. Prentice Hall, New Jersey

[9] Hoover A, Kouznetsova V, Goldbaum M (2000) Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Transactions on Medical Imaging 19(3):203–210

[10] Hwu WMW (2011) GPU Computing Gems Emerald Edition. Morgan Kaufmann

[11] Kirk DB, Hwu WMW (2010) Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann

[12] Krause M (2006) Corner detection in digital images using local tomography. Bachelor's thesis, Dept. of Mathematics and Computer Science, Saarland University

[13] Lowell J, Hunter A, Steel D, Basu A, Kennedy RL (2004) Measurement of retinal vessel widths from fundus images based on 2-D modeling. IEEE Transactions on Medical Imaging 23(10):1196–1204

[14] Mendonça AM, Campilho A (2006) Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reonstruction. IEEE Transactions on Medical Imaging 25(9):1200–1213

[15] Niemeijer M, van Ginneken B (2002) Drive database. URL www.isi.uu.nl/Research/Databases/DRIVE/results.php

[16] Niemeijer M, Staal J, van Ginneken B, Loog M, Abrámoff M (2004) Comparative study of retinal vessel segmentation methods on a new publicly available database. In: Proc. SPIE Medical Imaging, M. Fitzpatrick and M. Sonka, Eds., vol 5370, pp 648–656

[17] NVIDIA (2012) Nvidia. URL http://www.nvidia.com

[18] Ricci E, Perfetti R (2007) Retinal blood vessel segmentation using line operators and support vector classification. IEEE Transactions on Medical Imaging 26(10):1357–1365

[19] Sanders J, Kandrot E (2011) CUDA by example, An Introduction to General-Purpose GPU programming. Addison Wesley

[20] Savarimuthu TR, Kjaer-Nielsen A, Sorensen AS (2011) Real-time medical video processing, enabled by hardware accelerated correlations. Journal of Real-Time Image Processing 6:187–197

[21] Soares JVB, Leandro JJG, Cesar RM Jr, Jelinek HF, Cree MJ (2006) Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification. IEEE Transactions on Medical Imaging 25(9):1214–1222

[22] Sofka M, Stewart C (2006) Retinal vessel centerline extraction using multiscale matched filters, confidence and edge measures. IEEE Transactions on Medical Imaging 25(12):1531–1546

[23] Soille P (1999) Morphological Image Analysis. Springer

[24] Staal J, Abrámoff MD, Viergever MA, van Ginneken B (2004) Ridge-based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging 23(4):501–509

[25] Vermeer K, Vos F, Lemij H, Vossepoel A (2004) A model based method for retinal blood vessel detection. Computers in Biology and Medecine 34:209–219

[26] Wang L, Bhalerao A, Wilson R (2007) Analysis of retinal vasculature using a multiresolution Hermite model. IEEE Transactions on Medical Imaging 26(2):137–152