
ACTA CYBERNETICA

Editor-in-Chief: János Csirik (Hungary)

Managing Editor: Csanád Imreh (Hungary)

Assistant to the Managing Editor: Attila Tanács (Hungary)

Associate Editors:

Luca Aceto (Iceland)
Mátyás Arató (Hungary)
Stephen L. Bloom (USA)
Hans L. Bodlaender (The Netherlands)
Lothar Budach (Germany)
Horst Bunke (Switzerland)
Bruno Courcelle (France)
Tibor Csendes (Hungary)
János Demetrovics (Hungary)
Bálint Dömölki (Hungary)
Zoltán Ésik (Hungary)
Zoltán Fülöp (Hungary)
Ferenc Gécseg (Hungary)

Jozef Gruska (Slovakia)
Tibor Gyimóthy (Hungary)
Helmut Jürgensen (Canada)
Zoltan Kato (Hungary)
Alice Kelemenová (Czech Republic)
László Lovász (Hungary)
Gheorghe Păun (Romania)
András Prékopa (Hungary)
Arto Salomaa (Finland)
László Varga (Hungary)
Heiko Vogler (Germany)
Gerhard J. Woeginger (The Netherlands)

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in \LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage <http://www.inf.u-szeged.hu/actacybernetica/>.

EDITORIAL BOARD

Editor-in-Chief: **János Csirik**
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
csirik@inf.u-szeged.hu

Managing Editor: **Csanád Imreh**
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
cimreh@inf.u-szeged.hu

Assistant to the Managing Editor:

Attila Tanács
Department of Image Processing
and Computer Graphics
University of Szeged, Szeged, Hungary
tanacs@inf.u-szeged.hu

Associate Editors:

Luca Aceto
School of Computer Science
Reykjavík University
Reykjavík, Iceland
luca@ru.is

Mátyás Arató
Faculty of Informatics
University of Debrecen
Debrecen, Hungary
arato@inf.unideb.hu

Stephen L. Bloom
Computer Science Department
Stevens Institute of Technology
New Jersey, USA
bloom@cs.stevens-tech.edu

Hans L. Bodlaender
Institute of Information and
Computing Sciences
Utrecht University
Utrecht, The Netherlands
hansb@cs.uu.nl

Lothar Budach
Department of Computer Science
University of Potsdam
Potsdam, Germany
lbudach@haiti.cs.uni-potsdam.de

Horst Bunke
Institute of Computer Science and
Applied Mathematics
University of Bern
Bern, Switzerland
bunke@iam.unibe.ch

Bruno Courcelle
LaBRI
Talence Cedex, France
courcell@labri.u-bordeaux.fr

Tibor Csendes
Department of Applied Informatics
University of Szeged
Szeged, Hungary
csendes@inf.u-szeged.hu

János Demetrovics
MTA SZTAKI
Budapest, Hungary
demetrovics@sztaki.hu

Bálint Dömölki
IQSOFT
Budapest, Hungary
domolki@iqsoft.hu

Zoltán Ésik

Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Zoltán Fülöp

Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg

Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska

Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Tibor Gyimóthy

Department of Software Engineering
University of Szeged
Szeged, Hungary
gyimothy@inf.u-szeged.hu

Helmut Jürgensen

Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Zoltan Kato

Department of Image Processing
and Computer Graphics
Szeged, Hungary
kato@inf.u-szeged.hu

Alice Kelemenová

Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász

Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun

Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa

Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa

Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga

Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler

Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

Gerhard J. Woeginger

Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

CONFERENCE OF PHD STUDENTS IN COMPUTER SCIENCE

Guest Editor:

Kálmán Palágyi

Department of Image Processing and Computer Graphics
University of Szeged
Szeged, Hungary
palagyik@inf.u-szeged.hu

Preface

The seventh Conference for PhD Students in Computer Science (CSCS) was organized by the Department of Computer Science of the University of Szeged (SZTE) and held in Szeged, Hungary from June 29 to July 2, 2010. The members of the Scientific Committee were the following representants of the Hungarian doctoral schools in computer science: András Benczúr (ELTE), Hasszan Charaf (BME), Tibor Csendes (SZTE), László Cser (BCE), János Csirik (Chair, SZTE), János Demetrovics (ELTE), József Dombi (SZTE), Zoltán Ésik (SZTE), Ferenc Fiedler (PE), Zoltán Fülöp (SZTE), Aurél Galántai (ÓE), Tibor Gyimóthy (SZTE), Zoltán Horváth (ELTE), Csanád Imreh (SZTE), Zoltán Kató (SZTE), Zoltán Kása (Sapientia EMTE), László Keviczky (SZIE), János Kormos (DE), László Kozma (ELTE), János Levendovszky (BME), Eörs Máté (SZTE), Attila Pethő (DE), András Recski (BME), Lajos Rónyai (Co-chair, SZTAKI), Tamás Roska (PPKE), Endre Selényi (BME), Tamás Szirányi (SZTAKI), and Tibor Tóth (ME). The members of the Organizing Committee were Balázs Bánhelyi, Tamás Gergely, Zoltán Kincses, and Kálmán Palágyi.

There were more than 60 participants and 51 talks in several fields of computer science and its applications. The talks were going in sections in computer graphics, computer networks, database theory, discrete mathematics, distributed computing, image and signal processing, numerical analysis, optimization, software engineering, and stochastic processes. The talks of the students were completed by four plenary talks of leading scientists: Bruno Buchberger (Johannes Kepler University, Hagenberg, Austria), Agoston E. Eiben (Vrije Universiteit Amsterdam, The Netherlands), Aurél Galántai (Óbuda University, Hungary), and Mihály Kovács (The University of Otago, New Zealand).

Two scientific journals, viz. *Periodica Polytechnica* (Budapest) and *Acta Cybernetica* (Szeged) offered students to publish the paper version of their presentations after a selection and review process. Altogether 24 manuscripts were submitted for publication. The present special issue of *Acta Cybernetica* contains 12 such papers.

The full program of the conference, the collection of the abstracts and further information can be found at <http://www.inf.u-szeged.hu/~cscs>.

On the basis of our repeated positive experiences, the conference will be organized in the future, too. According to the present plans, the next meeting will be held in July 2012 in Szeged.

Kálmán Palágyi
Guest Editor

A Stochastic Approach to Improve Macula Detection in Retinal Images

Bálint Antal and András Hajdu*

Abstract

In this paper, we present an approach to improve detectors used in medical image processing by fine-tuning their parameters for a certain dataset. The proposed algorithm uses a stochastic search algorithm to deal with large search spaces. We investigate the effectiveness of this approach by evaluating it on an actual clinical application. Namely, we present promising results with outperforming four state-of-the-art algorithms used for the detection of the center of the sharp vision (macula) in digital fundus images.

Keywords: biomedical image processing, simulated annealing, learning and adaptive systems

1 Introduction

Diabetic Retinopathy (DR) is the most common cause of blindness in the developed countries. Nowadays, the automatic screening of DR received much attention in the medical imaging community [1], [7], [9], since replacing a resource-demanding and expensive manual screening is a very challenging task. Automatic screening is based on the analysis of retinal images taken at eye hospitals. One class of the difficulties originates from the use of different kinds of retinal images, which leads to varying performance in the anatomy or lesion detection processes. Some detectors are based on machine learning, while others consider non-training approaches.

In this paper, we present a technique to improve a detection algorithm on retinal images via a learning-based approach. The idea behind this technique is to fine-tune the parameter setup for a certain detector. Since the selection of the optimal parameter setup usually traverses a large search space, we decided to use a stochastic approach, simulated annealing for this task. To demonstrate the effectiveness of this technique, we present a novel macula detector and show that the proposed framework improves detection performance. The contribution in this particular area is justified by the fact that the detection of macula involves the lowest number of reported works in the field of DR screening research [16]. A comparative analysis

*University of Debrecen, Hungary. E-mail: {antal.balint, hajdu.andras}@inf.unideb.hu

reveals that our tuned algorithm outperforms other state-of-the-art algorithms in the field.

The main contributions of the paper are organized into the following sections:

1. A stochastic approach to improve detector performance is introduced. We also discuss the advantages of using simulated annealing over stochastic hill climbing (Section 2).
2. We show how to adopt the simulated annealing based search method to improve the performance of the proposed macula detector (Section 3).
3. A novel macula detector is proposed, which, in addition to its good performance, can be easily fine-tuned by a search algorithm (Section 3.1).
4. We define an error measure to efficiently characterize macula detection performance (Section 3.3).
5. We evaluate the performance of our macula detector using the proposed tuning also in comparison with four state-of-the-art algorithms (Section 4).

2 A stochastic approach to improve detector performance

In this section, we present our approach to select an optimal parameter setup for a detector algorithm. For this task, we have to prepare for a large search space, since these algorithms may operate with several parameters. In literature, stochastic hill climbing is often recommended [10] [13]. Stochastic hill climbing is based on the idea that using random jumps between the neighbouring elements of the search space converges faster to the extrema than using exhaustive enumerations. An element is accepted, if it provides better result than the current extremum. This approach is an effective solution for many problems, but it can get stuck in a local extrema in search spaces with many peaks.

To overcome this difficulty, we used a simulated annealing-based method. Simulated annealing [5] avoids getting stuck in local extrema by using a random acceptance function for rejected elements. That is, if an element does not provide a better result than the current one, it is still accepted if the acceptance function allows that. See Figure 1 for a visual comparison of hill climbing and simulated annealing.

The formal description of the algorithm can be given as follows:

Algorithm 1.: Parameter setup selection by simulated annealing.

Input:

- An initial temperature $T \in \mathbb{R}$.
- A minimal temperature $T_{min} \in \mathbb{R}$.

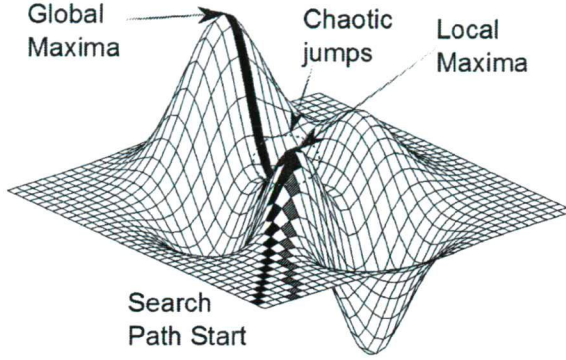


Figure 1: The path of the hill climbing and the simulated annealing algorithm is represented with gray and black colors, respectively. While hill climbing reaches only the local optimum, simulated annealing can continue towards the global optimum by using chaotic jumps.

- A temperature change $q \in \mathbb{R}$ with $(0 \leq q \leq 1)$.
- A search space $S \subset \mathbb{R}^n$ with $s \in S$ is a parameter setup.
- A function $r(X)$, which chooses a random element x from a set X .
- A function $accept : \mathbb{R}^4 \rightarrow \{true, false\}$, which is defined as the follows:

$$accept(e, e_i, T, y) = \begin{cases} true, & \text{if } \exp\left(\frac{e - e_i}{T}\right) > y, \\ false, & \text{otherwise.} \end{cases}$$

- An energy function $E : S \rightarrow \mathbb{R}$.

Output: $s_{optimal} \in S$, where $E(s_{optimal}) = \min_{s \in S} E(s)$. That is, $s_{optimal}$ is the parameter setup minimizing the energy function E .

-
1. $s \leftarrow r(S)$
 2. $e \leftarrow E(s)$
 3. $S \leftarrow S - \{s\}$
 4. **while** $S \neq \emptyset$ or $T < T_{min}$ **do**
 5. $s_i \leftarrow r(S)$
 6. $e_i \leftarrow E(s_i)$
 7. $S \leftarrow S - \{s_i\}$
 8. **if** $e_i < e$ **then**
 9. $s \leftarrow s_i$

```

10.    $e \leftarrow e_i$ 
11.    $T \leftarrow T \cdot q$ 
12.   else
13.      $y \leftarrow r(\mathbb{R})$ 
14.     if  $\text{accept}(e, e_i, T, y) = \text{true}$  then
15.        $s \leftarrow s_i$ 
16.        $e \leftarrow e_i$ 
17.        $T \leftarrow T \cdot q$ 
18.     end if
19.   end if
20. end while
21. return  $s$ 

```

3 Using the proposed approach: an example

In this section, we present an example to demonstrate the power of the proposed method. For this task, we chose a novel approach for macula detection in retinal images, which algorithm requires only two parameter to be optimized. Our proposed approach for obtaining the optimal parameter setup can be adapted to any similar problem, as well.

3.1 Macula detection

The macula is the central region of sharp vision in the human eye, with its center referred to as the fovea (see Figure 2). Any lesions (e.g. microaneurysms) which appear here can lead to severe loss of vision. Therefore, the efficient detection of the macula is essential in an automatic screening system for diabetic retinopathy.

3.2 A novel algorithm for macula detection

In this section, we present a novel approach to detect macula in a retinal image. As we can see later on, this algorithm outperforms state-of-the-art macula detectors with the use of the proposed framework for optimal parameter setup.

The proposed macula detection algorithm can be formulated as follows:

Algorithm 2.: A novel macula detector

Input:

- A digital retinal image I in 24 bit *RGB* format.
- A parameter $q \in \mathbb{R}$ with $0 \leq q \leq 1$ to adjust of the mask size in the median filtering step.
- A threshold $t \in [-255, \dots, 255]$.

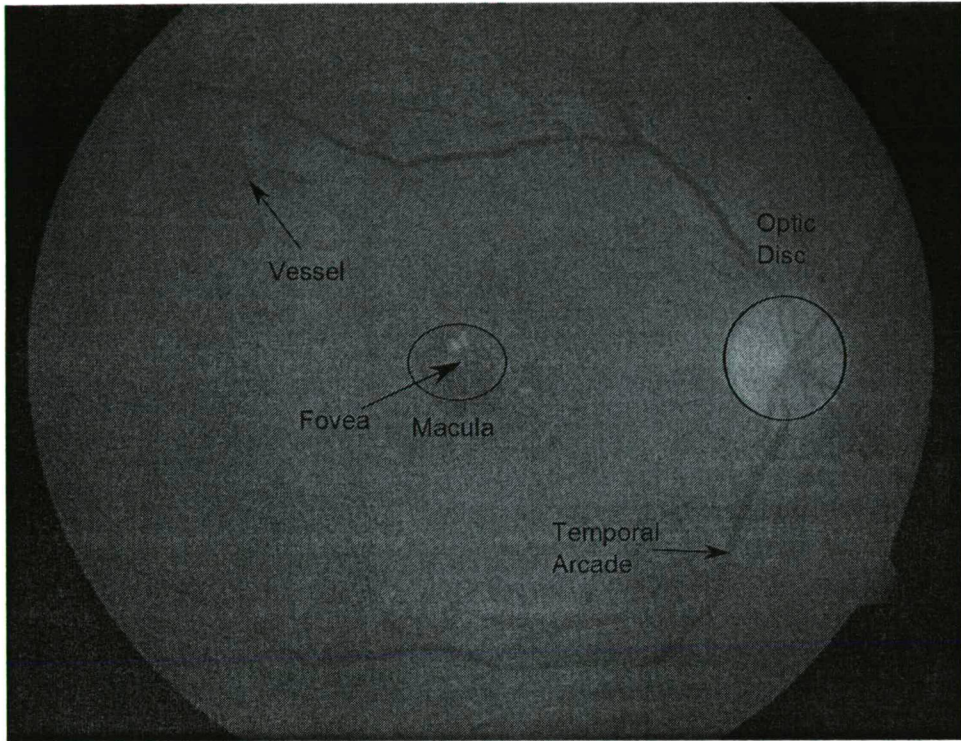


Figure 2: A sample fundus image with the main anatomical parts annotated.

Output: An image containing the macula region of the eye.

1. Extract the green intensity channel G from I .
 2. Let $A = \lceil \text{Min}(\text{width}(I), \text{height}(I)) \cdot q \rceil$.
 3. Produce image M with the same size as G by applying median filtering [12] on G with a mask size $A \times A$.
 4. Create the difference image $D = G - M$.
 5. Produce a binary image B by assigning all pixels with larger intensity than t in the D to the foreground, while the rest to its background.
 6. Select the largest binary component to locate the macula.
-

The results after each step of the algorithm can also be observed in Figure 3.

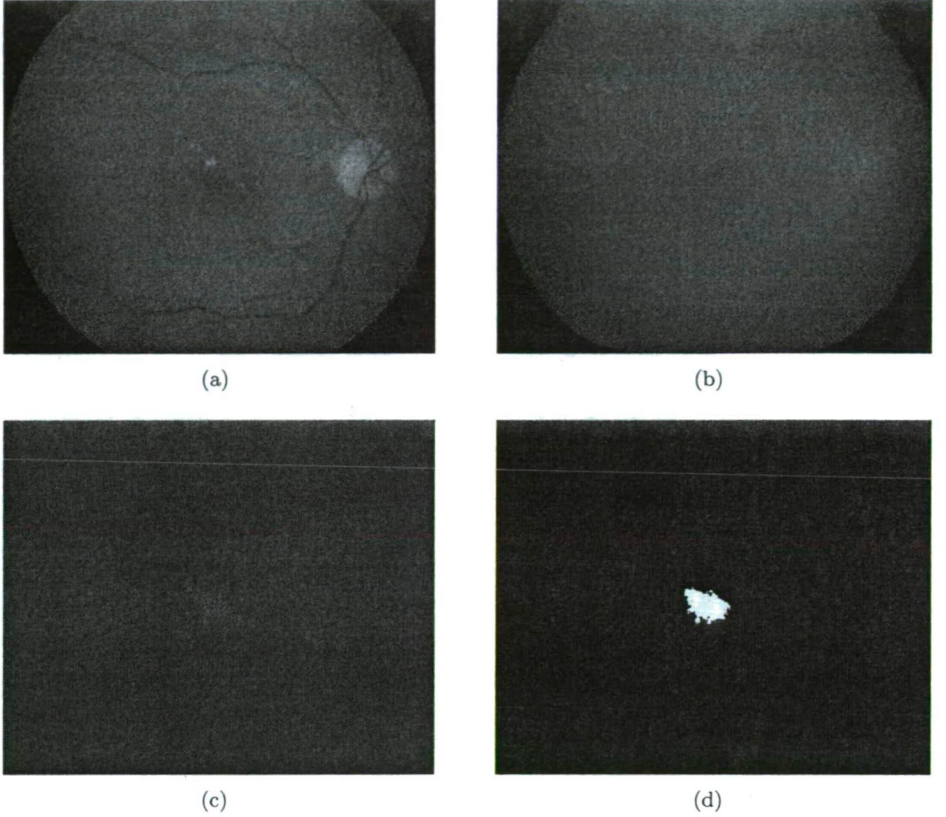


Figure 3: Steps of the proposed macula detection: (a) The green channel of the input image. (b) The result of the median filtering. (c) The difference image. (d) The binary image after thresholding and largest component selection.

3.3 Error measurement of macula detectors

To select the optimal parameter setup for the above detector algorithm, we need a proper energy function to be minimized. An obvious choice for this task is to minimize the distance of the centroid of the macula found by the detector and the manually selected center of the macula for each image in a dataset. To avoid overtraining, we also take into account the distance from the optic disc (see Figure 2), as the macula and the optic disc are spatially constrained [11].

Thus, we define the following energy function for this problem:

$$E = \sum_{I \in DS} d(M_{alg}(I), M_{hm}(I)) + \sum_{I \in DS} |d(M_{alg}(I), O) - MO_{avg}|,$$

where

- DS is the dataset,
- d denotes the 2D Euclidean distance,
- M_{alg} is the centroid pixel of the detected macula,
- M_{hm} is the manually selected macula center,
- O is the manually selected optic disc center,
- MO_{avg} is the average Euclidean distance of the manually selected macula and optic disc center for the dataset DS .

4 Comparative results

We evaluate our method by comparing it with four other state-of-the-art macula detectors (Section 4.1) on different datasets (Section 4.2). As our results will show, the novel macula detector outperforms the others after finding its optimal parameter setup.

4.1 State-of-the-art macula detection algorithms

In this section, we list four macula detection algorithms, which are involved in our comparative analysis. The parameters of the algorithms were set according to the corresponding recommendations in literature.

4.1.1 Petsatodis et al. [8]

In [8] a region of interest (ROI) is defined to process macula detection. A Gaussian low-pass filter is applied to smooth the image. The statistical mean and standard deviation of the ROI area are used to compute a threshold for segmentation to get binary objects. The object that is located nearest to the center of the ROI is labelled as macula. Its center of mass is considered to be the center of the macula. However, we did some modification to this approach, because it is not mentioned how this ROI is defined; therefore we applied the smoothing to the whole image using a large kernel (70×70 pixels with $\sigma = 10$) so that vascular network and small patches do not interfere in detection. Then, an iterative thresholding process is launched to generate a set of binary images corresponding to different threshold values. In each binary image, the component satisfying the area and distance from the center constraints are identified, and the component found nearest to the center with minimum area is marked as macula.

4.1.2 Sekhar et al. [11]

In [11] a region of interest (ROI) for macula is defined regarding its spatial relationship to the optic disc. That is the portion of a sector subtended at the center of the optic disc by an angle of 30° above and below the line between this center and

the center of the retinal image disc. The macula is identified within this ROI by iteratively applying a threshold, and then applying morphological opening (erosion followed by dilation) on the resulting blob. The value of the threshold parameter is selected such that the area of the smoothed macula region is not more than 80% of that of the detected optic disc. The fovea is simply determined as the centroid of this blob.

4.1.3 Fleming et al. [2]

Fleming et al. [2] proposed to identify the macular region based on the information of the temporal arcade and OD center. First, the arcade is found by using semielliptical templates. Next, the optic disc is detected by using Hough transformation with circular templates having diameters from 0.7 to 1.25 OD diameter (DD). Finally, the fovea was detected by finding the maximum correlation coefficient between the image and a foveal model. The search was restricted to a circular region with diameter 1.6 DD centered on a point that is 2.4 DD from the optic disc and on a line between the detected optic disc and the center of the semi-ellipse fitted to the temporal arcades.

4.1.4 Zana et al. [17]

Zana et al. [17] presented a region merging algorithm based on watershed cell decomposition and morphological operations for macula recognition. After noise removal, morphological closing followed by opening is performed to remove the small dark holes and white spots. A watershed based decomposition of the gradient image into cells is done, and the cell with darkest gray level inside the macula is selected as the first step of a merging algorithm. A complex criterion based on the gray values and of edges of the filtered image is calculated to merge the cells of the macula, while rejecting perifoveal inter-capillary zones in order to produce the contour of the macula.

4.2 Datasets

We have tested our approach on 199 images from three publicly available data sources: DiaretDB0 [3], DiaretDB1 [4] and DRIVE [15]. The characteristic properties of these datasets can be seen in Table 1. We have selected the optimal parameter setup for each dataset using a separate training subset of a total of 60 images (20 images from each dataset). For each dataset, the ground truth are used only for parameter selection.

4.3 Results

Table 2 shows the selected optimal parameters for each dataset. The size parameter q and the threshold parameter t have been found by the proposed stochastic approach. Each dataset performed optimally using a different parameter setup. We have evaluated our approach in two aspects [6]: whether the detected macula

Dataset	Images	Normal	DR	FOV	Resolution
DiaretDB0	130	20	110	50	1500x1152
DiaretDB1	89	5	84	50	1500x1152
DRIVE	40	33	7	45	768x584

Table 1: Properties of the datasets.

Dataset	q	t
DiaretDB0	0.6	0
DiaretDB1	0.6	5
Drive	0.7	0

Table 2: Parameters selected by the proposed algorithm for macula detection.

center falls into the 0.5DD (Optic Disc Diameter) distance of the manually selected macula center and we also measured the Euclidean distance of them (calculated on normalized images). Table 3 and 4 contain the quantitative results using these measures, respectively. We disclose the results for each macula detector evaluated in all dataset. For the more straight-forward comparison, we also calculated the simple average of these performance values. In the terms of the first measure, the use of the proposed algorithm on the novel macula detector resulted in a 85% average accuracy, while the second best method only earned 77%. However, in the terms Euclidean error it is only third in the comparison, mainly because of its difficulties on the DRIVE database.

Dataset	Petsatodis	Sekhar	Fleming	Zana	Proposed
DiaretDB0	68%	72%	85%	63%	86%
DiaretDB1	62%	76%	79%	71%	92%
DRIVE	66%	76%	53%	82%	68%
Average	66%	74%	77%	69%	85%

Table 3: Percentage of detected macula centers falling in the correct region.

5 Conclusion

In this paper, we have presented an approach to improve detection algorithms by fine-tuning their parameters. For this task, we have used a simulated annealing-based search algorithm. As our experiments have proved, this approach is capable of improving a detector that outperforms state-of-the-art algorithms in the field of macula detectors. As a future work, the selection of different preprocessing methods for the dataset can further improve the detection of the macula. In addition, both simulated annealing [14] and the proposed detector could be implemented in parallel to reduce their computational needs.

Dataset	Petsatodis	Sekhar	Fleming	Zana	Proposed
DiaretDB0	26.59	26.85	37.82	24.11	24.02
DiaretDB1	26.32	27.45	35.67	24.77	25.72
DRIVE	18.15	26.20	37.29	20.85	30.25
Average	23.69	26.83	36.92	23.24	26.75

Table 4: Average euclidean distance of the detected macula centers from the manually selected ones.

Acknowledgement

This work was supported in part by the János Bolyai grant of the Hungarian Academy of Sciences, and by the TECH08-2 project DRSCREEN - Developing a computer based image processing system for diabetic retinopathy screening of the National Office for Research and Technology of Hungary (contract no.: OM-00194/2008, OM-00195/2008, OM-00196/2008). We also acknowledge the Moorefields Eye Hospital, London for their clinical support. We are thankful to Brigitta Nagy and Ignác Csősz for their technical assistance.

References

- [1] Abramoff, M. D., Reinhardt, J. M., Russell, S. R., Folk, J. C., Mahajan, V. B., Niemeijer, M., and Quellec, G. Automated early detection of diabetic retinopathy. *Ophthalmology*, 117(6):1147–1154, 2010.
- [2] Fleming, A. D., Philip, S., Goatman, K. A., Olson, J. A., and Sharp, P.F. Automated assessment of diabetic retinal image quality based on clarity and field definition. *Investigative Ophthalmology and Visual Science*, 47:1120–1125, 2006.
- [3] Kauppi, T., Kalesnykiene, V., Kamarainen, J.K., Lensu, L., Sorri, I., Uusitalo, H., Kalviainen, H., and Pietila, J. Diaretdb0: Evaluation database and methodology for diabetic retinopathy algorithms. Technical report, Lappeenranta University of Technology, Lappeenranta, Finland, 2006.
- [4] Kauppi, T., Kalesnykiene, V., Kmrinen, J.K., Lensu, L., Sorri, I., Raninen, A., Voutilainen, R., Uusitalo, H., Klviinen, H., and Pietil, J. Diaretdb1 diabetic retinopathy database and evaluation protocol. *Proc. of the 11th Conf. on Medical Image Understanding and Analysis (MIUA2007)*, pages 61–65, 2007.
- [5] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220:671–680, May 13, 1983.
- [6] Kovacs, L., Qureshi, R.J., Nagy, B., Harangi, B., and Hajdu, A. Graph based detection of optic disc and fovea in retinal images. In *Soft Computing Applications (SOFA), 2010 4th International Workshop on*, pages 143–148, 2010.

- [7] Niemeijer, M., van Ginneken, B., Cree, M.J., Mizutani, A., Quéllec, G., Sanchez, C.I., Zhang, B., Hornero, R., Lamard, M., Muramatsu, C., Wu, X., Cazuguel, G., You, J., Mayo, A., Li, Q., Hatanaka, Y., Cochener, B., Roux, C., Karray, F., Garcia, M., Fujita, H., and Abramoff, M.D. Retinopathy online challenge: Automatic detection of microaneurysms in digital color fundus photographs. *IEEE Transactions on Medical Imaging*, 29(1):185–195, 2010.
- [8] Petsatodis, T. S., Diamantis, A., and Syrcos, G. P. A complete algorithm for automatic human recognition based on retina vascular network characteristics. *Era1 International Scientific Conference, Peloponnese, Greece*, 16-17 September 2006.
- [9] Ravishankar, S., Jain, A., and Mittal, A. Automated feature extraction for early detection of diabetic retinopathy in fundus images. In *CVPR*, pages 210–217. IEEE, 2009.
- [10] Ruta, D. and Gabrys, B. Classifier selection for majority voting. *Information Fusion*, 6(1):63 – 81, 2005. Diversity in Multiple Classifier Systems.
- [11] Sekhar, S., Al-Nuaimy, W., and Nandi, A. K. Automated localization of optic disc and fovea in retinal fundus images. *16th European Signal Processing Conference, Lausanne, Switzerland*, 2008.
- [12] Sinha, P. K. and Hong, Q. H. An improved median filter. *IEEE Transactions on Medical Imaging*, 9(3):345–346, 1990.
- [13] Sivanandam, S. N. and Deepa, S. N. *Introduction to Genetic Algorithms*. Springer, 2008.
- [14] So, O. and Özdamar, L. Parallel Simulated Annealing Algorithms in Global Optimization. *Journal of Global Optimization*, 19:27–50, 2001.
- [15] Staal, J., Abramoff, M. D., Niemeijer, M., Viergever, M. A., and van Ginneken, B. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23:501–509, 2004.
- [16] Winder, R.J., Morrow, P.J., McRitchie, I.N., Bailie, J.R., and Hart, P.M. Algorithms for digital image processing in diabetic retinopathy. *Computerized Medical Imaging and Graphics*, 33(8):608–622, 2009.
- [17] Zana, F., Meunier, I., and Klein, J. C. A region merging algorithm using mathematical morphology: application to macula detection. In *ISMM '98: Proceedings of the fourth international symposium on Mathematical morphology and its applications to image and signal processing*, pages 423–430, Norwell, MA, USA, 1998. Kluwer Academic Publishers.

Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints*

Tamás Bartók[†] and Csanád Imreh[‡]

Abstract

In this paper we introduce a new, pickup and delivery vehicle routing model where weight limits and also packing constraints are taken into account. In the model the vehicles have to transport 3-dimensional boxes from their pickup points into their delivery points. The boxes have weights and the vehicle has to satisfy a weight limit. We present a heuristic algorithm for the solution of the problem. The efficiency of the algorithm is evaluated by an experimental analysis.

Keywords: vehicle routing, multidimensional packing

1 Introduction

In logistics there are two important problems related to combinatorial optimization. One is the routing of the vehicles and the other problem is the container loading. Both areas have huge literature and many different models have been investigated. If both problems are taken into account then more difficult models appear, but they give a more adequate description of the real life problems.

In the area of vehicle routing many models are investigated, one can find a detailed description of the models in the surveys [4] and [16]. Usually the goal is to minimize the cost of the transportation, but in some recent works other cost functions like minimizing pollution are also considered (see [2]). One of the most important subfields of vehicle routing is the area of pickup and delivery problems. In these problems the requests are goods with a pickup point and a delivery point and the vehicles must transport them from the pickup point to the delivery point with minimal cost. These problems are NP-hard and several exponential time exact solution algorithms and metaheuristics are developed for their solution. One can find an overview about pickup and delivery problems in the survey [3].

Loading the vehicles leads to a 3-dimensional bin packing problem. If weight limits are taken into account, then we receive a common generalization of the vector

*Cs. Imreh was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

[†]Institute of Informatics, University of Szeged, E-mail: tbartok@inf.u-szeged.hu

[‡]Institute of Informatics, University of Szeged, E-mail: cimreh@inf.u-szeged.hu

packing and box packing problems (see [1]). Algorithms for models, where both the routing and loading of vehicles are considered have been published only in the recent years. The capacitated vehicle routing problem (where each request has to be collected into a depot) with two dimensional loading constraints has been investigated first in the paper [9]. An algorithm which gives the exact optimal solution is presented for the case of two dimensional loading constraints in [10]. Improved tabu search algorithms are presented for the solution of that problem in [9] and [17]. An algorithm based on ant colony optimization is presented in [6]. The first paper on capacitated vehicle routing with 3-dimensional loading constraints is presented in [8]. An improved tabu search based algorithm for 3-dimensional constraints is presented in [17], and an ant colony based algorithm is given in [7]. Pickup and delivery vehicle routing problems with two dimensional loading constraints are presented in [13]. An overview about the results on vehicle routing problems with loading constraints can be found in [11] and [18].

In this paper we consider a pickup and delivery problem with 3-dimensional loading constraints and with weight limit on the vehicles. As far as we know no algorithm for this model is presented in the literature. In the next section we give the mathematical model which is used in this paper. Then in Section 3 a heuristic algorithm is presented to solve the problem. Section 4 contains the description of the tests, we used to analyze the algorithm.

2 The mathematical model

In this model we are given an undirected simple graph $G = (V, E)$ which describes the road system which can be used by the vehicles. The input of the problem is a list of demands denoted by D and a list of vehicles denoted by R . The demands have the following parameters:

- the size which is a 3-dimensional box given by the sides x_j, y_j, z_j ,
- the weight which is a positive real number w_j ,
- pickup and delivery points: $s_j \in V, e_j \in V$.

The vehicles have the following parameters

- the size of the cargo which is a 3-dimensional box given by the sides X_v, Y_v, Z_v ,
- the weight limit which is a positive real number W_v ,
- the speed of the vehicle sp_v ,
- the start and end point of the vehicle s_v and e_v ,
- a cost function c_v which defines for each edge of G the travelling cost for vehicle v , we suppose that this is proportional to the distance, thus the time spent by the vehicle travelling on edge e is $c_v(e)/sp_v$

- time limit of the vehicle l_v which gives an upper bound on the time which the vehicle can travel before delivering its last demand

Our goal is to transport all of the demands by the given vehicles by a minimal cost solution, while not violating the defined constraints. In this model we do not allow to change the vehicle during the transportation of a demand, the vehicle must pick up the demand at its pick up point and deliver it to the delivery point. A vehicle can transport an unlimited number of freights at the same time as long as the size and weight constraints are not violated. We suppose that the cost of the solution is the total cost of the routes done by the vehicles. During the route of the vehicles the constraints must be satisfied at each time, which means that the demands have to be packed into it without overlapping and the total weight is not allowed to be more than the weight limit. We note that in this model rotation of the items is not allowed. This assumption is realistic for automation based container loading. Moreover there is an extra assumption on the routes: each route has to be finished before the time limit of the vehicle. Here we do not take into account the time which is used to return to the end point, the limit is on the last delivery time. On the other hand one can easily modify our algorithm to the case when the time limit is for the arrival time at the end point.

Therefore the solution can be described as follows: Each vehicle v has a travel plan P_v which contains a list $p_1, \dots, p_{k(v)}$ of vertices which - with the graph being simple - is a walk and for each vertex two sets are given: $in(p_i) \subseteq S$ contains the demands packed into the vehicle at vertex $p(i)$ and $out(p_i) \subseteq S$ contains the demands packed out from the vehicle at vertex $p(i)$, where S is a set of (a, b) couples ($a, b \in V$), where there is at least one transport request from a to b .

The solution is feasible if the following conditions are satisfied:

- $s_j = p_i$ for each $j \in in(p_i)$ and $v_j = p_i$ for each $j \in out(p_i)$, which means that a demand can be packed into a vehicle at its pickup point and can be packed out of the vehicle at its delivery point,
- $(c_v(s_v, p_1) + \sum_{i=1}^{k(v)-1} c_v(p_i, p_{i+1}))/sp_v \leq l_v$ is valid for each vehicle v , which means that the route without the return trip has to be finished within the time limit,
- at each point p_i the items which are in the vehicle ($j \in S$ with $j \in in(P_r)$ and $j \in out(P_q)$ for some $r \leq i < q$) must satisfy the loading constraints (the weight limit and the 3-dimensonal packing constraint)

Then the objective function is

$$\sum_{v \in R} (c_v(s_v, p_1) + c_v(p_{k(v)}, e_v) + \sum_{i=1}^{k(v)-1} c_v(p_i, p_{i+1}))$$

3 The heuristic algorithm

After initialization we build routes on the given graph for each vehicle, considering pairs of vertices, where there are requests in between. We are repeating this step as long as we have any unassigned pair of vertices. After this step we apply a simple local search method, with which we are swapping vertices along a route as long as we can decrease the total distance of a route, while keeping the linear ordering of the vertices. After the simple local search step, comes the execution of the packing step. The algorithm used here, is what is described in [1]. Then, we repeat the steps above, as long as we have any uninitialized vehicle and at least one untransported item. The last step is an advanced local search method, with which we are changing vertices between routes, while keeping the feasibility of the packing.

Detailed description of the proposed algorithm

Example

In order to make the algorithm easier to understand, we use a simple example to demonstrate the steps of the algorithm.

In our example we have a simple graph, which consists of 7 vertices and 10 symmetrical edges. The vertices are named 1,2,...,7. The edges are as follows: 1-3, 1-4, 1-5, 2-3, 2-5, 3-4, 4-5, 4-7, 5-6, 6-7. The weight of the edges in the same order: 10, 10, 15, 15, 10, 5, 10, 5, 5, 15. We have 2 vehicles (v_1 and v_2), and 3 demands to fulfill. The vehicles are defined as follows: $s_{v_1}=1$, $e_{v_1}=7$, $s_{v_2}=2$, $e_{v_2}=7$, $sp_{v_1}=sp_{v_2}=1$, $X_{v_1}=Y_{v_1}=Z_{v_1}=W_{v_1}=X_{v_2}=Y_{v_2}=Z_{v_2}=W_{v_2}=1$, $l_{v_1}=l_{v_2}=30$. The items are defined as follows: $x_j=z_j=0.75$, $y_j=0.5$, $w_j=0.5$ for all demands. $s_1=1$, $e_1=4$, $s_2=3$, $e_2=4$, $s_3=5$, $e_3=6$. The cost of travelling on edge e : $c_v(e)$ = weight of e for both vehicles, thus it will be easier to follow the examples.

Phase 1 (Initialization)

First of all, given a graph $G(V, E)$, we create the set S (defined in previous chapter). Moreover, let S_i be the set of (a, b) couples, that are already inserted, initialized as an empty set. Let us assign a vehicle (taken from the initial set of vehicles R) to all couples in S , where $c_v(s_v, a) + c_v(b, e_v)$ is minimal.

Note: The assignment does not mean, that a certain demand can only be satisfied by v , it only means, if (a, b) is chosen as a first pair of vertices for a new route, then v is assigned to this route.

Example: $S=\{ (1, 4), (3, 4), (5, 6) \}$. The assigned vehicle will be v_1 for $(1, 4)$ and $(3, 4)$, and v_2 for $(5, 6)$.

Phase 2 (Route Building)

During this phase we repeatedly do the following steps:

Step 2.1 Choose a couple (a, b) from $S \setminus S_i$, for which $c_v(s, a, b, e)$ is minimal, where S_i denotes the set of investigated couples and $c_v(s, a, b, e) = c_v(s, a) + c_v(a, b) + c_v(b, e)$ with the vehicle v assigned to the pair (a, b) in the initialization. Let



Figure 1: Routes after Phase 2

r denote the route starting at s and continued by a and b , and ending at e . Insert (a, b) into S_i .

Step 2.2 Iteration: While the current route r can be continued with any couple from $S \setminus S_i$: Insert the couple (a, b) from $S \setminus S_i$, where total distance after inserting (a, b) into r before the position of e is minimal. Insert (a, b) into S_i . A route can be continued by a vertex, if the total time travelled (length of r from s to the last delivery point after the insertion divided by sp_v , where sp_v is the speed of v) is not greater than l_v , where l_v is the time limit for v .

Step 2.3 If vehicle v has been assigned to other couples in $S \setminus S_i$ before, repeat the vehicle assignment for those couples, allowing only unused vehicles. We have to do this reassignment since v cannot be used to serve these couples.

Step 2.4 If $S \setminus S_i$ is not empty and there is at least one unused vehicle that can be continued with any couple from $S \setminus S_i$, go to step 2.1.

Note: If there are no unused vehicles left, and no route can be continued by any item from $S \setminus S_i$, we still can not report the actual problem to be unsolvable, because the two local search methods in the later phases can still make it solvable.

Example: We start with couple $(1, 4)$, and we use v_1 , as v_1 is the assigned vehicle for couple $(1, 4)$. We can insert both $(1, 4)$ and $(3, 4)$ into the route of v_1 , at this point $c_{v_1}(r)=25$, but the total time, which is checked with l_{v_1} is 20. We can not insert the last couple $(5, 6)$ as it would increase the total time to 35, which is $> l_{v_1}$. After these steps: $S \setminus S_i = \{ (5, 6) \}$. (No reassignment of the vehicles is needed in Step 2.3, because v_1 is not assigned to $(5, 6)$). After the second iteration, the two routes are demonstrated on Figure 1.

Phase 3 (Simple Local Search)

This simple local search phase consists of many internal swaps within a single route, and no swaps are allowed between different routes. This is an essential step, if we consider that the vertices are inserted into the routes without any ordering. We repeat the following steps for route $r = p_1, \dots, p_{k(v)}$ assigned to vehicle v :

Step 3.1 Let M be a set of pairs of positions in the route, initialized as an empty set. We will use this set for memorizing the already investigated pairs of vertices.

Step 3.2 Choose a (i, j) pair of positions, where $0 \leq i < k(v)$ and $p_i \neq p_j$ and (i, j) or (j, i) is not in M . If there is no such pair of positions, go to Step 3.5.

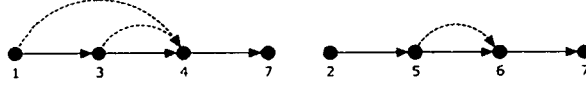


Figure 2: Routes after Phase 3

Step 3.3 Check if the swap at positions i and j would cause any of the requests, that are assigned to route r , unsatisfiable. This happens, if there is at least one request, for which, the start point does not precede the possible end points along r . If this swap would cause unfeasibility, insert (i, j) into M , go to Step 3.2. Otherwise, let us denote the new route r' , go to Step 3.4.

Step 3.4 Let $c_v(r)$ denote the travelling cost for r , using v . If $c_v(r') < c_v(r)$, let $r := r'$, and delete every occurrence of i and j from M . Doing so, we allow further swaps of p_i and p_j in their new position. Continue with Step 3.2.

Step 3.5 If $S \setminus S_i$ is not empty, try to insert a pair of vertices from $S \setminus S_i$ into r , using the same method that was described in Phase 2. If any pair from $S \setminus S_i$ was successfully inserted, go to Step 3.2, otherwise proceed to the final step of phase 3.

After the iteration the length of the route is not greater than before the iteration, and no demand has been made unsatisfiable, which had been satisfiable before.

Final step of Phase 3

We may have the same vertex on r multiple times, and we also note that we had a constraint that $c_v(a, a) = 0$, for every $v \in R$, where $a \in V$, therefore in many cases after the iteration we may have more instances of a vertex along r next to each other. In this final step, we iterate through r , and remove all surplus instances of a , so that a can not be followed by an other instance of a .

Note: We note that Phase 3 is indispensable in the algorithm. Without this phase the routes might contain several instances of a vertex and this would increase the cost a lot.

Example: Considering the route of v_1 , which is $1 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 4 \rightarrow 7$, the only swap, which is possible, does not create unfeasibility and shortens the route (thus decreasing total cost) is the swap at positions 3 and 4 ($p_3=4, p_4=3$). After processing the swap the resulting route will be $1 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 4 \rightarrow 7$, thus the length of the route decreases from 25 to 20. For v_2 we can not process any swaps, which could decrease the total length of the route and retain feasibility at the same time. During the final step the duplicate instances are deleted, resulting in the routes demonstrated on Figure 2.

Phase 4 (Packing)

During this phase we try to load as many items into the vehicle along the route as many is possible. We treat the items with priority, that are more difficult to be

packed. An item can be transported by vehicle v , if and only if it can be loaded in the loading area of the vehicle for every vertex along the route r between the item's pickup and delivery point. Hereby we note that the same vertex can be on r multiple times, so we need to check all feasible possibilities. If the item can be loaded in the loading area of v for any of these possibilities, it is considered transportable. The input for this phase is the list of vertices for route r , and a list of items, item j is determined by its size x_j, y_j, z_j , by its weight w_j and by its pickup and delivery points, s_j and e_j , where s_j and e_j are vertices along the route r . We note the loading area of v (X_v, Y_v, Z_v) and weight limit W_v , specific for v . To pack the items we use the following alteration of the Block algorithm which is defined in [1]. In this application of the Block algorithm we do not aim to pack all the items in a given set, to a minimal number of bins, but we pack the given items into one single vehicle's loading area. At each vertex in the route we use the following algorithm to pack the items.

Block algorithm

Step 4.1 (Classifying Phase) In this phase the items are divided into classes by their x -coordinates. Let $f_0 = L < f_1 < f_2 < \dots < f_k = X$ be the list of the border points let C_i be the set of the boxes having the x -coordinate between f_{i-1} and f_i . The border points are determined by algorithm *IPM* given below.

Step 4.2 (y -Block Building Phase)

Step 4.2.1 For each class C_i first order the elements by the y -coordinate.

Step 4.2.2 If the list in the class is empty then we move to the next class. Otherwise we take the longest prefix of this list, which still fits into the container and does not exceed the maximal weight of the container.

Step 4.2.3 We remove these items from the list and replace them with a single item (y -Block), which has the x -value of the greatest x -value and a z value of the greatest z -value in the list, and a height(y) of Y . Go to Step 4.2.2. using the new actual list.

Step 4.3 (z -Block Building Phase) We take only those boxes into account in each iteration, which are in the i -th interval according to its x -coordinate. These are only y -Blocks.

Step 4.3.1 We sort these boxes by their z -coordinates into decreasing order.

Step 4.3.2 We form z -Blocks from this list using first-fit strategy; we always choose the first element which fits into the container and does not exceed the maximal weight of the whole container. If no more such element exists we move to Step 4.3.3.

Step 4.3.3 We remove these y -Blocks from the list and replace them with a single item (z -Block), which has the x value of the greatest x -value, a z -value of Z , and a y -value of Y .

Step 4.4 (The packing phase) After the z -Block building phase there are blocks with size of the form (x_i, Y, Z, w_i) .

Step 4.4.1 Order the elements (z -Blocks) into decreasing order by $x_j w_j$. Go to Step 4.4.2.

Step 4.4.2 Pack the elements into the bins using a greedy strategy. This algorithm packs an element into the bin when it fits. If it does not fit then we skip the block. Note that in this case an element fits into the bin if it can be packed without overlapping and it does not violate the weight limit.

Considering the values of f_1, \dots, f_k we use the following algorithm to determine them. The algorithm puts at most K elements into each class, moreover it ensures that the sum of the elements is at least K for each consecutive pair of intervals.

Interval Preparation Method

Initialization part: Let $I = \{1, X\}$ be an ordered list of border points. Let K be the number of elements, we aim to have in each interval.

Step IPM1 Let f_i be the border point, for which the $[f_i, f_{i+1})$ interval contains the most elements. If this amount is at most K , then proceed with Step IPM2. Else, we divide this interval, with inserting a new border point into the list between f_i and f_{i+1} , with value of $(f_i + f_{i+1})/2$. Refresh the interval assignments and repeat Step 1.

Step IPM2: Let i be an index of I , for which the sum of elements in intervals $[f_i, f_{i+1})$ and $[f_{i+1}, f_{i+2})$ is minimal. If this sum is larger than K , then exit. Else concatenate these intervals by deleting border point f_{i+1} from the list I . Refresh element assignments and repeat Step IPM2.

Final step of Phase 4

If an item j is not transportable by v , and $(s_j, e_j) \in S_i$, where s_j and e_j are the pickup and delivery points for j , then delete (s_j, e_j) from S_i . This step is needed to ensure, that only those pair of vertices are prohibited from insertion into further routes, for those there are not unsatisfied demands. If $S \setminus S_i$ is not empty (means that there are untransported items), and there are unused vehicles, continue with Phase 2, Step 2.1. If there are not any unused vehicles, but there are untransported items, we still can not state, that we can not provide a feasible solution, because the local search method, described in Phase 5, can still make space for the untransported items.

Example: During this step the loading is trivial at all positions, and all items are transportable. Items 1 and 2 are packed to v_1 (picked up at positions 1 and 2), item 3 is packed to v_2 (picked up at position 2).

Phase 5 (Advanced Local Search)

The input for this phase is a set of routes, an assigned vehicle for each route, and a list of transported items for each route. During this phase, we will investigate all possible vertices, and we will try to move a vertex into an another route, keeping in mind to maintain the feasibility of the transportation.

Step 5.1 Choose a point a in route r_1 , which cannot be the start or end point of r_1 , and has not been investigated before, and choose a route r_2 . If all possibilities are investigated, go to Step 5.7.

Step 5.2 Generate the "must-move neighbourhood" of the designated point. This contains the vertices, which have to be moved to the other route together with the designated point. This "must-move neighbourhood" is a sublist of positions of a route, vertices included are: the designated point, and all points that are start points of a request fulfilled during this route, and ending at the designated point, or endpoints of such request, starting at the designated point. Let us denote this by L_a . The ordering of the points must be also kept within L_a . Note, that this could also be used recursively, generating the Kleene closure of the designated point, using it, we would not need the following step, but practically, this closure is usually close the whole route, and exchanging almost whole routes with one another would not end up in decrease of total distance.

Step 5.3 Determine the sublist L'_a of L_a . L'_a represents those positions of L_a , that can be deleted from their original route r_1 . A vertex at a position can be deleted from its previous route if there are no such demands assigned to this route, that have their start point in L_a and their endpoint not in L_a or vice versa. $L_a \setminus L'_a$ represents the set of points in the "must-move neighbourhood" of a , where the points are tied to the original route and to L_a at the same time.

Step 5.4 Insert L_a into r_2 , while keeping the linear order of points from L_a , delete L'_a from r_1 .

Step 5.5 Execute phases 2, 3 and 4 for the new r_1 and r_2 to determine, whether all the items, that were previously transportable, are still transportable. If no, discard the changes, go to Step 5.1.

Step 5.6 If the new total sum of costs has been decreased by the swaps, save the changes, else discard them. Go to Step 5.1.

Step 5.7 If the total result value has been improved during the previous run of this local search procedure, restart the procedure (go to Step 5.1), otherwise proceed to Step 5.8.

Note: It may also happen, because of the various speeds of vehicles, that the total amount of distance travelled increases, but the total amount time consumed decreases.

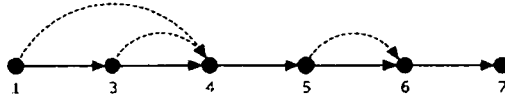


Figure 3: Solution after Phase 5

Step 5.8 If $S \setminus S_i$ is not empty, reexecute the algorithm, starting at Phase 2 Step 2.2, for each route. This reexecution places untransported items, wherever it may be possible. If there are still untransported items, report them as items not transported and a possible infeasibility of the input.

Example: Let us investigate the case, when r_1 denotes the route of v_2 and r_2 denotes the route of v_1 . We can not choose vertex 2 from r_1 , because it is a starting point, so let us choose vertex 5 (at position 2). The "must-move neighbourhood" L_a of vertex 5 will be: $\{2, 3\}$, representing vertices $\{5, 6\}$. L'_a will be set to $\{2, 3\}$, because there aren't other demands in r_1 , which could retain an instance of vertex 5 or 6 in r_1 . As we proceed to Step 5.4, we try to insert vertices 5 and 6 to r_2 , while keeping the order of them. One possibility is to insert them between positions 2 and 3 (in r_2), but this would increase the length of r_2 from 20 to 50, and the total time would be 45 (the length of the route minus the last edge, which has a weight of 5), which is $> l_{v_1}$, therefore the insertion at these points is not feasible. We refrain from presenting all infeasible possible insertions, so we continue to check insertion between positions 3 and 4. This increases the length of the route to 40, but the time consumed is only 30. (the last edge on this route is 6-7, which has a weight of 10). This can be accepted, because $l_{v_1}=30$. As we can not find any better insertion, we proceed to Step 5.5. Fortunately all demands can be fulfilled (trivial), so we can continue with Step 5.6. The new total sum of costs can be decreased from 50 to 40, so we decide to keep the changes. We note that vehicle v_2 will not transport any items, therefore it will not be included in the solution. We demonstrate the final solution in Figure 3.

4 Description of tests

To our knowledge pickup and delivery vehicle routing with 3-dimensional loading constraints has not been investigated before, therefore no test instances have been published yet. We generated our test instances, combining the following methods:

- **Graph:** For the graph we used a part of the public roadsystem of county capitals of Hungary. Below we can see the visualization of the graph, reflected on the map of Hungary:

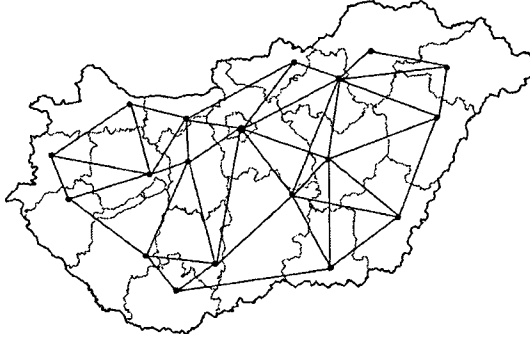


Figure 4: The graph

- **Items:** We extended the method with weights, which was used in [14] to generate the test demands (items), both uniform and gaussian (standard deviation: 1.0, the mean exactly halves the given interval) distributions were investigated. The pickup and delivery points were generated uniformly random among points from the input graph.
- **Vehicles:** Loading area generated as in [14], start- and endpoints are (uniformly) random points from the graph. Time limit is uniformly random from the 1000-2000 interval.

We were using the following testbed: Intel Core i5 750@3.15Ghz, Kingston 2x2 GB DDR3 1600Mhz, Gigabyte P55-UD3, MS Win7 x64, Java 1.6.0_23 x64

Testcases

For both distributions we performed 10 types of tests (the same intervals are used for generating the size as in [14]). We define in each test a maximal loading area for the vehicles it is $X \times Y \times Z$ and a weight limit denoted by W . In all testcases the sizes of the loading area of the vehicles are chosen as follows: 50% : $1.0 \cdot MS$, 10% : $0.9 \cdot MS$, 10% : $0.8 \cdot MS$... 10% : $0.5 \cdot MS$ respectively, where MS is the maximal possible size of the loading area.

In the first 5 types of tests $X = Y = Z = 100$ and $W = 10000$. The intervals which are used for the uniform distribution are the following:

- Type 1: $x \in [1, 1/2X]$, $y \in [2/3Y, Y]$, $z \in [2/3Z, Z]$, $w \in [1, 2/3W]$.
- Type 2: $x \in [2/3X, X]$, $y \in [1, 1/2Y]$, $z \in [2/3Z, Z]$, $w \in [1, 2/3W]$.
- Type 3: $x \in [2/3X, X]$, $y \in [2/3Y, Y]$, $z \in [1, 1/2Z]$, $w \in [1, 2/3W]$.
- Type 4: $x \in [1/2X, X]$, $y \in [1/2Y, Y]$, $z \in [1/2Z, Z]$, $w \in [1, 2/3W]$.
- Type 5: $x \in [1, 1/2X]$, $y \in [1, 1/2Y]$, $z \in [1, 1/2Z]$, $w \in [1, 2/3W]$.

We note that the loading in the case of type 4 is obvious, each vehicle can only transport one item at a time. In the next 3 tests the sizes are $X = Y = Z = 10$, $X = Y = Z = 40$ and $X = Y = Z = 100$ respectively, the weight limit is $W = 10000$. The intervals which are used for the uniform distribution are the following:

- Type 6: $x \in [1, X]$, $y \in [1, Y]$, $z \in [1, Z]$, $w \in [1, 2/3W]$.
- Type 7: $x \in [1, 35]$, $y \in [1, 35]$, $z \in [1, 35]$, $w \in [1, 2/3W]$.
- Type 8: $x \in [1, X]$, $y \in [1, Y]$, $z \in [1, Z]$, $w \in [1, 2/3W]$.

The following two testcases were added to simulate the behaviour on many small items. Size of the maximal loading area for both: $X = Y = Z = 100$, $W = 10000$.

- Type 9: $x \in [1, 1/4X]$, $y \in [1, 1/4Y]$, $z \in [1, 1/4Z]$, $w \in [1, 1/4W]$.
- Type 10: $x \in [1, 1/8X]$, $y \in [1, 1/8Y]$, $z \in [1, 1/8Z]$, $w \in [1, 1/8W]$.

In the first class of tests the maximal number of vehicles: 30 for 100 items, 275 for 1000 and 1250 for 5000 items. We note that in most cases only a portion of these vehicles were actually used. We expect more calculation time needed for the last two test cases, as in these test cases, the items are much smaller, therefore the packing is more difficult.

We generated inputs of size 100, 1000 and of size 5000. All testcases on all sizes and distributions were executed 100 times, except for the largest testcases ($N=5000$), which were executed 20 times each. We executed the algorithm on the test cases and also considered the algorithm without the last most time consuming local search phase. As far as the running time is concerned, the presented algorithms are fast, we summarize the running times in Table 1. The running times are very similar for the two investigated distributions. If we consider the algorithm without the last local search phase, it is faster, the running time is decreased with approximately 25 percent. The experienced results were very similar in case of both investigated distributions, therefore we only present here one of them.

Table 1: The average running time of the full algorithm (msec)

N	T1	T2	T3	T4	T5
100	698.40	633.19	661.95	698.58	632.85
1000	13349.9	11901.8	12598.6	18715.5	12136.2
5000	199479	182741	190315	318304	180405
N	T6	T7	T8	T9	T10
100	666.02	652.07	651.58	863.79	872.12
1000	13753.8	12602.7	12477.3	16311.1	19223.5
5000	206946	186038	182971	194910	218564

We also considered the value of the cost function for the algorithm. Table 2 and 3 contain the cost for the case where the expansion of the items is generated by uniform and gaussian distribution. The first three lines contain the results of the full algorithm, the next three lines show the results when the last local search phase is omitted.

Table 2: The average result value on gaussian distribution (km)

N	T1	T2	T3	T4	T5
100	14237.96	14238.44	14098.83	22360.46	14213.81
1000	124915	124825	124775	186269	125460
5000	583884	588904	586226	813913	583893
N	T6	T7	T8	T9	T10
100	17615.72	17632.82	17479.83;	34355.57	17655.96
1000	149285	149624	149656	287612	149873
5000	690341	693146	693911	1214585	689334
100	17490.91	14378.77	15381.34	9670.62	9522.66
1000	146063	126528	133402	87861	86065
5000	670274	594189	602684	435602	429330
100	23293.37	17775.66	19323.8	12492.3	12290.18
1000	189296	151233	161949	97865	96426
5000	828233	703710	719073	470851	462743

Table 3: The average result value on uniform distribution (km)

N	T1	T2	T3	T4	T5
100	14475.28	14356.12	14604.09	22307.21	14066.24
1000	125852	126217	126756	185120	123222
5000	574761	580987	582221	816104	571117
N	T6	T7	T8	T9	T10
100	18242.5	18247.02	18346.11;	34219.99	17889.46
1000	152161	152508	153411	287123	150197
5000	676405	681878	687960	1222230	675340
100	15807.48	14633.3	15146.15	9615.78	9398.92
1000	135009	127794	130426	88013	86410
5000	620470	581959	594788	434717	424375
100	20460.28	18418.25	19245.48	12527.19	12345.06
1000	167217	154564	159579	97967	96756
5000	742652	688937	704292	467698	457200

We saved the actual result value of each run of the local search procedure. In Figure 5 we can see the average result value in the ratio of the result value prior to the execution of the last local search phase (which is marked as 1.0). Each bar represents one run of the local search procedure for the actual testcase. We can observe, that the most improvements were made (to 67% of the starting result

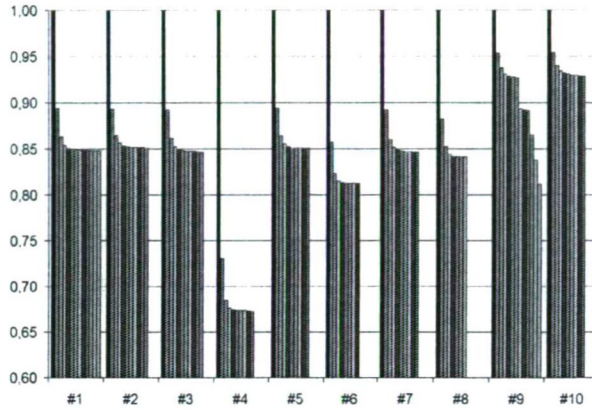


Figure 5: Percentages of result value, by each optimization cycle for the 10 testcases

value) in those testcases, when the initial result value had been worse, and the least improvement was made in testcase 10, in which the initial result value was previously the best. We also note that we can observe different number of bars for some testcases, it refers to the different number of local search runs needed for the testcase.

If we observe Table 4, we can see that the last local search has not only improved the solution value, but slightly decreased the number of required vehicles in all cases. Which can also be another reason for the decreased solution value. The first three lines contain the results of the full algorithm, the next three lines show the results when the last local search phase is omitted.

Table 4: The average number of vehicles in the solution (pcs)

N	T1	T2	T3	T4	T5
100	14.48	14.49	14.48	29.39	14.29
1000	136.8	137.29	137.04	272.85	137.70
5000	698.17	705.38	702.25	1247.5	700.77
N	T6	T7	T8	T9	T10
100	14.9	14.89	14.84;	30.00	14.75
1000	137.155	137.69	137.35	274.57	138.12
5000	698.75	705.83	702.62	1250.0	701.22
N	T6	T7	T8	T9	T10
100	19.93	14.71	15.93	9.07	9.04
1000	172.61	138.91	148.71	83.96	82.28
5000	820.1	712.87	724.83	453.14	441.62
N	T6	T7	T8	T9	T10
100	20.38	15.05	16.37	9.98	9.86
1000	173.24	139.18	149.04	85.2	84.13
5000	820.72	713.27	725.0	456.0	444.9

We also investigated the algorithm in the case where there are many items and fewer vehicles are available and it is not possible to serve all of the demands. In this case we investigated the number of the unsatisfied demands with the number of vehicles decreased to 60% of the previous tests. The results are summarized in table 5.

Table 5: The average number of unfulfilled demands (pcs)

N	T1	T2	T3	T4	T5
100	0.6	1.0	0.6	2.7	0.45
1000	0.75	2.5	3.25	28.5	1.0
5000	10.6	11.3	28.0	104.4	16.5
N	T6	T7	T8	T9	T10
100	0.45	0.45	0.95	0.0	0.0
1000	2.5	1.0	0.5	0.0	0.0
5000	27.5	4.1	8.0	0.0	0.0

5 Conclusions

In this paper we have presented a new vehicle routing model which gives an adequate description of practical problems. We presented a multi level heuristic algorithm for the solution of the problem which has a reasonable running time even for inputs of large time. Concerning the last more time consuming local search phase we could observe that it makes in average a 15 – 20% improvement in the solution given by the first part.

Acknowledgment

The authors would like to thank the anonymous referees for their helpful advice and suggestions concerning this paper.

References

- [1] T. Bartók, Cs. Imreh, Heuristic algorithms for the weight constrained 3-dimensional bin packing model, submitted for publication
- [2] M. Bárány, B. Bertók, Z. Kovács, F. Friedler, and L. T. Fan, Optimization Software for Solving Vehicle Assignment Problems to Minimize Cost and Environmental Impact of Transportation, *Chemical Engineering Transactions*, **21**, 499–504, 2010.
- [3] G. Berbeglia, J. F. Cordeau, I. Gribkovskaia, G. Laporte, Static pickup and delivery problems: a classification scheme and survey, *TOP*, **15**(1), 1–31, 2007

- [4] J.F. Cordeau, G. Laporte, M.W.P. Savelsbergh, D. Vigo, Vehicle routing, in: *Transportation, Handbooks in Operations Research and Management Science*, vol. 14, 367-417, 2007.
- [5] K.F. Doerner, G. Fuellerer, M. Gronalt, R. Hartl, M. Iori, Metaheuristics for vehicle routing problems with loading constraints, *Networks*, **49**(4), 294-307, 2007.
- [6] M. Fuellerer, K.F. Doerner, R. Hartl, M. Iori, Ant colony optimization for the two-dimensional loading vehicle routing problem, *Computers & Operations Research*, **36**, 655-673, 2009.
- [7] G. Fuellerer, K. F. Doerner, R. F. Hartl, M. Iori, Metaheuristics for vehicle routing problems with three-dimensional loading constraints *European Journal of Operational Research* **201**, 751-759, 2010.
- [8] M. Gendreau, M. Iori, G. Laporte, S. Martello, A tabu search algorithm for a routing and container loading problem, *Transportation Science*, **40**, 342-350, 2006
- [9] M. Gendreau, M. Iori, G. Laporte, S. Martello: A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints, *Networks*, **51**(1), 4-18, 2008.
- [10] M. Iori, G. Salazar, D. Vigo, An exact approach for the vehicle routing problem with twodimensional loading constraints *Transportation Science*, **41**, 253-264, 2007
- [11] M. Iori, S. Martello, Routing problems with loading constraints, *TOP* **18**(1), 4-27, 2010
- [12] A. Lodi, S. Martello, D. Vigo, Heuristic algorithms for the three-dimensional bin packing problem, *European Journal of Operational Research*, **141**, 410-420, 2002.
- [13] Malapert A, Guert C, Jussien N, Langevin A, Rousseau L-M Two-dimensional pickup and delivery routing problem with loading constraints. In: *Proceedings of the first CPAIOR workshop on bin packing and placement constraints (BPPC08)*, <http://contraintes.inria.fr/CPAIOR08/BPPC/bppc08submission10.pdf>
- [14] S. Martello, D. Pisinger, D. Vigo, The Three-Dimensional Bin Packing Problem, *Operations Research*, **48**(2), 256-267, 2000.
- [15] C.D. Tarantilis, E.E. Zachariadis, C.T. Kiranoudis, A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem *IEEE Trans Intell Transp Syst*, **10**, 255-271, 2009
- [16] P. Toth, D. Vigo, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

- [17] E.E. Zachariadis, C.D. Tarantilis, C.T. Kiranoudis, A guided tabu search for the vehicle routing problem with two-dimensional loading constraints, *European Journal of Operational Research*, **195**, 729-743, 2009.
- [18] F. Wang, Y. Tao, N. Shi, A survey on vehicle routing problem with loading constraints. *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, **2**, 602-606, 2009

Dynamic Communities and their Detection

András Bóta*, Miklós Krész[†] and András Pluhár[‡]

Abstract

Overlapping community detection has already become an interesting problem in data mining and also a useful technique in applications. This underlines the importance of following the lifetime of communities in real graphs. Palla et al. developed a promising method, and analyzed community evolution on two large databases [23]. We have followed their footsteps in analyzing large real-world databases and found, that the framework they use to describe the dynamics of communities is insufficient for our data. The method used by Palla et al. is also dependent on a very special community detection algorithm, the clique percolation method, and on its monotonic nature. In this paper we propose an extension of the basic community events described in [23] and a method capable of handling communities found a non-monotonic community detection algorithm. We also report on findings that came from the tests on real social graphs.

Keywords: graph mining, network analysis, community, community detection, dynamic communities

1 Introduction

The analysis of adaptive networks is considered to be a traditional research field, which in recent years, has received a new impulse, thanks to the variety of available test databases [3, 20]. These graphs are so large in some cases, that only the fastest, near linear time algorithms have a chance of tackling the given tasks. One of the approaches to network analysis, community detection has received a lot of attention both from the point of theory and applications [1, 6, 8, 9, 10, 12, 15, 19, 22, 24, 25]. The definition of communities centers around dense subgraphs of the network. In traditional community detection, we are looking for disjoint subsets of vertices, that are connected to each other more closely, that to the rest of the graph.¹ In overlapping community detection, the subsets are not disjoint. Hereafter by the

*University of Szeged, E-mail: bandras@inf.u-szeged.hu

[†]University of Szeged, E-mail: kresz@jgypk.u-szeged.hu

[‡]University of Szeged, E-mail: pluhar@inf.u-szeged.hu

¹Note that non-overlapping communities, or *clusters* had been researched even earlier. The clustering methods have huge literature, we refer to those notions only when they are significant in the case of overlapping communities.

notion of communities we will mean overlapping communities, otherwise we will use either clusters or non-overlapping communities. We will also take several basic definitions from graph theory [4].

However, communities are not static, they can change or even disappear in time; this might have caused the split of a club in the famous experiment of Zachary [27].

The dynamics of these networks is usually represented as a series of graphs $\{G_i\}_{i \in \mathcal{T}}$, where $\mathcal{T} = \{1, \dots, \ell\}$ represents a discrete time set. The number of time instances usually depends on two things: the length of the period, in which we observe the network, and the resolution, which governs, that in a given time period, how many “snapshots” do we take from the network. Although one might try to consider the whole history of the graph series in one step, for large graphs this is not feasible. Both Asur et al. [2] and Palla et al. [23] have chosen the path of following the entities from G_i to G_{i+1} , and reconstructing the whole process out of these. Note that Asur et al. deals with clusters, while Palla et al. follows the lifetime of (overlapping) communities. They both took a normative approach to communities: they have decided upon the possible events that might happen to a community. In other words, they have simply listed the basic events of a hypothetical classification. We reproduce some of the relevant work of Palla et al. as follows.

1.1 Basic events

The basic events described in [23] are the following:

- Birth, when a new community emerges without predecessor.
- Death, when a community disappears without successor.
- Merging, when several communities join together to form a new community.
- Splitting, when a community splits into several new communities.
- Growth, when a community gains new members.
- Contraction, when a community loses members.

Assuming the above described events, the problem is reduced to the following. Given the graphs G_1 and G_2 , describing the starting and destination graphs, compute the set of communities in both, let those be \mathcal{K}_1 and \mathcal{K}_2 . Then find a relation \mathcal{R} on $\mathcal{K}_1 \times \mathcal{K}_2$ in an “obvious way”; if a $C_1 \in \mathcal{K}_1$ is in no relation, then it is a *death*. If $C_1 \in \mathcal{K}_1$ is in relation with $C_1^2, \dots, C_\ell^2 \in \mathcal{K}_2$ and $C_i^2 \subset C_1$ for $i \in [1, \dots, \ell]$, then C_1 *splits into* C_1^2, \dots, C_ℓ^2 . Similarly, if $C_2 \in \mathcal{K}_2$ is in no relation, then it is a *birth*. If $C_2 \in \mathcal{K}_2$ is in relation with $C_1^1, \dots, C_\ell^1 \in \mathcal{K}_1$ and $C_i^1 \subset C_2$ for $i \in [1, \dots, \ell]$, then C_1^1, \dots, C_ℓ^1 *merged into* C_2 . Finally a related pair (C_1, C_2) , $C_i \in \mathcal{K}_i$ for $i \in \{1, 2\}$, should mean a *growth (contraction)* if $C_1 \subset C_2$ ($C_1 \supset C_2$).

1.2 Implementation

Computing the relation \mathcal{R} , assuming that the basic events cover all possible cases, is not hard theoretically. In practice this is different, since for large graphs a naive approach of considering all pairs $(C_1, C_2) \in \mathcal{K}_1 \times \mathcal{K}_2$ is too costly. The way out of this quadratic complexity is an idea due to I. Derényi [23]. They assume that the communities are always given by the Clique percolation method (CPM) [1, 22], which is *monotonic*. More exactly, if H and G are graphs on the same vertex set, $E(H) \subset E(G)$ and $C \in \mathcal{K}_H$ then there exists a $C' \in \mathcal{K}_U$, such that $C \subset C'$, where \mathcal{K}_H and \mathcal{K}_G are the sets of communities of H and G , respectively.

Then the *union graph* U is formed such that $V(U) := V(G_1) \cup V(G_2)$, $E(U) := E(G_1) \cup E(G_2)$ and \mathcal{K}_U is computed. Now instead of going through all (C_1, C_2) pairs from $\mathcal{K}_1 \times \mathcal{K}_2$, one needs to check only those pairs for which there is a $C' \in \mathcal{K}_U$, such that $(C_1, C_2) \subset C'$.

1.3 Results of Palla et al.

By applying this method, they get a convincing results on the dynamics of communities. The most significant results are: the larger a community the older it is. The expected lifetime of a community increases with its size. A small community is more stable if it does not change its members, while it is the opposite for large communities. Let us note that the methodology is crucial, for the definitions, time scale and database the reader should consult the paper [23].

2 Problems

We have conducted a similar research on two large social networks described in the results chapter, meaning we analyzed the changes in the community structure of the corresponding networks. We tried to adapt the methodology of [23] with little success. To find communities, we used CFinder² for the CPM, resulting in the computational issues described in subsection 2.2. Then we decided on using the implementation of the N^{++} method³ developed in [9], which was finally able to handle our networks. We have also found, that the framework they have used is insufficient to describe the community dynamics in our database. One of the reasons for this could be the difference between our database and the one they have used. This will be discussed in the results section. Another reason might be the difference between the community detection algorithms. This will be discussed in section 2.3.

2.1 Basic events

In our experience, the description of basic events is not complete. The deviations from the description have at least three main causes. (i) The time scale is too large,

²The software was downloaded from the page <http://angel.elte.hu/cfinder/>

³We would like to express our thanks for obtaining free access to the appropriate softwares.

and G_1 differs significantly from G_2 . This is unavoidable, since in several cases the measurement intervals are given. (ii) A community might become extinct not by splitting, but by leaving behind a number of overlapping communities on the same vertex set⁴; we will discuss this in depth in subsection 3.1. (iii) In dense real graphs, it happens frequently that a set of overlapping communities change into another set of communities, and the relations cannot be easily mapped.

2.2 Computational issues

For large dense graphs the CPM is time inefficient. Deciding the proper value of the parameter k is also problematic. The other problem is, that for some benchmark graphs, e.g. the Zachary, the CPM performs poorly. It is natural to try out other community detection methods, since those might give better predictions in applications [8, 10, 19].

2.3 Implementation

The method relies on the monotonicity of CPM. In general, communities do not behave this way, they might split when adding edges to a graph, or merge when deleting edges. This phenomenon makes it harder to map the communities \mathcal{K}_1 and \mathcal{K}_2 .

3 Solutions

To motivate our solution, we analyze a small problem in depth. It shows that the introduction of new classes for community events are unavoidable.

3.1 Motivation

The most obvious way to deal with the problem mentioned in subsection 2.1 (i) is an artificial refinement of the time scale. That is, we fix the list of the changes that happened between G_i and G_{i+1} , and insert new graphs into the series, such that each new graph differs from the previous one in only one item. However, we will see that changing this order changes both the number and types of appearing community events, which implies, that the artificial refinement of the time scale should be avoided in practice. More importantly, this example will show us, in correspondence with problem (ii), that the seven basic events defined above are inadequate when dealing with differences larger than one.

We illustrate the above mentioned problems with Zachary's karate club network [27]. Five edges were removed from the network one by one, in two different se-

⁴Let A, B, C and D be cliques. Add a few edges between among those, such that $A \cap B$ and $C \cup D$ are the two resulting communities. Deleting two and adding two edges, $A \cap C$ and $B \cap D$ will be the new communities, which does not fit in the scheme.

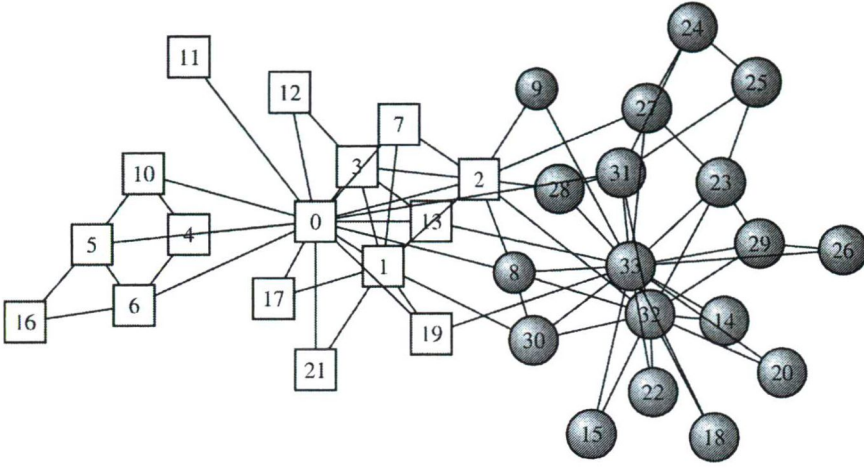


Figure 1: Zachary's karate club network. The boxes and naughts indicate the vertices corresponding to the split that has appeared during the original experiment.

Original	a	b	c	d	Final
7	7	7	7	7	13
9	9	9	9	9	13
10	10	10	10_a	10_a	10_a
			10_b	10_{ba}	13
				10_{bb}	13

Table 1: The changes in the community structure with the first edge removal sequence. Communities 1 through 6, 8, 11 and 12 do not change, and are omitted.

quences⁵. The N^{++} algorithm was used for the purpose of detecting communities; the important parts of the outputs are summarized in Tables 1 and 2.

In the first sequence, edges a, b, c, d and e are removed in this order. After removing the first two edges, nothing changes. In the next step however, community 10 containing the nodes $\{0, 1, 2, 3, 7, 13, 19\}$ splits into communities $10_a : \{0, 1, 3, 7\}$ and $10_b : \{0, 1, 2, 13, 19\}$. In the next step, 10_b splits further into $10_{ba} : \{1, 2, 19\}$ and $10_{bb} : \{0, 1, 2, 13\}$. After removing the fifth edge from the network, several communities merge together, namely $7 : \{0, 1, 17, 21\}$, $9 : \{0, 2, 8, 32\}$, $10_{ba} : \{1, 2, 19\}$ and $10_{bb} : \{0, 1, 2, 13\}$. Out of these a new community, 13, is born: $\{0, 1, 2, 8, 13, 17, 19, 21, 32\}$.

Thus far, the basic community events defined in [23] were sufficient for the description. What happens however, if we compare the communities from the original graph with the communities of the final graph? Ignoring those communities

⁵The edges $a = (0, 13)$, $b = (0, 19)$, $c = (2, 3)$, $d = (2, 7)$ and $e = (3, 19)$.

Original	c	a	e	d	Final
7	7	7	7	7	13
9	9	9	9	9	13
10	10	10	10 _a	10 _c	13
			10 _b	10 _d	10 _e

Table 2: Changes of the communities in the second sequence. Communities 1 through 6, 8, 11 and 12 do not change, and are omitted.

that did not change, we have number 7, 9 and 10 in the original graph, and 10_a and 13 in the modified graph. It would seem that community 10 is involved in a contraction event resulting in 10_a. Communities 7 and 9 merge into the new community 13, but 13 also acquires some nodes from community 10, meaning it is also a growth event. This gets further complicated considering the results of the one by one edge removal: It seems that the first event, that took place was a split event. One way to solve this problem is to extend the number of community events used by the algorithm with more complex events that involve multiple basic ones. Before deciding which combinations should be used, let us examine the second edge removal sequence.

In the second sequence, edges c, a, e, d and b are removed in this order. Despite the different sequence, the first few steps are the same until community 10 splits apart. The resulting communities are different however, with 10_a being $\{1, 2, 13\}$ and 10_b : $\{0, 1, 2, 3, 7, 19\}$. In the next step, 10_a is involved in a growth event, stealing a node from 10_b, which is in a contraction event, resulting in 10_c : $\{1, 2, 13, 19\}$ and 10_d : $\{0, 1, 2, 3, 7\}$. It is obvious that the two events are connected, but the original framework does not allow such complex situations.

In the final step, community 10_d loses another node resulting in 10_e : $\{0, 1, 3, 7\}$ which corresponds to 10_a from the previous example. A merge event occurs at the same time, joining communities number 7, 9 and 10_c. It is important to note, that 10_c is not identical to any community in the previous example, not even 10_b.

This example has two important consequences. First, if one tries to refine the time scale by creating an artificial series of graphs by adding or removing edges one by one, the result depends on the order of edges, rendering this approach meaningless. In the rest of the paper, we will not use this naive idea for solving the problem at hand.

The second consequence is closely tied to the first one. It appears from the example above, that the basic events used to describe the changes of communities are satisfactory only if we change the graph one edge at a time. In all the other cases, more complex events, combinations of the basic events, are required.

3.2 Extending the number of community events

Using all possible combinations of the basic events is not necessary, as it would over-complicate the results of the algorithm. The first four events listed below are straightforward, each one is a combination of two basic events. The obscure event represents combinations of possibly more than two events, including merge-split, merge-split-merge, grow-split-merge, and so on. This means, that this event could be divided further into other events, but using the test data available we have found, that these five additional events are sufficient for describing community dynamics. With them, the changes in community structure can be identified reasonably well without unnecessary complications of the algorithm.

Tables 3 and 4 list the number of different community events found. A more detailed analysis of the results will be given in the results section. In the first one, the number of newly introduced merge and split variants are more numerous than their original counterparts. The number of obscure cases is somewhat higher, but still has the same magnitude. In table 4 the newly introduced events are far less numerous, but still relevant. This justifies the introduction of these events, and also implies, that further dividing the obscure cases would result in categories containing a very few number of events.

- Grow-merge, several communities join together, and also absorb some additional members.
- Contraction-merge, several communities join together, but loose some members in the process.
- Grow-split, a community splits into several communities, but these communities absorb additional members.
- Contraction-split, a community splits into several communities, and these communities also loose members.
- Obscure case. Multiple communities are involved from both \mathcal{K}_1 and \mathcal{K}_2 , with their members reordering.

With respect to the above framework, we redefine the concepts of the split and merge events. Given the community sets \mathcal{K}_1 and \mathcal{K}_2 , and a community $C_1 \in \mathcal{K}_1$ which is in relation with communities $C_1^2, \dots, C_\ell^2 \in \mathcal{K}_2$, we define the split event if $|C_1| = |C_1^2 \cup C_2^2 \cup \dots \cup C_\ell^2|$. If $|C_1| < |C_1^2 \cup C_2^2 \cup \dots \cup C_\ell^2|$, then we define the grow-split event, and finally if $|C_1| > |C_1^2 \cup C_2^2 \cup \dots \cup C_\ell^2|$, we define the contraction-split event.

The definition of the merge event is symmetrical: given the community sets \mathcal{K}_1 and \mathcal{K}_2 , and communities $C_1^1, \dots, C_\ell^1 \in \mathcal{K}_1$ which are in relation with a community $C_2 \in \mathcal{K}_2$, if $|C_1^1 \cup C_2^1 \cup \dots \cup C_\ell^1| = |C_2|$, then we define the merge event. If $|C_1^1 \cup C_2^1 \cup \dots \cup C_\ell^1| < |C_2|$, then we define the grow-merge event, and finally if $|C_1^1 \cup C_2^1 \cup \dots \cup C_\ell^1| > |C_2|$, we define the contraction-merge event.

The obscure case is less strictly defined. If there are multiple communities from \mathcal{K}_1 and \mathcal{K}_2 connected by a relation, without further analysis, we classify this relation as an obscure event.

4 Details of the algorithm

The method described in this section follows the idea of I. Derényi [23], with a few important modifications. We will adopt the concept of the union graph U , and use it to solve the originally quadratic problem described in subsection 1.2 in almost the same way as in [23]. Our additional work centers around two important modifications of the original method.

The first one is abandoning the requirement of monotonicity. This means, that we can no longer assign only one community from \mathcal{K}_U to any community from the original graphs. The solution of this problem is described in the subsection 4.4.

The second modification incorporates the detection of the extended community events. The solution to this is fairly straightforward, and will be discussed in subsection 4.5.

We will also describe an optional modification of the original idea of [23]. Instead of using U , we will use I , the *intersection* graph of G_1 and G_2 . That is $V(I) := V(G_1) \cap V(G_2)$, $E(I) := E(G_1) \cap E(G_2)$, and \mathcal{K}_I represents the communities of the intersection graph I .⁶ We will describe this approach in 4.3.

4.1 Overview

The input of the algorithm consists of three community sets $\mathcal{K}_1, \mathcal{K}_2$ and \mathcal{K}_U , where \mathcal{K}_U is the community set of U .

The output of the algorithm is a relation \mathcal{R} on $\mathcal{K}_1 \times \mathcal{K}_2$, corresponding to community events described in [23] and subsection 3.2.

The algorithm can be divided into two phases. The first phase creates a relation \mathcal{R}_1 on $\mathcal{K}_1 \times \mathcal{K}_U$ and \mathcal{R}_2 on $\mathcal{K}_2 \times \mathcal{K}_U$. The second phase combines \mathcal{R}_1 and \mathcal{R}_2 into \mathcal{R} . Because of the non-monotonic nature of the community detection algorithm, a preprocessing step is required before executing the second phase.

4.2 First phase

In the first phase we search for relations among $C_i^1 \in \mathcal{K}_1$ for all i and $C_\ell^u \in \mathcal{K}_U$ for all ℓ . We are looking for two types of relations. The first type is the *exact match*: $C_i^1 = C_\ell^u$. The second is a *contain match*: $C_i^1 \subset C_\ell^u$. Because of the non-monotonic nature of the community detection algorithm $C_\ell^u \subset C_i^1$ might also occur. This case also counts as a contain match.

For the purpose of finding these relations, we iterate over the elements of \mathcal{K}_1 and compare each C_i^1 to every element of \mathcal{K}_U . If we find a contain match, we create

⁶The community sets can be created by any, possibly non-monotonic, community detection algorithm.

a relation $r_1(C_i^1, C_\ell^u)$. If we find an exact match, we also create $r_1(C_i^1, C_\ell^u)$, but we ignore C_ℓ^u in the subsequent computations. This step is repeated for \mathcal{K}_2 and \mathcal{K}_U .

4.3 Intersection approach

Since a community detection algorithm is not necessarily monotonic, there might be elements of \mathcal{K}_1 or \mathcal{K}_2 that are not in relation with any element of \mathcal{K}_U , these were counted as “deaths” before. The intersection approach tries to solve this problem by replacing \mathcal{K}_U with \mathcal{K}_I . The only difference in the implementation is, that in the first phase \mathcal{K}_1 (and \mathcal{K}_2 later) is replaced with \mathcal{K}_I , and \mathcal{K}_U is replaced with \mathcal{K}_1 (and \mathcal{K}_2 later). After the first phase has finished, the algorithm continues after inverting the relations r_1 and r_2 .

The intersection method provides almost the same results as the union approach with a few exceptions, that will be noted in the results chapter. The size of the intersection graph I is smaller than or equal (in the special case when $G_1 = G_2$) to the size of the union graph U . From the computational point of view, this implies that the running time of the community detection algorithm should be lower on I . Other than this, the use of the intersection approach is completely optional.

4.4 Preprocessing

As we noted before, there may be more than one element of \mathcal{K}_U , that is in relation r_1 with a given C_i^1 . The same holds for C_j^2 and the relation r_2 . To solve this, we put a new, fictitious community C_a^u to \mathcal{K}_u , set $r_1(C_i^1, C_a^u)$, and delete all former relations containing C_i^1 . The same is done for C_j^2 . If C_i^1 and C_j^2 were in relation with the same elements of \mathcal{K}_u , then the same C_a^u is used.

4.5 Second phase

In this phase we run through the elements of \mathcal{K}_u . For each element $C_\ell^u \in \mathcal{K}_u$, let the elements $\mathcal{C}^1 := \{C_1^1, \dots, C_i^1\} \subset \mathcal{K}_1$ and $\mathcal{C}^2 := \{C_1^2, \dots, C_j^2\} \subset \mathcal{K}_2$ be in relation with C_ℓ^u according to r_1 and r_2 respectively. Let $\cup \mathcal{H}$ be the $\cup_{H \in \mathcal{H}} H$ for any set \mathcal{H} .

Depending on i and j , and the sizes of the communities involved, we create the relation $r(C_{i'}^1, C_{j'}^2)$ for every i' and j' , and we assign community events to these relations.

- If $\mathcal{C}^1 = \emptyset$ and $|\mathcal{C}^2| > 0$, then $r(\emptyset, C_{j'}^2)$ is a *birth* event for every j' .
- If $|\mathcal{C}^1| > 0$ and $\mathcal{C}^2 = \emptyset$, then $r(C_{i'}^1, \emptyset)$ is a *death* event for every i' .
- If $|\mathcal{C}^1| = 1$ and $|\mathcal{C}^2| = 1$, that is $\mathcal{C}^1 = \{C_1^1\}$ and $\mathcal{C}^2 = \{C_1^2\}$, then
 - If $|\mathcal{C}_1^1| = |\mathcal{C}_1^2|$, $r(C_1^1, C_1^2)$ is an *exact match*.
 - If $|\mathcal{C}_1^1| > |\mathcal{C}_1^2|$, $r(C_1^1, C_1^2)$ is a *contraction* event.
 - If $|\mathcal{C}_1^1| < |\mathcal{C}_1^2|$, $r(C_1^1, C_1^2)$ is a *growth* event.

- If $|\mathcal{C}^1| = 1$ and $|\mathcal{C}^2| > 1$, then
 - If $|\mathcal{C}_i^1| = |\cup \mathcal{C}^2|$, $r(\mathcal{C}_i^1, \mathcal{C}_{j'}^2)$ is a *split* event for every j' .
 - If $|\mathcal{C}_i^1| < |\cup \mathcal{C}^2|$, $r(\mathcal{C}_i^1, \mathcal{C}_{j'}^2)$ is a *grow-split* event for every j' .
 - If $|\mathcal{C}_i^1| > |\cup \mathcal{C}^2|$, $r(\mathcal{C}_i^1, \mathcal{C}_{j'}^2)$ is a *contraction-split* event for every j' .
- If $|\mathcal{C}^1| > 1$ and $|\mathcal{C}^2| = 1$, then
 - If $|\cup \mathcal{C}^1| = |\mathcal{C}_j^2|$, $r(\mathcal{C}_{i'}^1, \mathcal{C}_j^2)$ is a *merge* event for every i' .
 - If $|\cup \mathcal{C}^1| < |\mathcal{C}_j^2|$, $r(\mathcal{C}_{i'}^1, \mathcal{C}_j^2)$ is a *grow-merge* event for every i' .
 - If $|\cup \mathcal{C}^1| > |\mathcal{C}_j^2|$, $r(\mathcal{C}_{i'}^1, \mathcal{C}_j^2)$ is a *contraction-merge* event for every i' .
- If $|\mathcal{C}^1| > 1$ and $|\mathcal{C}^2| > 1$ then we introduce an obscure event $r(\mathcal{C}_{i'}^1, \mathcal{C}_{j'}^2)$ for every i' and j' .

4.6 Time complexity

The time complexity of the algorithm will be addressed both from the theoretical and empirical point of view. In the first phase, we compare each community from \mathcal{K}_1 to communities from \mathcal{K}_U . In the worst case, no exact matches are found resulting in an $O(n * m)$ complexity, where n is the size of $|\mathcal{K}_1|$ and m is the size of $|\mathcal{K}_U|$. An exact match always reduces the number of further computations. Therefore when we find an exact match $r_1(\mathcal{C}^1, \mathcal{C}^u)$ ($r_2(\mathcal{C}^2, \mathcal{C}^u)$), we can ignore all other relations involving \mathcal{C}^1 , \mathcal{C}^2 and \mathcal{C}^u .

The obscure events are catchier. Note, that the sizes of \mathcal{K} 's might be exponential in $|V(G_1)|$, while a complex event could involve any number of communities on both sides, which would result in doubly-exponential running time. Fortunately, usually we have less communities than vertices, and the number of obscure events are small, consisting of only a few sets. So in practice the obscure events have only a negligible effect on the (actual) running time.

In the second phase, if each element from \mathcal{K}_1 and \mathcal{K}_2 were connected with every element from \mathcal{K}_U , we would obtain a worst case complexity of $O(n * m * k)$, $k = |\mathcal{K}_2|$. In practice, a community from \mathcal{K}_U corresponds to only a few communities from \mathcal{K}_1 and \mathcal{K}_2 . This reduces the actual running time of this phase to $O(c * m)$, where c is a small constant.

5 Results

To test our algorithm, besides the small examples like Zachary's graph, we have used two large test databases. One came from an international bank [8], while the other one is a large social network. The bank graph is based on a transaction database, and consists of roughly 80000 nodes and 270000 edges, and we were provided with three time instances taken in a six month period. The edges of the

Events	12u	23u	13u
Deaths	6586	6481	9901
Births	6729	6477	9971
Unchanged	4888	5062	2529
Growths	2185	2084	1629
Contractions	1985	2123	1569
Splits	83	91	45
Grow-splits	154	133	172
Cont-splits	144	149	111
Merges	200	184	117
Grow-merges	356	315	256
Cont-merges	350	326	418
Obscure	531	595	594

Table 3: Results on the bank database. The columns show the community events in the following order: first and the second time instances, the second and third time instances, finally the first and the third instances.

social graph were also determined as the output of a data mining project, and the graph contains roughly one million nodes and 1.5 million edges. Here four time instances were taken in a four month period. Due to the size and structure of these networks, the CFinder fails to provide communities, so the communities were listed by the N^{++} method only.

5.1 Observations

Table 3 shows the results for the bank dataset for different time intervals. Notice, that the number of death and birth events are very high, about 40% of the communities die. A possible explanations for this is, that the time lapse between the instances is relatively long (three months), and in the last case, where the first and last instances are compared, this is extended to six months. During this long time period, the community structure of a network changes significantly.

Another explanation might be based on problem (iii), referred in 2.1. If a dense community changes significantly, it is hard to know what had really happened to it, while the algorithm classifies it as a death event.

The number of unchanged, growing and contracting communities has the same magnitude, which dominates the splitting and merging events.⁷

It indicates a certain dynamic equilibrium that the number of growth and contract events, and the death and birth events are balanced. The number of pure split/merge events are less than the newly introduced grow/contraction split/merge

⁷Since the first types of events involve the exact match events described in 4.2, this explains the low running time.

Events	12u	23u	34u	14u
Deaths	6847	7812	7931	23519
Births	6112	4119	3782	14934
Unchanged	59985	59247	56031	38272
Growths	2999	2161	1963	4343
Contractions	3468	3629	3458	6753
Splits	457	454	383	538
Grow-splits	83	55	37	32
Cont-splits	115	83	102	266
Merges	919	820	785	1030
Grow-merges	232	180	178	438
Cont-merges	142	137	93	79
Obscure	90	92	59	66

Table 4: Results for the social network database. The columns show the community events in the following order: first and the second time instances, the second and third time instances, the third and fourth time instances, and finally the first and the fourth instances.

events. This justifies the introduction of these events, and underlines the complications in community dynamics.

The running time of our community matching algorithm was 8 seconds, while the search for communities took 5 minutes. The used computer configuration was a machine of two cores, with each processor running at 2.0 GHz and supplied with 2 gigabytes of memory.

Table 4 shows the matching results on the social network. In contrast to our previous results, the number of deaths and births are significantly lower, and the number of unchanged communities dominates all other events. The lapse between the time instances is short (one month), but in the last case, the first and last instances are compared resulting in a four months interval. Even in this case, the number of unchanged communities is much larger than the number of deaths, so one concludes that the social graph is more stable than the graph of the bank dataset. It is important to note, that the number of birth events are significantly lower than the number of death events, and the number of split-like events do not balance this by generating more communities. This indicates that the number of communities of the network is decreasing. Indeed, the network has lost around 12.5 percent of its communities in the observed period.

As in the previous case, the number of growth and death events are balanced. Here the number of pure split and merge events outnumber the newly introduced grow/contraction and split/merge events. This also points to the fact, that this network is more stable than the previous.

Even though this network is larger than the first, the running time of the matching algorithm is about the same due to the very high number of unchanged commu-

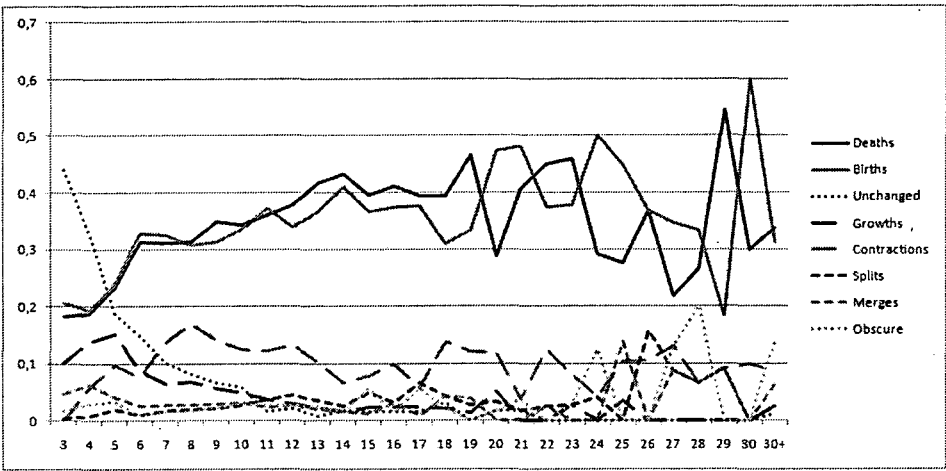


Figure 2: The percentage of community events compared to the community sizes for the bank dataset. The results were generated with the union graph approach.

nities. Note that the N^{++} needed about 60 seconds for finding the communities, our community matching algorithm took 14 seconds. We have used the same machine as before.

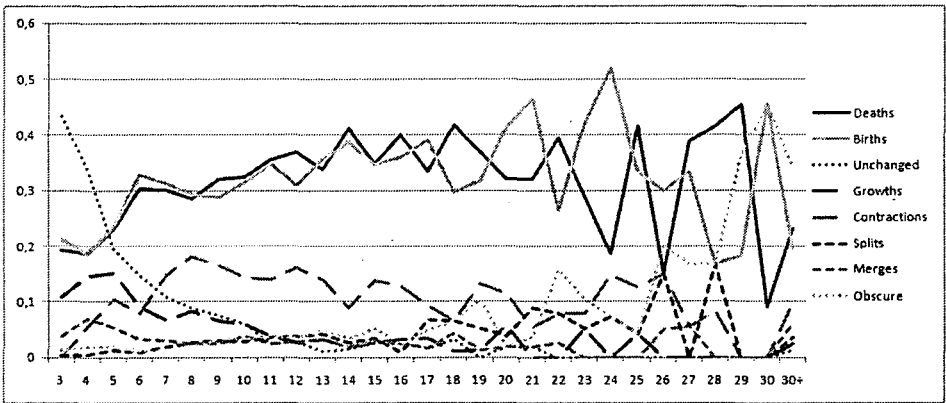


Figure 3: The percentage of community events compared to the community sizes for the bank dataset. The results were generated with the intersection graph approach.

5.2 Community evolution

Following the footsteps of Palla et al. [23], we compared the sizes and lifetimes of communities. Figure 2 displays our findings on the bank dataset. As we have concluded in the previous section, the community structure of this dataset changes very rapidly. For almost all sizes of communities the number of deaths outweigh all other community events. The only exceptions are very small communities of sizes three and four. It should also be noted, that there is a small spike of the obscure cases for the large communities. Aside from these, it can be said, that the size of a community does not relate to the community event it is involved in. Most importantly, the findings do not confirm the results of [23] that the expected lifetime of a community is a monotone function of its size. For a decisive result, further studies are needed.

If we take the intersection graph approach displayed on Figure 3, the results are almost the same, except for the largest communities. These cases are classified as obscure cases in contrast to declaring them dead as before.

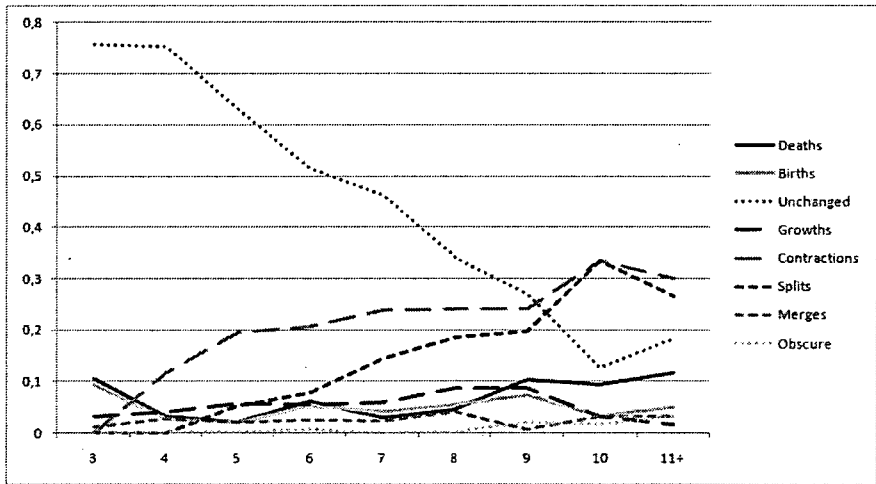


Figure 4: The percentage of community events compared to the community sizes for the social graph dataset with one month difference. The results were generated with the union graph approach.

The social network shows very different behavior. Figure 4 shows results for a one month time lapse. The percentage of death events is very small and constant compared to the community sizes, which is in contrast to the previous dataset. The number of unchanged communities is very high for small sizes, and decreases monotonically. This behavior was also reported in [23].

For large communities the contraction and split events dominate. The number of these events increases monotonically. The number of growth and merge events is small and independent of the sizes, which is somewhat surprising considering their

symmetrical counterparts. The number of all other events is small and independent of the community sizes. These results also indicate a more stable nature for this dataset.

Figure 5 shows the results for the four month interval. It is clear that in a four month interval, the community structure of a network can change dramatically, yet the difference between the results for the bank dataset and this one is clear. The number of deaths is significantly lower, but still relevant, and except for the smallest communities, independent of size. The number of birth events is lower than the number of death events, this matches with our observations in the previous chapter. The number of unchanged events is high for small communities, and it decreases somewhat faster, than in the previous network. The number of split events is increasing much faster than before, and contraction events are dominant even for medium sized communities. It should also be noted that small communities aside, the number of contraction events is independent from the community sizes. The number of all the other cases is small and constant.

The findings above reveal a strange behavior: while this network is more stable than the banking network, in some sense it is steadily losing communities. The banking network on the other hand changes more dynamically, but it is in a state of equilibrium.

For the social network, the intersection method provides almost the same results, with the percentage of death events being slightly lower.

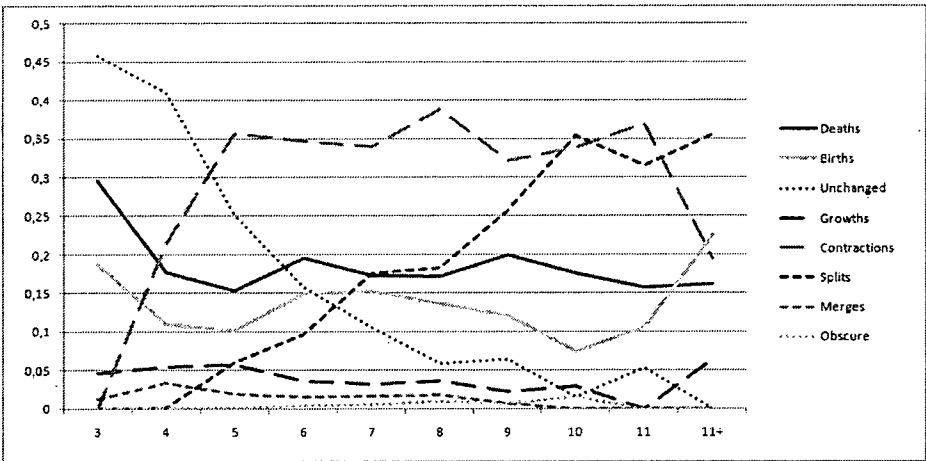


Figure 5: The percentage of community events compared to the community sizes for the social graph dataset with four months difference. The results were generated with the union graph approach.

6 Conclusion

Based on the earlier results of [23, 2] we have developed a new algorithm and methodology for following the life cycle of communities in dynamic graphs. The methodology successfully extends the incomplete notions used in [23], and results in an algorithm that works based on general community detection methods. The algorithm is very fast, it solves large community matching problems in seconds.

We have worked with two large datasets with fixed observation periods. Our findings indicate, that there is a significant difference between the behavior of the community structures and dynamics of these datasets. One of the networks is more stable, but loses communities steadily, the other network is more dynamical, but maintains its community number. We have also examined the changes in community structure in relation with the sizes of the involved communities. Our findings confirm some of the claims of [23], however not all of them.

Acknowledgment

The authors were supported by the Project named “TÁMOP-4.2.1/B-09/1/KONV-2010-0005 - Creating the Center of Excellence at the University of Szeged” also supported by the European Union and co-financed by the European Regional Fund.

The second author was partially supported by HSC-DAAD Hungarian-German Research Exchange Program (project P-MÖB/837) and Gyula Juhász Faculty of Education, University of Szeged (project CS-001/2010).

The third author was partially supported by the grant OTKA K76099 and by the grant TÁMOP-4.2.2/08/1/2008-0008.

References

- [1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, T. Vicsek CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics* **22**, 1021–1023 (2006).
- [2] S. Asur, S. Parthasarathy and D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data* Volume **3**, Issue 4, November 2009.
- [3] R. Albert, A. L. Barabási Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74** 2002.
- [4] B. Bollobás, *Modern Graph Theory*, Springer, New York (1998).
- [5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, Fast unfolding of community hierarchies in large networks. arXiv (2008): 0803.0476.
- [6] B. Bollobás and O. Riordan, Clique percolation. *Random Structures Algorithms* **35** (2009), no. 3, 294–322.

- [7] D. Braha, Y. Bar-Yam, Time-Dependent Complex Networks: Dynamic Centrality, Dynamic Motifs, and Cycles of Social Interaction, Adaptive Networks, chapter 3, 2009.
- [8] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár and T. Tóth, Parameter optimization of infection models. *Proc. of the Challenges for Analysis of the Economy, the Business, and Social Progress International Scientific Conference*, Szeged, November 19–21, 617–623, 2009.
- [9] L. Csizmadia, Recognizing communities in social graphs, MSc thesis, University of Szeged, 2003. (in Hungarian)
- [10] L. Csizmadia, A. Bóta and A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 13th International Multiconference INFORMATION SOCIETY - IS* 2010, 393–396.
- [11] T. S. Evans and R. Lambiote, Edge Partitions and Overlapping Communities in Complex Networks. arXiv:0912.4389v1, 2009.
- [12] S. Fortunato, Community Detection in graphs. *Physics Reports* Volume 486, Issues 3–5, February 2010, Pages 75–174.
- [13] M. Granovetter, The Strength of Weak Ties. *American Journal of Sociology* **78**(6) 1360–1380, 1973.
- [14] M. Granovetter, Threshold models of collective behavior. *American Journal of Sociology* **83**(6) 1420–1443, 1978.
- [15] E. Griechisch and A. Pluhár, Community Detection by using the Extended Modularity. $(CS)^2$ - Conference of PhD Students in Computer Science, Szeged, 2010.
- [16] J. Kleinberg, An Impossibility Theorem for Clustering. *Advances in Neural Information Processing Systems (NIPS)* **15**, 2002.
- [17] J. M. Kumpula, M. Kivela, K. Kaski and J. Saramaki, A sequential algorithm for fast clique percolation. *Phys. Rev. E* **79**, 026109 (2008).
- [18] S. Milgram, The Small World Problem. *Psychology Today*, **1**(1), 60–67, 1967.
- [19] T. Népusz, A. Petróczi, L. Négyessy and F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E* **77**, 016107 (2008).
- [20] M.E.J. Newman, The structure and function of complex networks. *SIAM Rev.* **45**, 167–256, 2003.
- [21] M. E. J. Newman, Detecting community structure in networks. *Eur. Phys. J. B* **38**, 321–330 (2004).

- [22] G. Palla, I. Derényi, I. Farkas and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814 (2005).
- [23] G. Palla, A.-L. Barabási and T. Vicsek, Quantifying social group evolution. *Nature* **446**, 664–667 (2007).
- [24] A. Pluhár On the clusters of social graphs based on telephone log-files. (in Hungarian) Research Report 2001.
- [25] A. Pluhár Determination of communities in social graphs by fast algorithms. (in Hungarian) Research Report 2002.
- [26] G. Thilo, S. Hiroki, (Eds.) *Adaptive Networks*. Springer Verlag, New York, 2009.
- [27] W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452–473 (1977).

Identification of the Place and Materials of Knocking Objects in Flow Induced Vibration

Tibor Dobján^{*†}, Szilveszter Pletl[†], Tamás Deák^{*},
László Doszpod^{*}, and Gábor Pór^{*}

Abstract

Flow induced vibration can be found and identified by acoustic methods. Using acoustic sensors, the first task, the event detection has been solved using the sequential probability ratio test after autoregressive filtering the measured signal. The LABView program and actual test of the event recognition technique are presented. The signals were recorded in a 100 m long test loop having artificially placed flow induced vibrating objects hitting the wall. Time delay between the fronts of detected events has been used to localize the actual place of the acoustic source. The recognition of the material of the knocking object is based traditionally on the spectrum estimation. However, this is rather time consuming task by naked eyes. We are proposing to introduce the skeleton method for event identification.

Keywords: Autoregressive filtering, Sequential Probability Ratio Test, IAEA Benchmark measurement, event detection, skeleton method

Introduction

Many industrial systems contain pipes with fluid flow either for transmitting materials or for cooling purposes. If solid parts of the system are detached or loosened they may go to chaotic or deterministic motion due to forces gained from the flow energy. It is also quite common when either a disattached part of the equipment or just a forgotten object after maintenance work remains in the pipes causing so called loose parts, which might be even dangerous for the given industrial system. If a loose part knocks on the inner surface of the tube (or other compartment) then audible events are generated. These are surface waves traveling on the metal surfaces. The place of the knocks and the knocking material are crucial from the point of view of the fate of the given industrial objects. Therefore detection of

^{*}Hungarian Acoustic Industrial Diagnostic Laboratory, College of Dunaujváros,
E-mail: {dobjan.tibor, doszpod.laszlo, por.gabor}@mail.duf.hu, thomasckik@gmail.com

[†]Department of Technical Informatics, University of Szeged,
E-mail: pletl@inf.u-szeged.hu

the event, finding of its place and identification of knocking material have primary importance from the point of view of the safety and maintenance of the system.

Flow via piping systems in industry may excite vibrations. They are called flow-induced vibration. Some of those vibrations are expressed as eigenvalues of the pipes filled with streaming liquid. In some other cases solid parts transported by the liquid phase may also emit sound. These sounds are regarded as background noise in our case.

The task is to notice sound emissions due to loose parts on a relatively large background level and to identify their origin [1, 2, 3, 4]. In this paper we investigate the following areas. Improvement of the identification of the event recognition using autoregressive (AR) modeling based filtering and sequential probability ratio test (SPRT). The test measurements are described in Section 1. Data processing has been realized in LABVIEW. This was a very important preparation for identifying the event, since the correct event selection is the basic for that. The method and the program test are presented in Section 2.

To give a hint on the material of the knocking objects first we estimated the auto power spectral density function (APSD). Then we divided the total frequency band into high frequency part and low frequency part. It was shown, that the ratio of these partial root mean square (RMS) values are different for knocking object of different materials. We present the first results of division of APSD into several parts. It can be clearly seen, that this may improve the identification. (See Section 4.) Besides the identification the source localization is also important (Section 3.2.).

In event identification we are going to use skeleton method, neural network and later a specifically designed expert system, which has its own library based on experiments (Section 4.).

1 Measurement

1.1 Test section

We carried out measurement sampling signals of accelerometers in frequency range from 1 kHz up to 50 kHz. The lower frequencies may have very high industrial noise, while the frequency range above audible 16 kHz is still containing weak components from the knocks of the loose parts and the background noise is surprisingly weak in that range [2, 4].

The test section where measurements have been made generally serves for cooling a shaker. The test section has two iron tubes: the cold leg and the hot leg (see Fig. 1). We installed 4 accelerometers on the cold leg. We generated internal and external events on the tube. For internal excitation we inserted different small objects in the tube. Due to turbulent flow, small parts fixed on a wire went on wild vibration knocking on inner wall of the tube. For external excitation we hit the outer side of the tube with an iron stack.

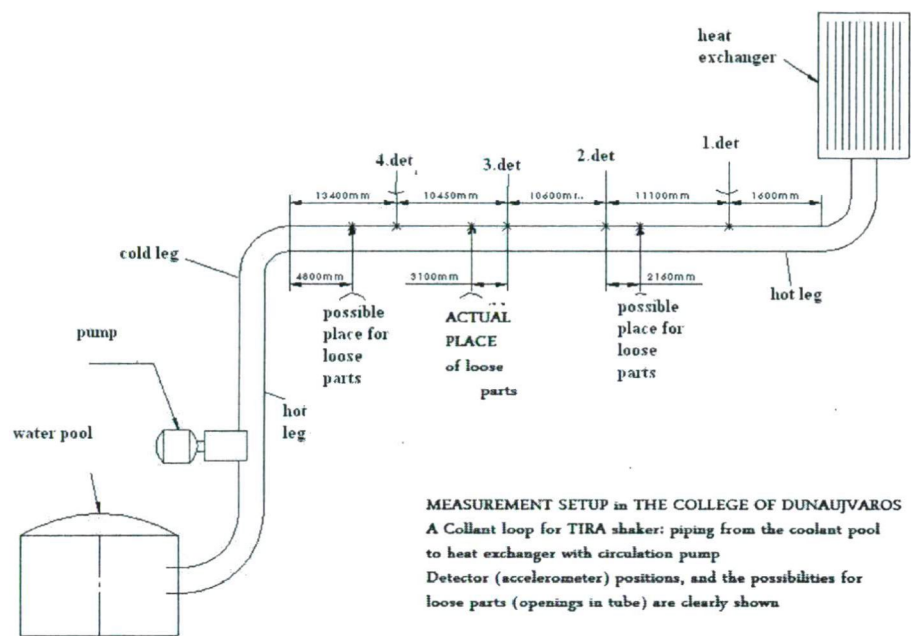




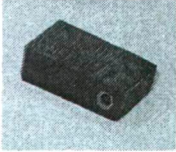
Figure 1: Measurement Setup

We put 3 different types of objects into the tube (Table 1):

- The first was the biggest one: a M10 iron bolt of 11g.
- The second was the middle sized one: M8 iron bolt of about 6g
- And the last one was a piece of Bakelite with weight of 1.7 g.

There was only one object at one time inside the tube.

Table 1: Objects in the tube.

Type	M10 bolt	M8 bolt	bakelite piece
Mass	11,16g	5,58g	1,67g
Photo			

1.2 Measurement software

Based on LABVIEW we developed a software for measurement, which can sample and store the signals of accelerometers up to 4 sensors in ASCII format specified by us.

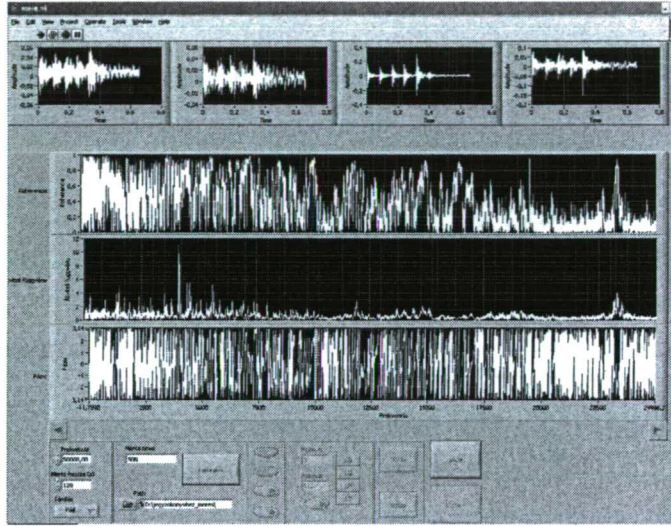


Figure 2: Display of measurement software during sampling

While measuring the program shows the time series, the coherence, the transfer function and the phase in real-time (cf. Fig. 2) for the operator. The user has to set the length of the measure, while a process bar shows the actual state of measuring. The software has three states:

Beginning state First the operator has to set the sampling frequency, the time length of measurement, the method of storage.

Calibrating state In the second step the user can select sensor pairs to see the cross functions. It is possible to zoom into special points of functions.

Measuring state If every sensor pairs has coherence peaks than the last step is the measuring. All the buttons are gray colored except the STOP button.

2 Event Detection

Once we did the record, we have to find and select events from the whole time series. Our preferred method is the Sequential Probability Ratio Test (SPRT), since it is generally accepted [5] and we also found that they are the most effective and reliable for this purposes. However, for carrying out the SPRT first we have to

filter the signal to remove large background noise. We did that using Autoregressive (AR) filtering.

2.1 Autoregressive filtering

We build an autoregressive model using Durbins method on a record without acoustic events, to describe the background without the acoustic events.

The expression of autoregressive modeling is:

$$y_k = \sum_{i=1}^P a_i y_{k-i} + w_k \quad (1)$$

where y are the sampled data, a_i are the autoregressive coefficients, while the w is a white, additional noise. It is supposed, that the autoregressive model will describe everything, what is deterministic in the background having in mind, that P is the degree of freedom of the system. This AR model was used for filtering the actual records with events.

The formula of autoregression filtering:

$$x_k^{\text{filtered}} = x_k^{\text{measured}} - \sum_{i=1}^P a_i x_{k-i}^{\text{measured}} \quad (2)$$

This filtering method is very effective in real-time environment, because after the model has been built, i.e. the autoregressive coefficients have been estimated from the background measurements, the filtering can be made by subtracting the AR modeled data from the actually measured data.

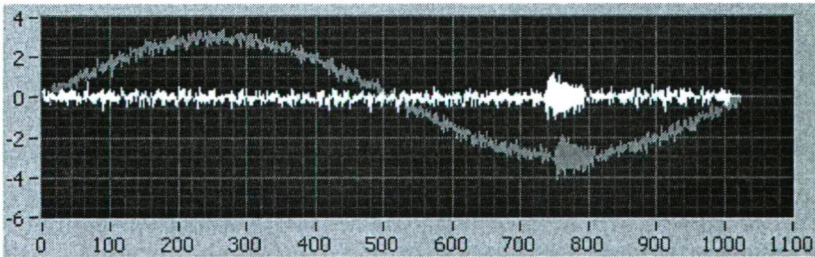


Figure 3: Test of autoregressive filtering

On Fig. 3 one may see a simulated example for autoregressive filtering. The red line shows a generated (simulated) signal: one period of sine function with added white Gaussian noise and one transient event (burst) on it. The white is the filtered signal.

2.2 Sequential Probability Ratio Test

The next step is the Sequential Probability Ratio Test, which serves for event detection in the filtered signal. The theoretical formula of SPRT [2]:

$$\lambda_i = \ln \left(\frac{p(x_1, x_2, \dots, x_i | H_1)}{p(x_1, x_2, \dots, x_i | H_0)} \right) \quad (3)$$

The point-by-point formula of SPRT:

$$\Delta \lambda_i = \lambda_i - \lambda_{i-1} = \ln \frac{f_1(x_i | H_1)}{f_1(x_i | H_0)} \quad (4)$$

Lambda function (3) is a logarithm of a ratio of two conditioned probabilities. In the numerator we have the probability that the samples belong to the probability functions with condition H_1 , i.e. the probability density function of that. While in the denominator we have the probability that the samples belong to the probability function conditioned by hypothesis H_0 . If we substitute the normal distribution probability variable density function to the elementary lambda increment function of SPRT (4) then we get the following equation (5):

$$\Delta \lambda_i = \frac{\sigma_1^2 - \sigma_0^2}{2\sigma_1^2\sigma_0^2} x_i^2 - \ln \frac{\sigma_1}{\sigma_0} \quad (5)$$

To take decision by the SPRT lambda function we need two parameters A and B . When the measured signal similar in statistical sense to the background, i.e. when process can be described by Hypothesis H_0 the lambda function has a negative increment values and tends to go to negative infinity, step by step. We take decision, that the signal belongs to background type when the value of the lambda function exceeds value A (becomes less than A). When the measured signal deviation is different from the background then lambda function has positive increment value and tends to raise. We take decision, when it exceeds value B .

A can be calculated from the probability inequality [2]

$$1 - AFP \leq FAP \cdot e^A \quad (6)$$

Expressing A :

$$A = \ln \frac{AFP}{1 - FAP} \quad (7)$$

B can be calculated also from:

$$1 - AFP \geq FAP \cdot e^B \quad (8)$$

Expressing B :

$$B = \ln \frac{1 - AFP}{FAP} \quad (9)$$

where AFP means the Alarm Failure Probability and FAP means False Alarm Probability.

Table 2: The A and B parameters

AFP	FAP	A	B
10%	10%	-2,2	2,2
1%	1%	-4,6	4,6
0,1%	0,1%	-6,91	6,91
1%	10%	-4,5	2,29
0,1%	10%	-6,8	2,3

Calculated [2] values for A and B using equation (7) and (9) are presented in Table 2.

If the value of λ hits one of the limits then SPRT takes a decision A or B , where A means that, there was no any changes in the signal in comparison to the background, and B means that something has happened in the signal. After a decision has been taken we set the next λ value to zero.

There is a trivial example on Fig. 4 showing how SPRT works. One can see the acoustic event in the upper time signal. SPRT shown in the bottom window goes down regularly exhibiting a saw tooth shape until there is only background noise in the time signal. At the beginning of the acoustic event λ function will start to grow toward the positive values. However, this is a trivial case, here one does not need really the SPRT for event recognition.

There is a more realistic case on Fig. 5. The original sampled signal is shown in red color. The white signal in the upper window is that signal filtered by AR. Without the help of SPRT λ function it is rather difficult to select real events, real burst even in the filtered signal. With SPRT events are clearly marked, and really one can see, that there is something in the filtered signal, which deviates at those time spots from the general behavior of the background. SPRT senses the standard deviation differences, where human eye doesn't see it at all.

We processed by this method all measured signals described in Section 1.

3 Analyses of selected time sequences

3.1 Identification of the event using spectra

The next task is to characterize the records classified by SPRT method into different classes. We calculated the averaged Power Spectrum Density function for backgrounds (A decisions of SPRT) and the events (B decisions of SPRT). They are shown on Figs. 6 and 7, respectively.

It is clearly seen on Fig. 6, that there are two peaks a and b sitting on the noise level when there was no event in the measurement. The noise level is falling with the growing frequency. We believe that peaks are due to eigenfrequencies of the tube, while the falling noise level is due to friction of the flow. They characterize

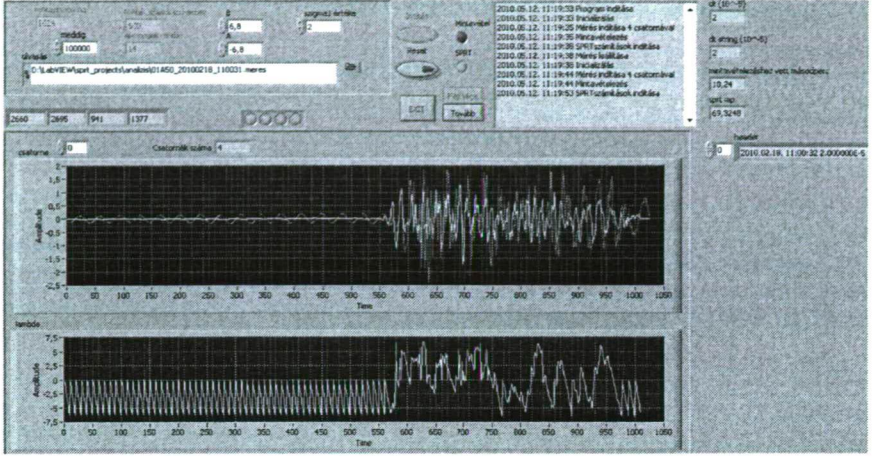


Figure 4: Trivial example

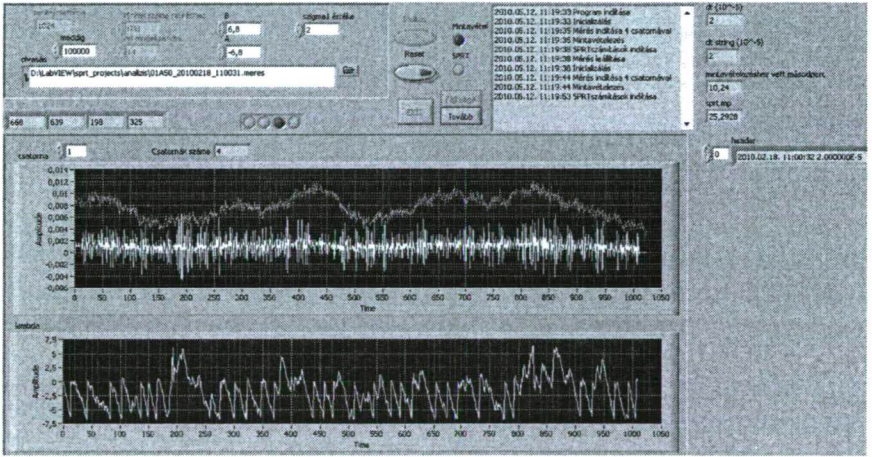


Figure 5: Real measurement example

the background noise of the measurements.

One can see a and b peaks and falling noise level from the background measurement in the average power spectrum density function of events as well (Fig. 7). But there are 3 other specific peaks (I,II,III) which are responsible for the events.

The numerical difference between the spectrum of events and the spectrum of noises (Fig. 8), shows the specific descriptors of the event. The smaller peaks are due to uncertainties in standard deviation of the measured signal, but the bigger peaks marked on figure are useful for identification purposes.

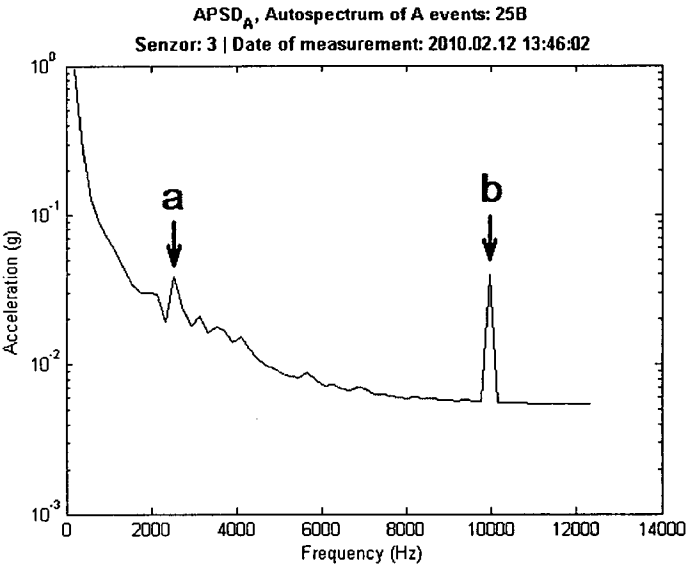


Figure 6: The averaged Power Spectrum Density function of background

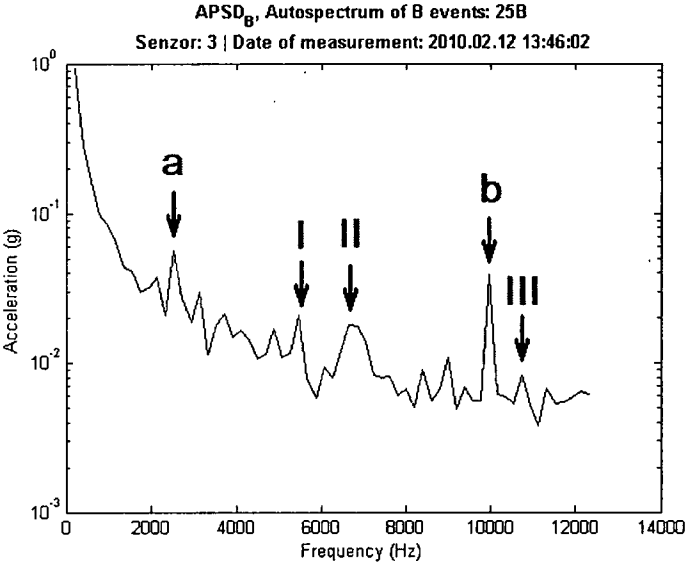


Figure 7: Averaged Power Spectrum Density function of acoustic events, what selected by SPRT.

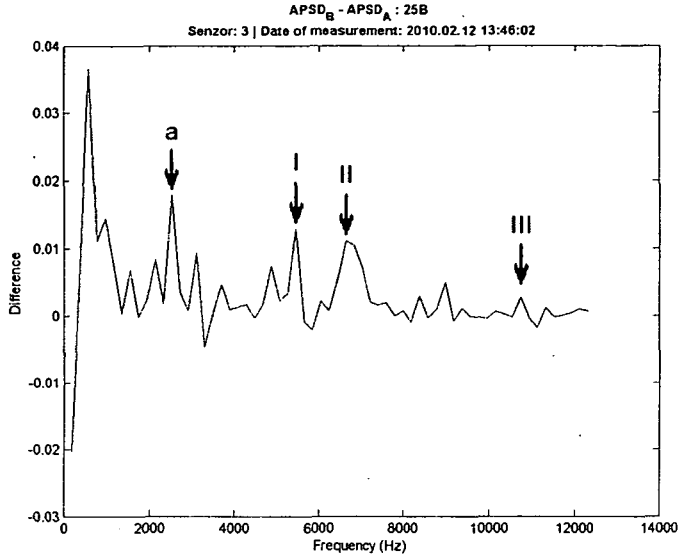


Figure 8: The numerical difference between the two spectra, B-A.

3.2 Localization

The simplest method for localization is based on the front of the events identified by SPRT. There are two cases clearly depicted on Figs. 9 and 10. The impact can be either outside or inside of the section determined by the two sensors.

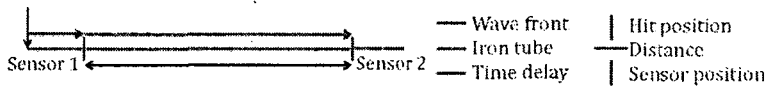


Figure 9: Hit outside the analyzed interval.

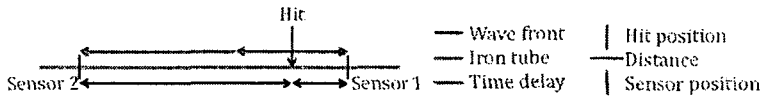


Figure 10: Hit inside the analyzed interval.

On these figures we used the following notations:

- The orange line is the tube itself.
- The purple arrow shows the point of hits.

- The blue arrows show the propagation of the front of wave until it arrives to the first sensor.
- The green arrows show the propagation after the front has passed the first sensor. It is the time difference. It is the time delay.
- The black line shows the known distances in Fig. 9 and the unknown distances in Fig. 10.

For localization procedure we have to estimate the time delays between the events measured by different sensors.

3.2.1 Method using Cross Correlation Function (CCF)

In case of traditional localization one can calculate the Cross Correlation Function (Fig. 11) to estimate the time delay and thus to localize the hits. The argument of maximum of this function shows the time delay between the two signals.

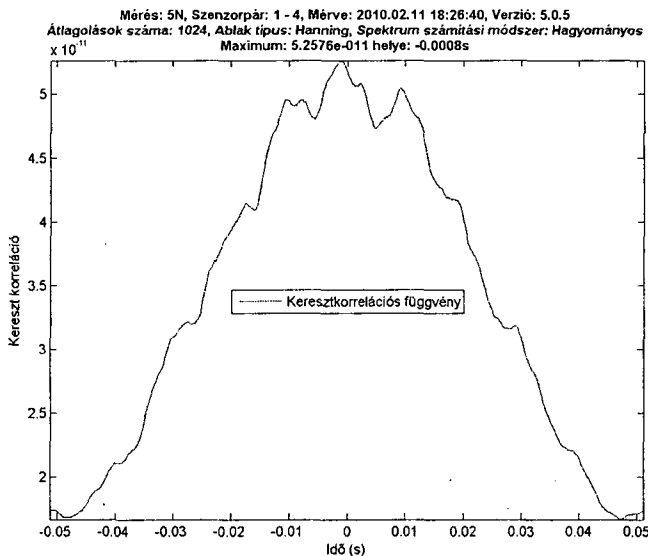


Figure 11: Example CCF

3.2.2 Method using Impulse Response Function (IRF)

There is another function in traditional localization procedures [3]: one can calculate the Impulse Response Function (Fig. 11) to estimate time delays and thus to localize the hits. The argument of maximum of this function shows the time delay between the two signals, like the cross correlation function. This method is

better in the sense that it may show also the velocity distribution, but it has higher uncertainties.

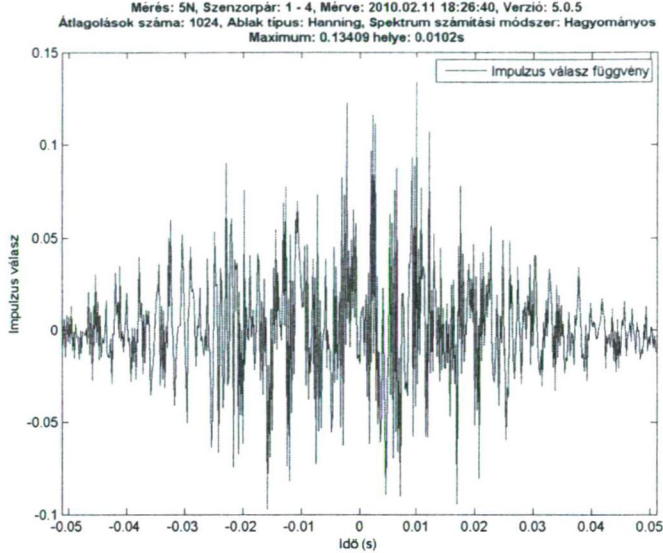


Figure 12: Example of IRF

3.2.3 New method

The new way of localization is to use learning algorithms. We hit the tube several times on the marked points on Fig. 13. Then we calculate different description parameters from power spectrums to train the learning algorithm (for example neural network).

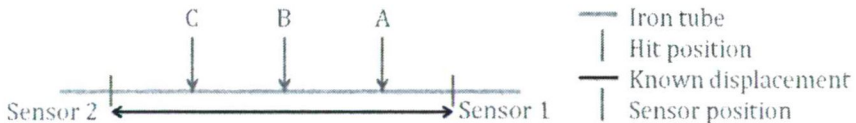


Figure 13: Localization with identification

4 Identification

4.1 Traditional way

To distinguish the material of the hitting object at the beginning we applied the spectrum division method proposed in [7]. We divide the power spectrum of acoustic events into four parts (see Fig. 14).

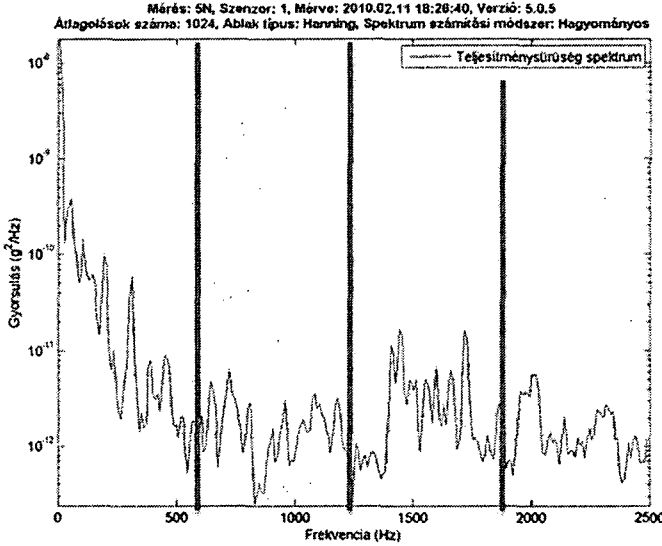


Figure 14: Spectrum dividing method

Using this spectrum division we can estimate their RMS Root Mean Square value by formula:

$$RMS_i = \Delta f \times \sum_{j=(i-1)(n/4)}^{i \cdot (n/4) - 1} APSD_j \quad (10)$$

In this way, we have got 4 numbers, which characterize the actual event. This method is insensitive for the position of peaks. However, the peaks contain important information. For example cutting a peak into two parts can produce big mistake in this method.

4.2 New method

We are looking for methods to describe spectra in a more effective way using machine learning processes.

We are trying to calculate the skeleton of spectrums (Fig. 15). The endpoints of skeletons are positive peaks that we are interested in. We borrowed the skeleton

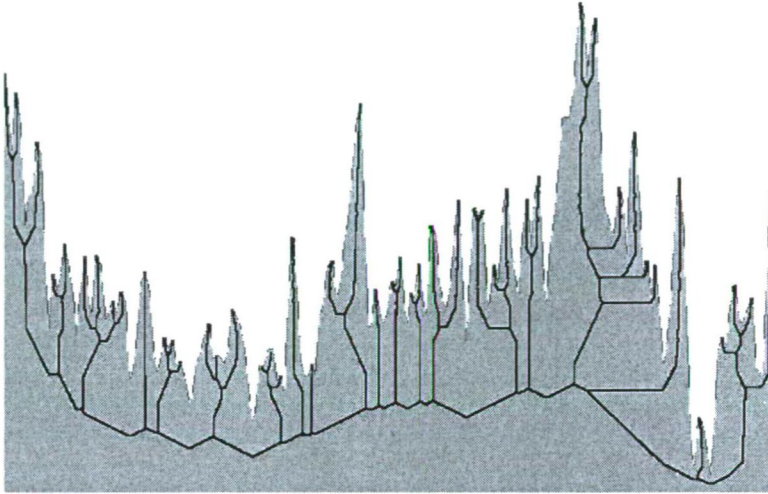


Figure 15: Skeleton of a spectrum

method and program from Gábor Németh [8]. It produces skeleton which has different nodes and different distribution of the branches. In the future we shall use these parameters for characterizing the event.

5 Summary and Conclusions

The fact that sequential probability ratio test is one of the best methods for identifying abnormal events had been proposed and demonstrated earlier cf. [1, 2, 3, 4, 5]. However, earlier only offline data evaluation programs have been used to select events in high background noise. The presented LABVIEW program opens the way for on-line application of the SPRT method for event selection. A benchmark measurement has been evaluated to show the capability of the event selection method. Efficiency of the method was clearly demonstrated in our paper. Spectral estimation of selected methods clearly pointed at differences of spectral components, when different object impacted the wall. This opens the route for object identification using autospectrum. However, it is not easy to automate the spectrum pattern recognition having several different peaks in the spectrum. We are involving the skeleton method. For the other important task, for the localization of the impact place we tried three methods: we used time delay of the fronts of the events identified by SPRT; we used the cross correlation and the impulse response estimation to retrieve the time delay. The front cannot be estimated very precisely; the cross-correlation seems to be too wide for precise time delay estimation; the impulse response has rather large uncertainty. We tried a new method based on neural network, which seems to be very promising.

References

- [1] Szappanos, G., Feher, A., Lorincz, J., Nemes, L., Por, G., Csikos, T., Glodi, O., Lipcsei S., Tri, T. D. A New Digital Expert Loose Part Detection System. *Annals in Nuclear Energy*, 24(14):1097–1103, 1997
- [2] Szappanos, G., Por, G. Improvements in the Theory of Identification of Burst-Shaped Events for Fault Diagnosis. *Nuclear Science and Engineering*, 13:261–267, 1999
- [3] Por, G., Szappanos, G. Advanced Loose Part Monitoring System for Nuclear Power Plants. *Proc. of International Conference Nuclear Energy in Central Europe 2000*
- [4] Por, G., Szappanos, G. ALPS, Advanced Loose Parts System for Paks NPP. *Proc. of International Conference Nuclear Energy in Central Europe 2001*
- [5] Schoonewelle, H., Van Der Hagen, T. H. J. J., and Hoogenboom, J. E. A comparison of three time-domain anomaly detection methods. *Ann. Nucl. Energy*, 23(2):159–170, 1996
- [6] Por, G., Berta, M., Csúvár, M. Measurement of the coolant flow rate using correlation of temperature fluctuations. *Progress in Nuclear Energy*, Proceedings of the 8th Symposium on Nuclear Reactor Surveillance and Diagnostics, 43(1-4):281-288, 2003
- [7] Tsunoda, T., Kato, T., Hirata, K., Sekido, Y., Sendai, K., Segawa, M., Yamatoku, S., Morioka, T., Sano, K., and Tsuneoka, O. Studies on the loose part evaluation technique. *Progress in Nuclear Energy* 15:569-576, 1985
- [8] Kardos, P., Németh, G., and Palágyi, K. An OrderIndependent Sequential Thinning Algorithm. Wiederhold, P. and Barneva, R. P., editors, *IWCIA 2009, LNCS 5852* pages 162–175, 2009

Community Detection by using the Extended Modularity

Erika Griechisch and András Pluhár

Abstract

This article is about community detection algorithms in graphs. First a new method will be introduced, which is based on an extension [16] of the commonly used modularity [17, 18, 19, 20] and gives overlapping communities. We list and compare the results given by our new method and some other algorithms yielding either overlapping or non-overlapping communities. While the main use of the proposed algorithm is benchmarking, we also consider the possibility of hot starts, and some further extensions that considers the degree distribution of the graphs.

Keywords: graph mining, community detection, fuzzy partition, modularity

1 Motivation

Graph mining has become an important field of data mining, especially the community detection is very popular, because it can be applied in sociology, computer science, biology or finance.

The main problem in community detection is that there is no exact definition for communities; all we can do is to postulate some properties. One approach is that a community is a set of vertices that are relatively densely connected to each other, but sparsely connected to other dense groups in the graph. Several methods are proposed for community detection. The majority of those define non-overlapping communities (clusters, partitions) [24, 13, 2], some of them define overlapping sets (henceforward *communities*), see [1, 22, 11].

However, it became clear that overlapping communities must be also considered in Small World graphs, see [10, 12, 22, 16]. Moreover, the given algorithms turned out to be useful in data mining and modeling [8, 9]. In fact, one way to evaluate the performance of different algorithms is to check their predictive value in models [11]. Another possibility is to define plausible measures. The most successful of those is the Newman modularity that is defined for a clustering of a graph. The higher the modularity, the better the clustering [17]. Nevertheless, for large graphs the modularity is just approximated [7, 5], since it is an NP-hard problem to maximize modularity [6].

Népusz et al. [16] proposed an algorithm for the detection of overlapping communities in quadratic programming terms. To measure their results they also extended the notion of modularity to overlapping communities. We continue their work, but change the objective function, and examine the possibility of directly optimizing the extended modularity. Of course, one cannot expect an effective algorithm for this, since it is also a quadratic optimization problem. Still, it can be computed for small graphs, and provides good benchmarks for the community detection heuristics. It is also instructive, how good approximation of the modularity is achieved by those heuristics?

Other methods for extending modularity exist. Nicosia et al. [21] introduce a general extension of the modularity on directed graphs which moves from simple considerations about the meaning and structure of the original modularity function, using an enriched null-model.

2 Definitions

A graph G with vertex set V and edge set E will be denoted $G = (V, E)$, the cardinality of the vertex set will be denoted n , and the cardinality of the edge set will be denoted m . In this paper every graph is undirected and unweighted. The objective of classical community detection in networks is to partition the vertex set of the graph into c distinct subsets in a way that puts densely connected groups of vertices in the same community. Here c can either be given in advance or determined by the community detection algorithm itself. If c is known, a convenient representation of a given partition is the partition matrix $\mathbf{U} = [u_{ik}]$. \mathbf{U} has $n = |V|$ rows and c columns, and $u_{ik} = 1$ if and only if vertex i belongs to the k th subset in the partition, otherwise it is zero [3]. From the definition of the partition, it follows that $\sum_{k=1}^c u_{ik} = 1$ for all $1 \leq i \leq n$. The size of community k can then be calculated as $\sum_{i=1}^n u_{ik}$, and for any meaningful partition, we can assume that $0 < \sum_{i=1}^n u_{ik} < n$. These partitions are called *hard* or *crisp* partitions, because a vertex can belong to one and only one of the detected communities [3]. The generalization of the hard partition follows by allowing u_{ik} to attain any real value from the interval $[0, 1]$. The constraints imposed on the partition matrix remain the same:

$$u_{ik} \in [0, 1], \quad \forall 1 \leq i \leq n, 1 \leq k \leq c; \quad (1)$$

$$\sum_{k=1}^c u_{ik} = 1, \quad \forall 1 \leq i \leq n; \quad (2)$$

$$0 < \sum_{i=1}^n u_{ik} < n, \quad \forall 1 \leq k \leq c. \quad (3)$$

Equation 2 simply states that the total membership degree for each vertex must be equal to one. Informally, this means that vertices have a total membership degree of one, which will be distributed among the communities. Inequality 3 is the formal

description of a simple requirement: we are not interested in empty communities (to which no vertex belongs to any extent), and we do not want all vertices to be grouped into a single community. Partitions of this type are called *fuzzy partitions*.

Several methods have been developed to search fuzzy clusters (see e.g. [4]), but we can not apply these to graph partitioning, because these methods need some additional data.

In [16] N  pusz et al. use a different approach using vertex similarities. They observe that a meaningful partition should group vertices that are somehow similar to each other in the same community and assume that a similarity function $s(\mathbf{U}, i, j)$ satisfies the following criteria:

- $s(\mathbf{U}, i, j) \in [0, 1]$
- $s(\mathbf{U}, i, j)$ is continuous and differentiable for all u_{ij}
- $s(\mathbf{U}, i, j) = 1$ if the membership values of vertex i and j suggest that these are as similar as possible
- $s(\mathbf{U}, i, j) = 0$ if the membership values of vertex i and j suggest that these are completely dissimilar.

From now on, we denote $s(\mathbf{U}, i, j)$ by s_{ij} . We use the similarity matrix from [16], that is

$$s_{ij} = \sum_{k=1}^c u_{ik}u_{jk}. \quad (4)$$

In [16] the following objective function was optimized

$$D_G(\mathbf{U}) = \sum_{i=1}^n \sum_{j=1}^n w_{ij}(\tilde{s}_{ij} - s_{ij})^2, \quad (5)$$

where w_{ij} 's are optional weights and \tilde{s}_{ij} is a prior assumption of the actual similarity of the vertices. Instead of using Equation 5, we extend the commonly used modularity [17, 18, 19, 20], and optimize this objective function directly over the space of feasible solutions.

Modularity is a property of a network and a specific proposed division of that network into communities. The measure of a division is large, if most of the edges are within clusters and only a few are between those.

Let A be the adjacency matrix of the network, k_i the degree of vertex i , $\delta(c_i, c_j)$ is the Kronecker delta and suppose that vertex i belongs to community c_i . Then

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

is the modularity of the given division.

The modularity can be either positive or negative. Positive values indicate the possible presence of community structure. Thus one can search for community

structure precisely by looking for the divisions of a network that have positive, and preferably large values of the modularity.

By replacing $\delta(c_i, c_j)$ with s_{ij} we get a quadratic function called extended (fuzzy) modularity [16]:

$$Q' = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] s_{ij} \quad (6)$$

$$= \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \langle u^i, u^j \rangle, \quad (7)$$

where u^i is the i th row in the fuzzy partition matrix \mathbf{U} and $\langle \cdot, \cdot \rangle$ denotes the dot product.

So our the problem is to

$$\begin{aligned} & \text{maximize} && Q' \\ & \text{subject to} && u_{ik} \in [0, 1], \quad \forall 1 \leq i \leq n, 1 \leq k \leq c; \\ & && \sum_{k=1}^c u_{ik} = 1, \quad \forall 1 \leq i \leq n; \\ & && 0 < \sum_{i=1}^n u_{ik} < n, \quad \forall 1 \leq k \leq c. \end{aligned}$$

Higher modularity usually means better division, so our aim was to maximize the extended modularity directly, and thereby gaining a benchmark compare the results given by other methods.

3 Examined methods

This section introduces the compared methods. The results of these methods were not just compared in a sense of modularity value, but the detected communities were given to the quadratic solver as an initial point, thus we used their results as a “hot starts.”

In the case of overlapping communities, it is not obvious how to choose the community membership vectors. We decided to use uniform distribution, e.g. if the CPM (or N^{++}) method produces 4 communities and the community 1 and 2 also contains vertex v , then the membership vector of v is $(0.5, 0.5, 0, 0)$.

Table 1 shows the running time of the examined methods depending on the number of nodes n , the number of edges m and the average degree d_{avg} .

3.1 Community detection based on edge betweenness

The method of Girvan and Newman [18] is based on the definition of edge betweenness. The edge betweenness of an edge is the number of shortest paths between pairs of vertices run along it. Iteratively removing the edges with highest betweenness, we can determine a hierarchical tree and then communities.

3.2 Leading eigenvector method

Newman showed in [19] that the modularity can be expressed in terms of the eigenvectors of a characteristic matrix for the network, which he calls the *modularity matrix*, and that this expression leads to a spectral algorithm for community detection that returns better results in shorter running times than competing methods, e.g. edge betweenness.

3.3 Greedy modularity optimization method

Clauset, Newman and Moore presented a hierarchical agglomeration algorithm for detecting community structure [20]. This algorithm uses a greedy optimization in which, starting with each vertex being the sole member of a community of one, repeatedly join together the two communities whose amalgamation produces the largest increase in the modularity Q .

3.4 Label propagation

The label propagation method [25] is based on the following iteration. Suppose that a node x has neighbors and each neighbor carries a label denoting the community to which they belong to. Then x determines its community based on the labels of its neighbors. We assume that each node in the network chooses to join the community to which the maximum number of its neighbors belong to, such that the occurring ties are broken uniformly randomly.

At the beginning every vertex gets a unique label. Iteratively at every step, each node updates its label based on the labels of its neighbors. The iterative process continues until no node in the network changes its label.¹

3.5 Walktrap

The walktrap method uses the idea of Markov chains, and it is based on random walks to tell a distance for every pair of vertices in the graph and with this it generalizes the distance of clusters [23].

3.6 Clique percolation

The clique percolation method (CPM) uses adjacent cliques to determine overlapping communities [12, 22]. It builds up the communities from k -clique (fully connected graphs), and two k -cliques are considered adjacent if they share $k - 1$ nodes. Then communities are the unions of the adjacent k -cliques. Note that the result heavily depends on the value of the parameter k .

¹The algorithm may give oscillation if the input is a bipartite graph; this case should be handled.

3.7 N^{++} method

The

$$N^{++}(v) = N^+(N^+(v)) = \{u \in V \mid d(u, v) \leq 2\}$$

is the set of vertices which are “close” to v . Using these sets and their dense subsets, the method provides overlapping communities [10, 11].

Table 1 shows the running time, where n is the number of vertices, m is the number of edges in the graph.

Table 1: Running time of the compared methods

Method	Running time worst case	
Edge betweenness	$\mathcal{O}(m^2n)$	[18]
Leading eigenvector	$\mathcal{O}(m + n^2 \cdot \text{steps})$	[19]
Newman greedy	$\mathcal{O}(md_{\text{avg}} \log n)$	[20]
Label propagation	$\mathcal{O}(m + n)$	[25]
Walktrap	$\mathcal{O}(mn^2)$	[23]
CPM	$\mathcal{O}(\exp(n))$	[12]
N^{++}	$\mathcal{O}(\exp(n))$	[10]

3.8 Sequential greedy method

We also proposed a new sequential greedy (SG) method that might be considered as a heuristic for the modularity optimization problem. We determined the number of the communities (c) as the size of a maximal independent set I and placed those into different clusters. (The membership vector of the k th vertex of I is e_k , the k th element of the standard orthonormal base.) Then we greedily decide about the membership vector of the leftover vertices using a partial modularity based on the already placed vertices. Since the extended modularity Q' (see Equation 7)

$$Q' = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \langle u^i, u^j \rangle,$$

in step j we set u^j such that it maximizes the partial modularity function Q_j , where

$$Q_j = Q(u^j) = \frac{1}{2m} \sum_{i < j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \langle u^i, u^j \rangle = \frac{1}{2m} \left\langle \sum_{i < j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] u^i, u^j \right\rangle,$$

and u^j satisfies the constraints 2, 3. Obviously the function $Q(u^j)$ reaches its maximum if u^j equals the k th element of the standard orthonormal base, where k is the position of the largest coordinate of the vector $\sum_{i < j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] u^i$.

4 Technical background

This section describes the main software tools which were used during the research.

4.1 igraph library

The *igraph* [29] is a free software package for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, such as the community structure search.

The igraph contains functions for generating regular and random graphs, manipulating graphs, assigning attributes to vertices and edges. It can calculate various structural properties, graph isomorphism, includes heuristics for community structure detection and supports many file formats.

We used the igraph functions `IC_edge_betweenness`, `IC_leading_eigenvector`, `IC_fastgreedy`, `IC_spinglass` and the `IC_walktrap` functions (where IC equals to `igraph_community`) to determine the corresponding community structure and the function `igraph_modularity` to determine the modularity value of the clustering.

4.2 octave

GNU Octave is a high-level language, primarily intended for numerical computations which is mostly compatible with Matlab. We used the octave function `sqp` to optimize the objective function which is an implementation of the SQP.

The sequential quadratic programming (SQP) is one of the most popular and robust algorithms for nonlinear continuous optimization. The method is based on solving a series of sub-problems designed to minimize a quadratic model of the objective subject to a linearization of the constraints.

4.3 Visualize graphs

Several methods exist for visualizing graphs, we used the `igraph_layout_graphopt` function, which optimizes vertex layout via the graphopt algorithm.

In contrast to other graph optimizers, graphopt does not use a finite-pass approach to layout optimization. Instead, it uses basic principles of physics to iteratively determine an optimal layout.

Each node is given a mass and an electric charge, and each edge is represented as a spring. The node mass, the electric charge, the optimal spring length, and the spring constant are tweakable in the gui in real time [30].

4.4 Communities

We used CFinder for CPM and the implementation of the N^{++} method is due to Csizmadia [1, 10]. We would like to express our thanks for obtaining free access to the appropriate softwares.

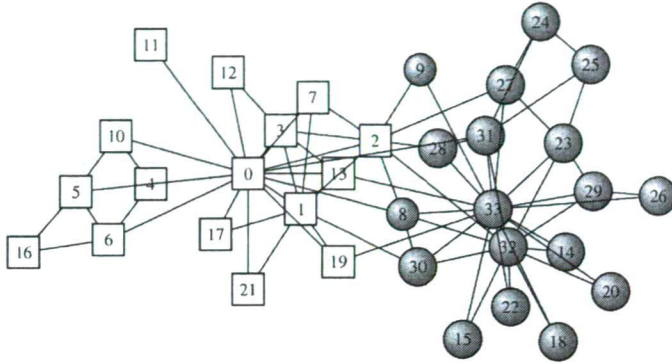


Figure 1: Zachary graph – The instructor and the administrator are represented by nodes 0 and 33. White squares represent individuals who ended up aligning with the club’s instructor after the split, shaded circles those who aligned with the administrator.

5 Results

Newman has released his benchmark graphs in [28]. We have chosen the following graphs from his database to examine:

Zachary’s karate club: social network of friendships between 34 members of a karate club at a US university in the 1970s.

Les Miserables: co appearance network of characters in the novel Les Miserables.

Books about US politics: A network of books about US politics published around the time of the 2004 presidential election and sold by the on-line bookseller Amazon.com.

Several researchers worked with these graphs, e.g. [17, 18, 19, 20, 21] from the perspective of community structure.

5.1 Zachary’s karate club network

The first example is taken from one of the classic studies in social network analysis. In the late 1970s Wayne Zachary observed social interactions between the members of a karate club at an American university [27]. The club split in two as a result of an internal dispute, see Figure 1.

The critical node of the graph is the node 2, which (in real life) stayed with the node 0 (he was the instructor). It is critical, since many clustering methods fail to find the correct community it belongs to. Maximizing extended optimality performs well: in every case (from $c = 2$ to $c = 6$) node 2 was assigned correctly.

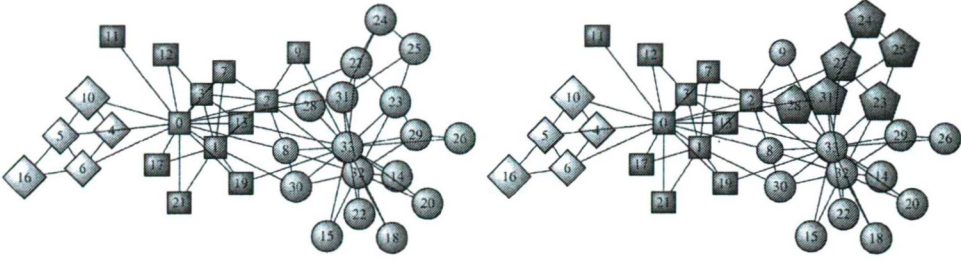


Figure 2: Zachary graph – The communities determined by the optimal extended modularity. Different shapes denote different community. (a) $c = 3$ (b) $c = 4$

Table 2: Zachary graph – membership vectors

Node	Membership vector
0 (inst)	(1.000000,0.000000,0.000000,0.000000,0.000000)
2	(0.985774,0.000000,0.000000,0.000000,0.014226)
9	(0.000000,0.014226,0.000000,0.000000,0.985774)
23	(0.000000,0.117156,0.882844,0.000000,0.000000)
32	(0.000000,1.000000,0.000000,0.000000,0.000000)
33 (adm)	(0.000000,1.000000,0.000000,0.000000,0.000000)

We tried a number of communities from $c = 2$ to $c = 6$. In these cases the optimal extended modularity determines a strict partition, except in the case $c = 5$. The nodes 2, 9 and 23 were assigned to more than one communities. All these nodes lie on the brink of two communities, see Table 2.

The result of the case $c = 3$ coincided with the result of label propagation method [25]. The optimum of the extended modularity in case $c = 4$ gave the same result as the walktrap method [23]. The case $c = 4$ gave the highest modularity (0.4198), above $c = 4$ the value of modularity started decreasing. See the modularity values in Table 3 .

5.2 “Les Miserables” graph

In the “Les Miserables” graph (see Figure 3) nodes represent characters in Victor Hugo’s novel “Les Miserables” as indicated by the labels and edges connect any pair of characters that appear in the same chapter of the book [14].

The Table 5 is the summarizing table of the modularity values. In case $c = 4, 5$ and 8 the optimal extended modularity determines strict partitions again, but in the case $c = 11, 12$ we got some overlaps. In Table 4 the reader can see the non-binary membership vectors when the number of communities is 12.

Table 3: Modularity values – Zachary graph

Method	Modularity	No. of comm.
Edge betweenness	0.4013	5
Leading eigenvector	0.3727	3
Newman greedy	0.3807	3
Label propagation	0.4020	3
Spinglass	0.4063	6
Walktrap	0.4198	4
CPM ($k = 3$)	0.2438	3
CPM ($k = 4$)	0.2557	3
N^{++}	0.1947	12
Sequential greedy	0.3599	2
Optimum of the extended modularity	0.4086	6
	0.4159	5
	0.4198	4
	0.4020	3
	0.3718	2

Table 4: “Les Miserables” graph - Membership vectors of some nodes ($c = 12$)

Node	Membership vector
Valjean	(0.00, 0.00, 0.00, 0.00, 0.00, 0.31, 0.69, 0.00, 0.00, 0.00, 0.00, 0.00)
Marguerite	(0.06, 0.00, 0.21, 0.00, 0.00, 0.73, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Isabeau	(0.00, 0.00, 0.00, 0.00, 0.00, 0.31, 0.00, 0.00, 0.00, 0.00, 0.69, 0.00)
Gervais	(0.00, 0.00, 0.00, 0.73, 0.00, 0.27, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Tholomyes	(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.75, 0.00, 0.00)
Scaufflaire	(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.75, 0.00, 0.00)
Woman1	(0.00, 0.00, 0.00, 0.78, 0.00, 0.22, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Boulatruelle	(0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.68, 0.00, 0.00, 0.00, 0.00, 0.00)
Woman2	(0.00, 0.56, 0.00, 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00, 0.27)
MotherPlutarch	(0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.68, 0.00)
Toussaint	(0.00, 0.56, 0.00, 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00, 0.27)

Table 5: Modularity values – “Les Miserables” graph, *indicates the non-strict (fuzzy) partitions

Method	Modularity	No. of comm.
Edge betweenness	0.5381	11
Leading eigenvector	0.5116	4
Newman greedy	0.5006	5
Label propagation	0.5222	5
Spinglass	0.3809	12
Walktrap	0.5214	8
CPM ($k = 6$)	0.4041	4
CPM ($k = 7$)	0.4359	5
N^{++}	0.2946	16
Sequential greedy	0.4705	7
	0.4805	5
	0.4658	4
Optimum of the extended modularity	0.4489*	12
	0.5402*	11
	0.5453	8
	0.5562	5
	0.5218	4

Table 6: Modularity values – Graph of political books [15]

Method	Modularity	No. of comm
Edge betweenness	0.5168	5
Leading eigenvector	0.4516	4
Greedy	0.5020	4
Label propagation	0.4946	3
Walktrap	0.5253	4
CPM ($k = 3$)	0.4695	4
N^{++}	0.1656	34
Sequential greedy	0.4243	5
	0.4578	4
	0.3883	3
The optimum of the extended modularity	0.5217	5
	0.5254	4
	0.5269	3

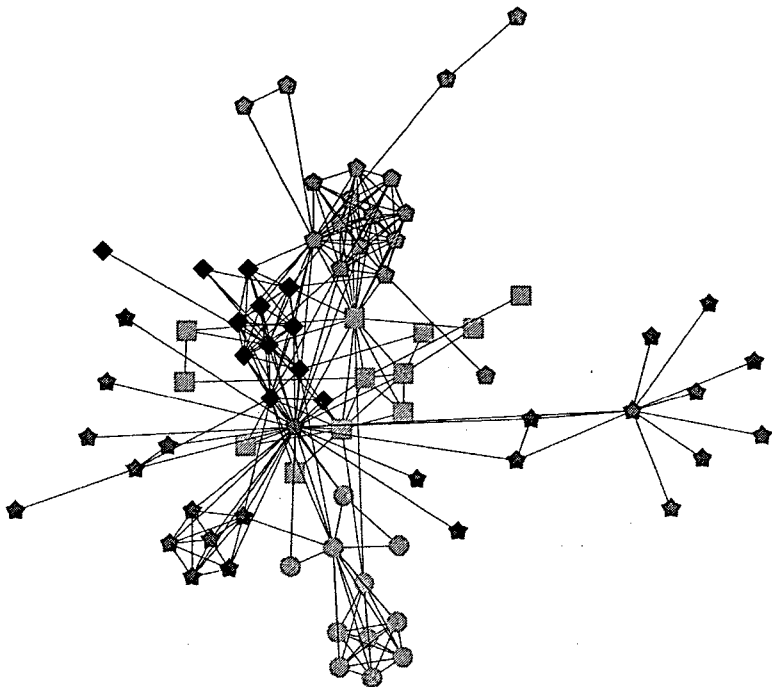


Figure 3: “Les Miserables” graph – Community structure with highest modularity ($c = 5$)

Table 7: Graph of political books – distribution of the same type books among communities

Community or cluster	Edge betweenness	Eigen- vector	Walktrap	Label propagation	Newman greedy	CPM
liberal/neutral/conservative books						
0	3/2/2	0/1/2	5/4/3	4/4/2	5/4/3	1/2/0
1	0/3/42	0/2/35	0/2/39	0/7/46	0/6/43	0/4/44
2	39/2/1	43/7/3	38/2/1	39/2/0	38/2/1	43/4/6
3	0/4/4	0/3/9	0/5/6	–	0/1/2	0/4/4
4	1/2/0	–	–	–	–	–

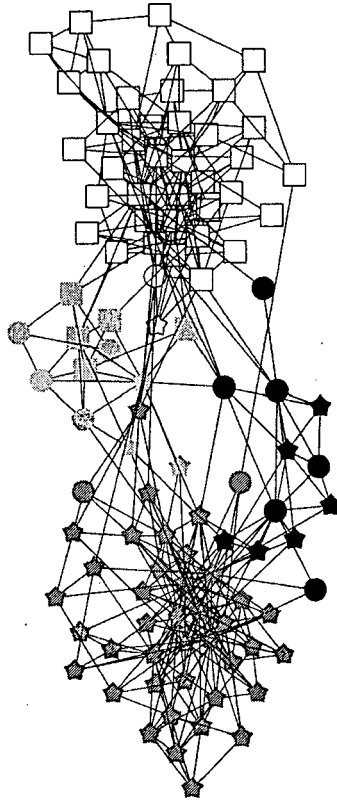


Figure 4: Political books graph – The communities in case $c = 4$, the modularity is 0.5254. The intensity shows the community, the shape indicates the original type of the book (square - liberal, circle - neutral, star - conservative).

5.3 Political books graph

The nodes in political books graph represent books about US politics sold by the online bookseller Amazon.com. Edges represent frequent co-purchasing of books by the same buyers, as indicated by the “customers who bought this book also bought these other books” feature on Amazon.

Nodes have been given values ℓ , n , or c to indicate whether they are *liberal*, *neutral*, or *conservative*. These alignments were assigned separately by Mark Newman based on a reading of the descriptions and reviews of the books posted on Amazon.

The optimum value of extended modularity in case $c = 3, 4$ and 5 gave strict partitions. In case $c = 4$ the optimal extended modularity gave almost the same division as the walktrap method, just one node was classified to another group, see Table 6.

The Table 7 shows the distribution of the book types among communities. It clearly shows, that every methods found two big and few (1, 2 or 3) small communities. One of the big communities contains mainly liberal books, the other contains conservative. The examination of the community/cluster 1 shows it mainly contains conservative book and few neutral, but none of the methods classified any liberal book to this class. As a contrast the community/cluster 2 mainly contains liberal books and some neutral, but almost every methods (except the label propagation) classified some conservative book in this community.

The rest of the clusters/communities are small and there are few which contains both liberal and conservative. The majority contains liberal and neutral or conservative and neutral books as we expect from a clustering method.

6 Conclusion

A new community detection method was proposed based on the definition of fuzzy partition and extended modularity. It was showed that the optimum of the extended modularity can determine overlapping and can give similar or sometimes better results, than earlier heuristics. However, it takes considerable more time to determine the optimum of the extended modularity, because it needs a solution of a quadratic optimization problem.

In solving QP problems it is important to find a good initial point. Thus we tried out the outputs of other methods proposed in [12, 22, 18, 19, 25] (see in Table 3, 5) as an initial point. Then the running time was reduced, but it is still too long. So the proposed method can be used only for small social networks.

The other proposed SG method provides good results measured in modularity, and its running time is acceptable as well (see Table 3, 5 and 6). In some case it gives higher modularity than other, more sophisticated methods (e.g. leading eigenvector method in Table 6). However it is strictly a clustering method, it cannot provide overlap.

There are still several ways for future work. We can define other similarity measures, or extend the modularity in some other reasonable way.

Other idea, to redefine the Equation 2, which is a constraint about the sum of the membership values of node i ($\forall 1 \leq i \leq n$). Instead of restrict this sum to be equal to 1, the sum can depend on the centrality of node i , so we can define it in the following way

$$\sum_{j=1}^c u_{ij} = g(k_i), \quad (8)$$

where $g(\cdot)$ is an increasing function of the degree. This change allows for central nodes to have higher membership values. It is a rational change, because nodes with many connections usually belongs to more than one community.

We have already tested some cases on the Zachary graph. We increased the upper bound constraints of the sum of the membership values of the 3 main node: the 0, 32 and the 33. We can obtain from our observation that the increase of the upper bounds gives overlaps and maybe gives interesting results, thus it is a possible way for future work.

Acknowledgments. The second author was partially supported by the grants TÁMOP-4.2.1/B-09/1/KONV-2010-0005, and OTKA K76099.

References

- [1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, T. Vicsek, CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics* **22**, 1021–1023 (2006).
- [2] R. Albert and A. L. Barabási, Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74** 2002.
- [3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Publishing, New York, NY, USA, 1981, ISBN 978-0306406713.
- [4] J. C. Bezdek and S. K. Pal, *Fuzzy models for pattern recognition: methods that search for structures in data*, IEEE Press, New York, NY, USA, 1992
- [5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. arXiv:0803.0476v2 [physics.soc-ph]
- [6] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski and D. Wagner, On Finding Graph Clusterings with Maximum Modularity. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)* (2007)
- [7] A. Clauset, M.E.J. Newman and C. Moore, Finding community structure in very large networks. *Physical Review E* **70**(066111) (2004)
- [8] A. Csérenszy, Gy. Kovács, M. Krész, A. Pluhár, T. Tóth, *The use of infection models in accounting and crediting, Methods for Analyzing Social and*

- Economical Processes*, International Conference, University of Szeged, November 19 – 21, 2009.
- [9] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár: *Parameter Optimization of Infection Models*, (CS)² - Conference of PhD Students in Computer Science, Szeged, Hungary, June 29 – July 2, 2010
 - [10] L. Csizmadia: *Közösségfelismerés ismeretségi gráfokban [Community detection in social graphs]*, masterthesis, University of Szeged, 2003
 - [11] L. Csizmadia, A. Bóta and A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 13th International Multiconference INFORMATION SOCIETY - IS* 2010, 393-396.
 - [12] I. Derényi, G. Palla, T. Vicsek: *Clique Percolation in Random Networks*, Physical Review Letters **94**, 160202:1–4. (2005),
 - [13] S. Fortunato: *Community detection in graphs*, Physics Reports **486**, 75–174 (2010),
 - [14] D. E. Knuth: *The Stanford GraphBase: A Platform for Combinatorial Computing*, Addison-Wesley, Reading, MA (1993)
 - [15] V. Krebs: unpublished, <http://www.orgnet.com/>
 - [16] T. Nepusz, A. Petróczy, L. Négyessy, F. Bazsó: *Fuzzy communities and the concept of bridgeness in complex networks*, Physical Review E **77** 016107 (2008),
 - [17] M.E.J. Newman: *Modularity and community structure in networks*, Proc. Natl. Acad. Sci. USA **103**, 8577–8582 (2006),
 - [18] M. Girvan, M. E. J. Newman: *Community structure in social and biological networks*, Proc. Nat. Acad. Sci. USA **99**, 7821–7826 (2002),
 - [19] M.E.J. Newman: *Finding community structure in networks using the eigenvectors of matrices*, Physical Review E **74**, 036104 (2006),
 - [20] M. E. J. Newman: *Fast algorithm for detecting community structure in networks*, Physical Review E **69**, 066133 (2004),
 - [21] V. Nicosia, G. Mangioni, V. Carchiolo and M. Malgeri: *Extending the definition of modularity to directed graphs with overlapping communities* J. Stat. Mech. (2009) P03024
 - [22] G. Palla, I. Derényi, I. Farkas and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814 (2005).
 - [23] P. Pons, M. Latapy: *Computing communities in large networks using random walks*, Springer Berlin / Heidelberg, pp. 284–293,

- [24] M. A. Porter, J.-P. Onnela and P. J. Mucha: Communities in Networks, *Notices of the American Mathematical Society* **56**, 1082–1097 & 1164–1166 (2009)
- [25] U.N. Raghavan, R. Albert, S. Kumara: *Near linear time algorithm to detect community structures in large-scale networks*, Physical Review E **76**, 036106 (2007),
- [26] J. Reichardt, S. Bornholdt: *Statistical Mechanics of Community Detection*, Physical Review E **74** (2006),
- [27] Wayne W. Zachary: *An information flow model for conflict and fission in small groups*, Journal of Anthropological Research **33**, pp. 452–473 (1977)
- [28] Network data <http://www-personal.umich.edu/~mejn/netdata/>
- [29] igraph package <http://igraph.sourceforge.net/>
- [30] graphopt algorithm <http://www.schmuhl.org/graphopt/>

Sufficient Conditions for Order-Independency in Sequential Thinning*

Péter Kardos[†]

Abstract

The main issue of this paper is to introduce some conditions for template-based sequential thinning that are capable of producing the same skeleton for a given binary image, independent of the visiting order of object points. As an example, we introduce two order-independent thinning algorithms for 2D binary images that satisfy these conditions.

Keywords: skeleton, sequential thinning, order-independency, digital topology, topology preservation

1 Introduction

Skeleton provides a reduced-dimensional representation, which describes the general shape of objects [10]. Thinning is a widely used strategy for skeletonization that is based on an iterative peeling of the object boundary [8, 11]. Several parallel and sequential alternatives have been proposed for this method. In the sequential case, an iteration step of the thinning process is usually performed in two phases. Algorithm 1 shows the “classic” sequential thinning scheme.

Basically, a “deletable” point must not be a so-called endpoint (which is important in the view of shape preservation), and its removal must be topology-preserving. A usual way to define “deletable” points is to construct a set of matching templates \mathcal{T} . In this case, an object point can be considered as deletable, if it matches at least one template $T \in \mathcal{T}$. Further on, we call such object points as \mathcal{T} -deletable. Without loss of generality, we assume square templates of size $(2k + 1) \times (2k + 1)$ ($k \in \mathbb{N}$). The *central point* of a template is then the point with position (k, k) .

As sequential thinning algorithms remove only one point at a time, topology preservation can be much easier guaranteed than in the parallel case [6, 8]. However, Algorithm 1 suffers from the problem that it may produce various skeletons for

*This work was supported by TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency.

[†]Institute of Informatics, University of Szeged, Hungary,
E-mail: pkardos@inf.u-szeged.hu

Algorithm 1:

```

repeat
  // Phase 1: contour tracking
  mark all border points
  // Phase 2: reduction
  foreach marked point  $p$  do
    if  $p$  is "deletable" in the actual image then
      delete  $p$ 
until no points are deleted

```

different visiting orders of border points. Finding order-independent strategies (i. e., algorithms that produce the same skeletons for any visiting orders) is a key problem in sequential thinning.

First Ranwez and Soille [9], then Iwanowski and Soille [3] investigated this problem. The main disadvantage of the their order-independent algorithms lies in the fact that they are basically anchor preserving reductive shrinking methods [2]. This means that, as a preprocessing step, endpoints must be previously detected as anchors. An order-independent algorithm with built-in endpoint-criterion has been proposed by Kardos, Németh, and Palágyi [4]. Their method is based on the classification of simple points. For this purpose, simple points are grouped into four sets in the first phase of an iteration, which means that this solution lies far from the sequential thinning scheme according to Algorithm 1.

In this paper we formulate some criteria which are sufficient for order-independency. The paper is organized as follows. In Section 2, we introduce some basic notions of digital topology, Section 3 presents the mentioned sufficient conditions and the proof of their correctness. Using these conditions, in Section 4, two possible 2D template sets, \mathcal{T}^1 and \mathcal{T}^2 , are proposed. In Section 5, it will be proved that if we consider \mathcal{T}^1 - or \mathcal{T}^2 -deletable points as "deletable" in Algorithm 1, we get order-independent and topologically correct thinning algorithms. Finally, some experimental results are shown in Section 6.

2 Basic Notions

Applying the basic concepts of digital topology as reviewed in [5], we introduce some additional notions for our purposes in this section.

First, we give an extension of the definition of a 2-dimensional $(8, 4)$ binary digital picture: a 2-dimensional $(8, 4)$ *labeled binary digital picture* (in the following referred to as $(8, 4)$ picture or simply as picture) can be described with the 5-tuple $(\mathbb{Z}^2, 8, 4, \mathbf{B}, \mathbf{B}^+)$, where \mathbb{Z}^2 is the set of picture points, $\mathbf{B} \subseteq \mathbb{Z}^2$ is the set of black points, its complement, $\mathbb{Z}^2 \setminus \mathbf{B}$ is the set of white points, $\mathbf{B}^+ \subseteq \mathbf{B}$ denotes the set of *active* black points, and $\mathbf{B} \setminus \mathbf{B}^+$ is the set of *inactive* black points.

A *black component* is a maximal 8-connected set of black points, while a *white*

component is defined as a maximal 4-connected set of white points.

A black point p in the picture $(\mathbb{Z}^2, 8, 4, \mathbf{B}, \mathbf{B}^+)$ is called as a *border point*, if it is 4-adjacent to at least one white point. A black point which is not a border point is said to be an *interior point*. The notation $N_k(p)$ will be used to refer to the set of points k -adjacent to p ($k \in \{4, 8\}$), and let $N_k^*(p) = N_k(p) \setminus \{p\}$. Further on, let us denote by $C(p)$ the number of 8-connected black components in the picture $(\mathbb{Z}^2, 8, 4, \mathbf{B} \cap N_8^*(p), \mathbf{B}^+)$.

A black point p is said to be a *simple point* if its deletion (i.e., changing it to white) preserves the topology of the picture [5]. We make use the following characterization of simple points.

Theorem 1. [1] *Black point p is simple if and only if p is a border point and $C(p) = 1$.*

In order to retain some relevant information about the shape of objects, thinning algorithms preserve endpoints. Hence, thinning algorithms are coupled with endpoint-characterizations. We define endpoints as follows.

Definition 1. *The black point $p \in \mathbf{B}$ in $(\mathbb{Z}^2, 8, 4, \mathbf{B}, \mathbf{B}^+)$ is an e_j -endpoint if and only if there is not any inactive point in $N_j(p)$ ($j \in \{4, 8\}$).*

Here we note that if we consider the interior points as inactive black points, then similar criteria can be discovered as “hidden” endpoint-characterizations in the thinning scheme used by Manzanera et al. [7]. The importance of the notion of inactive and active black points rests on the fact that the thinning algorithm presented in [7] is parallel, which does not alter the state of interior points during an iteration before the simultaneous removal of deletable points. However, in sequential algorithms an interior point may change to a border point right after removing a deletable point. Thus, the algorithm must “memorize” actual border points in the beginning of the given iteration in order to be able to use such endpoint-characterizations like in [7].

A background point $p \in \bar{\mathbf{B}}$ is called an *isolated cavity point* if for any $q \in N_4(p)$, $q \in \mathbf{B}$ or $N_4(q) \cap (\mathbf{B} \setminus \mathbf{B}^+) \neq \emptyset$. An object point p is called as *single border point* if $N_4(p) \cap \bar{\mathbf{B}} = \{q\}$ where $N_4(q) \cap (\mathbf{B} \setminus \mathbf{B}^+) = \emptyset$. As an example, Fig. 1 shows a possible configuration for isolated cavity points and single border points, respectively. The symbols “•”, “★”, “○” stand for active black points, inactive black points, and white points, respectively.

The reason of order-dependency in the case of sequential thinning algorithms lies mainly in the fact that there is at least one pair of simple and non-end points $\{p, q\}$ in the picture which for p is no more simple after removing q and vice versa. We call such a pair of points as decision pair. For a more formal definition, see [4]. In the case of a template-based thinning algorithm with a set \mathcal{T} of matching templates, we can restrict this definition with the necessary condition that both p and q must be \mathcal{T} -deletable.

For the masks T, T' let us suppose that T' differs from T only in one point q , where q marks a border point in T' , while it is a background point in T such that it is neither an isolated cavity point nor a 4-neighbor of a single border point in T .



Figure 1: Examples where p is an isolated cavity point (a) and a single border point (b). The positions marked “•”, “★”, and “○” are considered to be active black points, inactive black points, and white points, respectively.

Then, q is the *difference point* of T and T' , and T' is a *contour-expanded version* of T .

3 The Conditions for Order-Independency

Before we formulate our sufficient conditions for the mentioned property we must write up Algorithm 1 in a more formal way by using the introduced notions in Section 2 (see Algorithm 2). We will use the abbreviation $STA(\mathcal{T})$ (where STA stands for “Sequential Thinning Algorithm” and \mathcal{T} for the input set of matching templates) to refer to this scheme.

Algorithm 2: $STA(\mathcal{T})$

Input: picture $(Z^2, 8, 4, X, \emptyset)$ and a template set \mathcal{T}

Output: picture $(Z^2, 8, 4, Y, Y^+)$

$Y = X$

$Y^+ = \emptyset$

repeat

 // Phase 1: contour tracking

foreach p in Y **do**

if p is a border point **then** $Y^+ = Y^+ \cup \{p\}$

 // Phase 2: reduction

 changed = false

foreach $p \in Y^+$ **do**

if p is \mathcal{T} -deletable **then**

$Y = Y \setminus \{p\}$

 changed = true

until changed = false

Theorem 2. *Algorithm $STA(\mathcal{T})$ is order-independent if it fulfils both of the following conditions:*

1. *No template in \mathcal{T} contains any position that is coincident with a \mathcal{T} -deletable point (with the exception of its central element).*

II. Let q be the difference point of an arbitrary template $T \in \mathcal{T}$ and its contour-expanded version T' . Then, q is not \mathcal{T} -deletable in T' .

Proof. We give an indirect proof. Let us suppose that Conditions I and II are both satisfied, but Algorithm $STA(\mathcal{T})$ is not order-independent. Hence, one of the following two situations must occur:

Case 1: There exists an object point p in the actual image, which is \mathcal{T} -deletable in the beginning of the iteration, but it is not \mathcal{T} -deletable when it is visited.

Case 2: There exists an object point p in the actual image, which is not \mathcal{T} -deletable in the beginning of the iteration, but it is \mathcal{T} -deletable when it is visited.

Let us suppose that Case I holds. Let us consider a visiting sequence $Q = \langle s_1, s_2, \dots, s_n \rangle$ of border points, where $p = s_k$ for a given $k \in \{1, 2, \dots, n\}$. Let $S_0 = \emptyset$, $S_i = \{s_1, s_2, \dots, s_i\}$ ($1 \leq i \leq n$),

$$D_i = \{x \mid x \in S_i \text{ and } STA(\mathcal{T}) \text{ removes } x \text{ when the visiting order is } Q\},$$

and let us define the picture $\mathcal{P}_i = (Z^2, 8, 4, Y \setminus D_i, Y^+)$. Note that $D_0 = \emptyset$ and $S_i \subseteq Y^+$, hence $p \in Y^+$.

Obviously, there must be a point $s_i = q \in D_i$ ($1 \leq i < k$) such that p is \mathcal{T} -deletable in \mathcal{P}_{i-1} , but p is not \mathcal{T} -deletable in \mathcal{P}_i . As $q \in D_i$, q is also \mathcal{T} -deletable in \mathcal{P}_{i-1} . q must fall into a marked position in a template $T \in \mathcal{T}$, else the removal of q would not influence the deletability of p . This, however, leads to a contradiction with Condition I.

Therefore, only Case II can occur, which means that there must be a point $s_i = q \in D_i$ ($1 \leq i < k$) such that p is not \mathcal{T} -deletable in the picture \mathcal{P}_{i-1} , but p is \mathcal{T} -deletable in \mathcal{P}_i . Let T be the template that p matches in \mathcal{P}_i . As q falls into a marked position in T , $q \notin N_4(p)$ or p is not a single border point in \mathcal{P}_i , or else p would have been an interior point in the beginning of the given iteration of the algorithm, which is not possible, as $p \in Y^+$ in our example. Furthermore, q can not be a cavity point in \mathcal{P}_i , or else q would have been an interior point in \mathcal{P}_{i-1} , thus it would not have been visited at all. From these observations follows that q must be the difference point of T and one of its contour-expanded versions, T' . As q is \mathcal{T} -deletable in picture \mathcal{P}_i , its position yields a \mathcal{T} -deletable point in template T' , as well. However, this contradicts Condition II. \square

4 The Proposed Template Sets

Here we introduce two template sets, \mathcal{T}^1 and \mathcal{T}^2 with the help of Fig. 2, which can be used for two possible realizations of Algorithm 2.

In the 5×5 templates $T_a - T_l$ depicted in Fig. 2, each black template element “●” or “◆” coincides with an active black point, while the black template elements “★” match inactive black points. Each white template element coincides with a white point. “Don’t care” elements (i.e., empty positions) stand for points which can

be either active black points, or inactive black points, or white points. The points marked “.” can be either active black points or white points. Let us define $T_x^1 = T_x$, and $T_y^2 = T_y$ for any $x \in \{a, b, c, d, e, f, g, h, i, j\}$, $y \in \{a, b, c, d, e, f, g, h, k, l\}$, with the supplement that points marked “○” count as “don’t care” in the templates T_y^2 , but in T_x^1 , they must be either active black points or white points. Taking these assumptions into consideration, we introduce the following sets:

$$\mathcal{T}_{base}^1 = \{T_x^1 \mid x \in \{a, b, c, d, e, f, g, h, i, j\}\},$$

$$\mathcal{T}_{base}^2 = \{T_y^2 \mid y \in \{a, b, c, d, e, f, g, h, k, l\}\}.$$

Finally, the proposed template sets $\mathcal{T}^1, \mathcal{T}^2$ contain the templates of $\mathcal{T}_{base}^1, \mathcal{T}_{base}^2$, respectively (see Fig. 2), plus all their possible $k \times 90^\circ$ rotated and reflected versions ($k \in \{1, 2, 3\}$). Later, we will also refer to the set of all such possible transformed versions of a given mask T_x^1 or T_y^2 , which will be denoted by \mathcal{T}_x^1 or \mathcal{T}_y^1 , respectively.

5 Discussion

Now we will show that the algorithms $STA(\mathcal{T}^1)$ and $STA(\mathcal{T}^2)$ are both topology-preserving and order-independent. For the proof of the mentioned properties, we need to introduce some further notions. For a given object point p , let us denote by $I_k(p)$ the number of elements in $N_k(p) \cap (B \setminus B^+)$ ($k \in \{4, 8\}$). Further on, for any active black points that are 4-adjacent to points p and q , the number of elements in $N_4^*(p) \cap N_8^*(q) \cap B^+$ will be denoted by $B_p(q)$.

Let $T \in \mathcal{T}^1 \cup \mathcal{T}^2$. Let us also consider an additional 5×5 template T' being p its central point where T' has the following properties:

- i) if the cell on the position (x, y) ($x, y \in \{0, 1, 2, 3, 4\}$) marks a black/white point in T , then the cell on the position (x, y) also marks a black/white point in T' ;
- ii) if the cell on the position (x, y) ($x, y \in \{0, 1, 2, 3, 4\}$) marks a point denoted by ‘.’ in T , then the cell on the position (x, y) marks an active black point or a white point in T' ;
- iii) if the cell on the position (x, y) ($x, y \in \{0, 1, 2, 3, 4\}$) is a “don’t care” point (i.e. an empty cell) in T , then the cell on the position (x, y) refers to a black or a white point in T' ;
- iv) each active black point in a position coinciding with a member of $N_8(p)$ in T' has at least one white 4-neighbor q for which $N_4(q)$ does not contain any inactive black point in T' . (A member of $N_4(q)$ being not coincident with any position of T' can have any values.)

T' is called the *properly composed version* of T . For a better understanding of this definition, Fig. 3 shows two templates where the first of them in Fig. 3a is a properly composed version of template T_k . However, in the template of Fig. 3b, both the

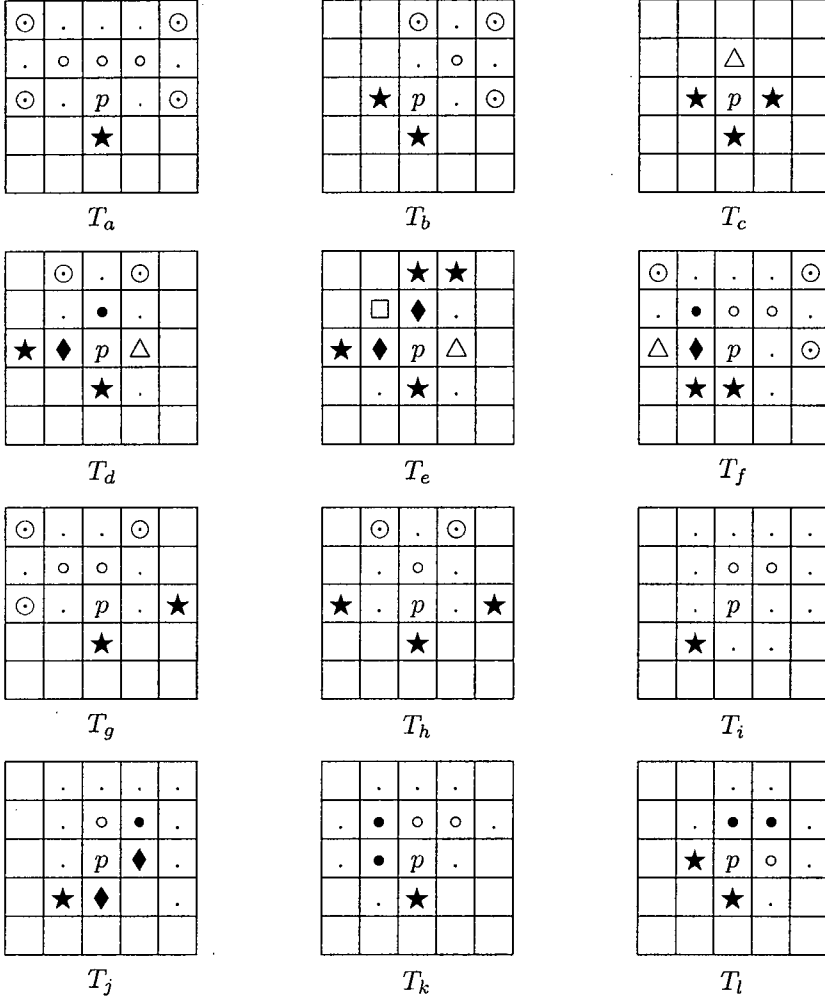


Figure 2: Sets of templates \mathcal{T}_{base}^1 and \mathcal{T}_{base}^2 . Notations: each position marked “●” and “◆” matches an active black point; positions denoted by “★” match inactive black points; each position marked “○”, “◻”, and “△” matches a white point; each “.” can yield either an active black point or a white point; each empty cell matches a “don’t care” point which can be either an (active or inactive) black point or a white point; the symbols “○” are to be considered as “.” for the members of \mathcal{T}^1 and “don’t care” for the members of \mathcal{T}^2 .



Figure 3: Examples for being (a) and not being (b) a properly composed version of the template T_k .

left and the right 4-neighbor of p are active black points, which for Condition iv) is not fulfilled.

For the proof of order-independency we will make use of an earlier result which states an important property of decision pairs.

Proposition 1. [4] *If $\{p, q\}$ is a decision pair, then $N_8(p, q)$ matches at least one of the configurations in Fig. 4 or their rotations by 90° , 180° , or 270° .*

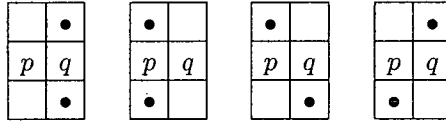


Figure 4: Possible configurations of the horizontal decision pair $\{p, q\}$. Points marked by empty cells may be either black or white.

By careful examination of the templates in \mathcal{T}^1 and \mathcal{T}^2 we can observe the following two facts.

Proposition 2. *In any $T \in \mathcal{T}^1$ there is an inactive black point (marked “★”) in $N_8(p)$.*

Proposition 3. *In any $T \in \mathcal{T}^2$ there is an inactive black point (marked “★”) in $N_4(p)$.*

Remark 1. From these propositions and from our endpoint-criterion introduced in Section 1 follows that algorithm $STA(\mathcal{T}^1)$ preserves e_8 -endpoints while $STA(\mathcal{T}^2)$ preserves e_4 -endpoints.

An important question in the view of order-independency is how the decision pairs are handled by algorithms $STA(\mathcal{T}^1)$ and $STA(\mathcal{T}^2)$. The lemma below serves as an answer.

Lemma 1. *Let $T \in \mathcal{T}^1 \cup \mathcal{T}^2$ and let T' be a properly composed version of T . If p is a member of a decision pair $\{p, q\}$ in T' , then one of the following conditions is satisfied:*

- $I_4(p) < I_4(q)$, or
- $I_4(p) = I_4(q)$ and $I_8(p) < I_8(q)$, or
- $I_4(p) = I_4(q)$, $I_8(p) = I_8(q)$, and $B_p(q) < B_q(p)$.

Remark 2. As at most one point can be removed from the decision pair $\{p, q\}$ without altering the topology, we must somehow set rules to be able to decide whether we may safely remove any point from the pair at all, and if so, then which one should be preferred. This lemma gives a possible preference for this decision, which prevails for the template sets \mathcal{T}^1 and \mathcal{T}^2 . The number of interior 4- and 8-neighbors is a useful property for the comparison of p and q , as the interior points do not change during an iteration. The values $B_p(q)$ and $B_q(p)$ can be also applied for this purpose, if it is ensured that the sets $N_4^*(p) \cap N_8^*(q) \cap B^+$ and $N_4^*(q) \cap N_8^*(p) \cap B^+$ contain only non-deletable border points. Note however that if the neighborhood of the decision pair is symmetric, then these rules do not determine any preferred point in $\{p, q\}$.

Proof. First we will show that at least one of the mentioned conditions holds for each $T \in \mathcal{T}_{base}^1 \cup \mathcal{T}_{base}^2$.

- If $T \in \{T_a^1, T_a^2, T_c^1, T_c^2\}$, then it is obvious that p is not a member of any decision pair in T' .
- If $T \in \{T_b^1, T_b^2\}$, then at least one of the positions marked “.” in $N_4(p)$ must be a white point, else p would be an interior point. Therefore, by Proposition 1, p is not a member of any decision pair in T' .
- If $T \in \{T_d^1, T_d^2\}$, then $I_4(p) = 1$ and $I_8(p) \geq 1$ in T' .
 - Let q be the left 4-neighbor of p in T' . Then, $I_4(q) \geq 1 = I_4(p)$. If $I_4(q) = 1$, then $I_8(q) = 2$ and $I_8(p) = 1$, which implies $I_8(q) > I_8(p)$.
 - Let q be the upper 4-neighbor of p in T . If $T = T_d^2$, then q is an e_8 -endpoint. Let $T = T_d^1$. q does not match the templates in \mathcal{T}_x^1 ($x \in \{a, b, c, d, e, f, g, h\}$), because $I_4(q) = 0$, and in the mentioned templates the central point has at least one inactive black 4-neighbor. By careful examination of the possible remaining templates it is also easy to see that q does not match any member of \mathcal{T}_i^1 and \mathcal{T}_j^1 . Thus, q is neither \mathcal{T}^1 -deletable nor \mathcal{T}^2 -deletable, hence $\{p, q\}$ cannot be a decision pair.
- If $T \in \{T_e^1, T_e^2\}$, then $I_4(p) = I_8(p) = 1$ in T' . Let q be the left or upper 4-neighbor of p in T' . Then, in both possible cases we get $I_4(q) = 1 = I_4(p)$, $I_8(q) \geq 2 > I_8(p)$.
- If $T \in \{T_f^1, T_f^2\}$, then $I_4(p) = 1, I_8(p) = 2$ in T' . Let q be the left 4-neighbor of p . By Proposition 1, only the set $\{p, q\}$ may be a decision pair in T' , and $I_4(q) = I_4(p) = 1$, $I_8(q) \geq 2 = I_8(p)$, further on, $B_p(q) = 0, B_q(p) = 1$, hence $B_p(q) < B_q(p)$.

- If $T \in \{T_g^1, T_g^2\}$, then let q be the right 4-neighbor of p and r be the left 4-neighbor of p . By Proposition 1, the set $\{p, r\}$ cannot be a decision pair in T' . Let us suppose that $\{p, q\}$ is a decision pair. In this case, q is a border point, and by Proposition 1, the upper 4-neighbor of q must be also a border point. Therefore, there is not any white point s in $N_4(q)$ which for $N_4(s)$ would contain any inactive black point. But this leads to a contradiction with the definition of the properly composed version of T . Hence, $\{p, q\}$ cannot be a decision pair in T' .
- If $T \in \{T_h^1, T_h^2\}$, we can show the same way as in the previous case that p can not be a member of a decision pair in T' .
- If $T = T_i^1$, then $I_4(p) = 0$ and by Proposition 1, p can only be a member of a decision pair $\{p, q\}$ in T' where q is the left or bottom 4-neighbor of p . Then, $I_4(q) \geq 1 > I_4(p)$.
- If $T = T_j^1$, then $I_4(p) = 0$ and the following two cases are to be examined.
 - Let q be the right 4-neighbor of p in T' . If the bottom 4-neighbor of q , say r , is not an inactive black point, then q is an e_8 -endpoint, which means, $\{p, q\}$ cannot be a decision pair. If r is an inactive black point, then $I_4(q) = 1 > I_4(p)$.
 - Let q be the bottom 4-neighbor of p in T' . Then, $I_4(q) \geq 1 > I_4(p)$.
- If $T \in \{T_k^2, T_l^2\}$, then it is easy to see by Propositions 1 and 3 that p cannot be a member of a decision pair in T' .

It is obvious that the amounts $I_4(p)$, $I_8(p)$, $B_p(q)$, $B_q(p)$ will not change for any $q \in N_4(p)$ after rotating or reflecting a template. Therefore, the lemma also holds in the case $T \in \mathcal{T}_x^1 \cup \mathcal{T}_y^2$, for any $x \in \{a, b, c, d, e, f, g, h, i, j\}$ and $y \in \{a, b, c, d, e, f, g, h, k, l\}$. \square

Finally, using Lemma 1 and Propositions 2-3, we give a proof for the mentioned properties of the proposed algorithms.

Theorem 3. *Algorithms $STA(\mathcal{T}^1)$ and $STA(\mathcal{T}^2)$ are both topology-preserving.*

Proof. It is easy to see that if an object point p matches in any iteration of $STA(\mathcal{T}^1)$ or $STA(\mathcal{T}^2)$ a $T \in \mathcal{T}^1$ or a $T \in \mathcal{T}^2$, then there must be a properly composed version T' of T which for p matches T' as well. Therefore, by Theorem 1 we have only to show that $C(p) = 1$ holds in such a T' . It is sufficient to prove this only for the case $T \in \mathcal{T}_{base}^1 \cup \mathcal{T}_{base}^2$, because $C(p)$ does not change after rotating or reflecting T .

- Let $T \in \{\mathcal{T}_x^1, \mathcal{T}_y^2\}$ where $x \in \{a, b, c, d, e, f, j\}$ and $y \in \{a, b, c, d, e, f, k, l\}$. By careful examination of these templates one can notice that $C(p) = 1$ in T' , hence p is simple.

- Let $T \in \{\mathcal{T}_g^1, \mathcal{T}_g^2\}$. It is easy to see that in the beginning of the actual iteration of $STA(\mathcal{T}^1)$ or $STA(\mathcal{T}^2)$, $C(p) = 1$ in T' , and the only way that this could change is when the right 4-neighbor of p , say q , and the upper 4-neighbor of q are both black points and q gets deleted before visiting p in Phase 2. But in this case, we would come into a contradiction with the definition of T' , as q would be an active black point which for $N_4(q)$ does not contain any white point without an inactive black 4-neighbor. Therefore, $C(p) = 1$ still holds in T' when p becomes the actual point in Phase 2, which means that p is simple.
- The proof for the situation $T \in \{\mathcal{T}_h^1, \mathcal{T}_h^2\}$ can be similar to the previous case.
- Let $T = \mathcal{T}_i^1$. It is easy to see that in the beginning of the actual iteration of $STA(\mathcal{T}^1)$, $C(p) = 1$ holds in T' , and there are two possible situations when this could change: either both the left 4-neighbor of p , say q , and the upper-left 4-neighbor of p are black points and q will be removed before visiting p , or the bottom 4-neighbor, say r , and the bottom-right 4-neighbor of p are black points and r will be removed before visiting p . We only give the proof for the first case as the other situation can be similarly examined. Let us suppose that q gets removed before visiting p . Thus, when algorithm $STA(\mathcal{T}^1)$ visits q , it matches a properly composed version U' of a $U \in \mathcal{T}^1$. It is obvious that $U \neq \mathcal{T}_i^1$ and $U \notin \mathcal{T}_i^1$, hence $U = \mathcal{T}_x^1$ or $U \in \mathcal{T}_x^1$ must hold for at least one $x \in \{a, b, c, d, e, f, j\}$. Above, we have already seen that in such a case, the central point of U' is a simple point. It can be also easily seen that by removing p before q , q would not be simple any more. This means that $\{p, q\}$ is a decision pair, and from Lemma 1 follows that in this situation, q will not be removed. Hence, p remains simple until it gets removed.

□

Theorem 4. *Algorithms $STA(\mathcal{T}^1)$ and $STA(\mathcal{T}^2)$ are both order-independent.*

Proof. For the positions represented by white or black symbols in any template $T \in \mathcal{T}^1 \cup \mathcal{T}^2$ the following observations can be made.

- Points marked by “o” do not have any inactive black 8-neighbor/4-neighbor, therefore if we consider such a point q and a contour-expanded version T' of $T \in \mathcal{T}^1/T \in \mathcal{T}^2$ with difference point q , then by Proposition 2 / Proposition 3, q is not \mathcal{T}^1 -deletable/ \mathcal{T}^2 -deletable in T' .
- Points marked by “•” do not have any inactive black 8-neighbor/4-neighbor, therefore, by Proposition 2 / Proposition 3, such a point is not \mathcal{T}^1 -deletable/ \mathcal{T}^2 -deletable in T .
- It is easy to see that if a position denoted by “♦” marks a simple point q and $N_8(q)$ contains an inactive black point, then by Lemma 1, q does not match any $T' \in \mathcal{T}^1 \cup \mathcal{T}^2$.

- Points marked by “ \triangle ” are 4-neighbors of a simple point, therefore we can not construct for such a point q any contour-expanded version T' of T with difference point q .
- The point represented by the symbol “ \square ” is an isolated cavity point, therefore we can not construct for such a point q any contour-expanded version T' of T with difference point q .

From these observations and from Theorem 2 follows that algorithms $STA(\mathcal{T}^1)$ and $STA(\mathcal{T}^2)$ are order-independent. \square

6 Results

In experiments our algorithms were tested on some test pictures. Our results were compared to the ones produced by the existing algorithm introduced by Ranwez and Soille [9]. Figs. 5-9 show some illustrative examples where “skeletons” extracted by the mentioned 2D algorithms are superimposed on the original objects. Numbers in parentheses indicate the counts of skeletal points. One can easily recognize that the method by Ranwez and Soille has extracted much more unwanted line segments for the selected pictures than the proposed algorithms. However, it is important to note that the aim of the paper is not to carry out a detailed comparison of order-independent sequential thinning methods, hence the shown results serve only demonstrational purposes.

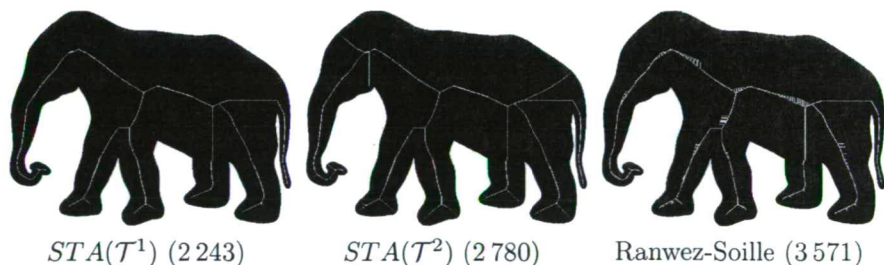


Figure 5: A 612×467 image with 179 293 object points of an elephant and its “skeletons”.

References

- [1] Hall, R. Parallel connectivity-preserving thinning algorithms. In Kong, T. Y. and Rosenfeld, A., editors, *Topological Algorithms for Digital Image Processing*, pages 145–179, Elsevier Science B. V., 1996.
- [2] Hall, R.W., Kong, T.Y., Rosenfeld, A. Shrinking binary images. In Kong, T. Y. and Rosenfeld, A., editors, *Topological Algorithms for Digital Image Processing*, pages 31–98, Elsevier Science B. V., 1996.



Figure 6: A 530×530 image of a seahorse with 79 293 object points and its “skeletons”.

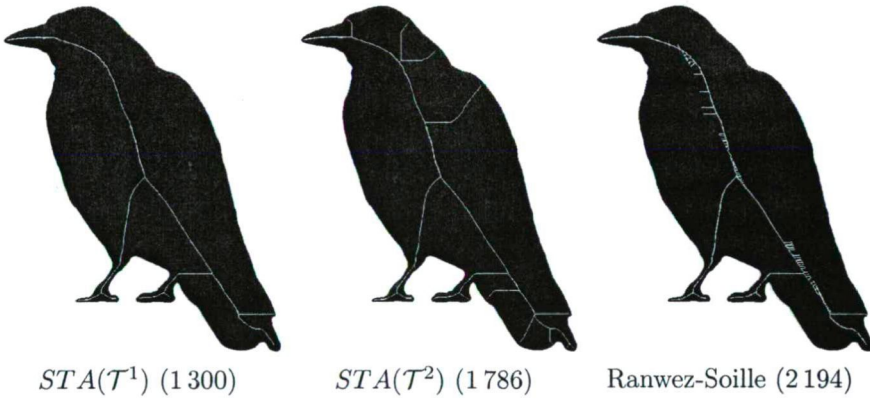


Figure 7: A 492×606 image of a crow with 126 538 object points and its “skeletons”.



Figure 8: A 1600×1600 image of a plane with 487 620 object points and its “skeletons”.



Figure 9: A 745×773 image with 152 611 object points of a leaf and its “skeletons”.

- [3] Iwanowski, M., Soille, P. Order independence in binary 2D homotopic thinning. In Kuba, A., Nyúl, L., Palágyi, K., editors, *DGCI 2006, LNCS 4245*, pages 592–604, Heidelberg, 2006, Springer.
- [4] Kardos, P., Németh, G., and Palágyi, K. An order-independent sequential thinning algorithm. In Wiederhold, P. and Barneva, R. P., editors, *IWCIA 2009, LNCS 5852*, pages 162–175, Berlin, 2009, Springer-Verlag.
- [5] Kong, T. Y. and Rosenfeld, A. Digital Topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48:357–393, 1989.
- [6] Kwok, P.C.K. A thinning algorithm by contour generation. *Communications of the ACM*, 31:1314–1324, 1988.
- [7] Manzanera, A., Bertrand, T. M., Prêteux, F., and Longuet, B. nD skeletonization: a unified mathematical framework. *Journal of Electronic Imaging*, 11:25–37, 2002.
- [8] Lam, L., Lee, S.-W., and Suen, C. Y. Thinning methodologies – a comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14:869–885, 1992.
- [9] Ranwez, V., Soille, P. Order independent homotopic thinning for binary and grey tone anchored skeletons. *Pattern Recognition Letters*, 23:687–702, 2002.
- [10] Siddiqi, K. and Pizer, S. M., editors. *Medial Representations — Mathematics, Algorithms, and Applications*, Series in Computational Imaging, 2008, Springer.
- [11] Suen, C. Y. and Wang, P. S. P., editors. *Thinning Methodologies for Pattern Recognition*. Series in Machine Perception and Artificial Intelligence (8), World Scientific, 1994.

Employing Pythagorean Hodograph Curves for Artistic Patterns

Gergely Klár* and Gábor Valasek*

Abstract

In this paper we present a novel design element creator tool for the digital artist. The purpose of our tool is to support the creation of vines, swirls, swooshes and floral components. To create visually pleasing and gentle curves we employ Pythagorean Hodograph quintic spiral curves to join a hierarchy of *control circles* defined by the user. The control circles are joined by spiral segments with at least G^2 continuity, ensuring smooth and seamless transitions. The control circles give the user a fast and intuitive way to define the desired curve. The resulting curves can be exported as cubic Bézier curves for further use in vector graphics applications.

1 Introduction

Floral elements, vines, tangled spirals and similar features are among the most popular design components. These components could be found in traditional ornamental and contemporary abstract designs as well. Whatever the actual media or purpose is, the common property of these patterns is they consist of an intricate network of gentle curves. Creating such curves by hand is a non-trivial task. Users of vector graphic applications often employ spirals as a starting shape, then apply transformations to achieve the desired curve. This method of curve creation heavily restricts the set of producible shapes. The motivation for our work was to enlarge this set of curves available for the artist, while guaranteeing the *pleasingness* of the curves.

To give a formal definition of pleasing curves, we built upon the empirical observation and the same assumption as Xu and Mould, that is, a pleasing curve is a curve of monotonically changing curvature. Our tool enables the digital artists to create such curve or sequence of curves with ease. Our method uses cubic and quintic splines to generate resolution independent curves, which can be used directly, serve as a skeleton of a design, or act as a path for strokes or objects. To support the widest range of third party tools possible, the generated curves can be exported as cubic Bézier splines.

*Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary,
E-mail: g.klar@creativereboot.hu, valasek@inf.elte.hu

To ensure this property of the drawn curves, we employ Pythagorean Hodograph (PH) curves. PH curves introduced by Farouki and Sakkalis [5] have very favourable properties, most importantly it is possible to define spiral segments using PH quintic curves whose curvature changes monotonically with arc-length. Spiral curves have been used in highway, railway and robot trajectory design [1], [7], [9]. Walton et. al. proposed to use PH spirals for such applications [12], which was followed by further research on PH quintic spirals [4] [6]. Now we demonstrate the efficiency of these curves as design elements as well.

2 Background

There have been several efforts to automate aspects of the artistic process of creating such designs and ornaments. Since the fundamental work of Prusinkiewicz and Lindenmayer [10] L-systems has been used widely to generate flowers and flower like patterns.

Wong et al. [8] presented a system to automatically generate space-filling floral ornaments. Their system uses proxy objects during generation what could be replaced by arbitrary elements created by any means.

Xu and Mould's *Magnetic Curves* [14] are more closely related to our work. They focus on the creation of the curves themselves, but their method uses a discrete time-step approach, and they commit the problem of creating smooth curves to approximation routines.

In our tool the user defines the curves by placing circles on the canvas. Using various intuitive defining elements such as this has been subject to research [2], for example Singh proposed a method in which ellipse arcs are used for defining the curves [11].

3 Pythagorean Hodograph curves

Let $x(t)$ and $y(t)$ be the x and y components of the parametric curve $\mathbf{Q}(t) : [0, 1] \rightarrow \mathbb{R}^2$. $\mathbf{Q}(t)$ is said to be a PH curve if there exists a polynomial $\sigma(t)$ such that $x'^2(t) + y'^2(t) \equiv \sigma^2(t)$, that is, its coordinate polynomials' derivatives $x'(t), y'(t)$ form a Pythagorean-triple with the parametric speed $\sigma(t)$. This holds if and only if

$$\begin{aligned} x'(t) &= (u^2(t) + v^2(t))w(t) \\ y'(t) &= 2u(t)v(t)w(t), \end{aligned}$$

where $u(t)$, $v(t)$, and $w(t)$ are polynomials and $u(t)$ and $v(t)$ are relative prime [5]. We set $w(t) \equiv 1$, that is we use primitive Pythagorean Hodograph curves, following Walton and Meek's work as in [12].

One of the appealing properties of PH curves is their curvature can be expressed

as a rational function

$$\kappa(t) = \frac{2(u(t)v'(t) - u'(t)v(t))}{(u^2(t) + v^2(t))^2}.$$

The polynomials $u(t)$ and $v(t)$ are defined to be quadratic, hence there are six parameters (six scalar degrees of freedom) for the final curve. Using quadratic $u(t)$ and $v(t)$ results in quintic $Q(t)$ what has desirable properties for our needs. This $Q(t)$ then can be expressed in the Bézier form

$$Q(t) = \sum_{i=0}^5 \binom{5}{i} P_i (1-t)^{5-i} t^i \quad t \in [0, 1],$$

where P_0 is the beginning point of the curve and $P_i, i \in \{1, 2, 3, 4\}$ are functions of the aforementioned six parameters. For the exact definition see [12].

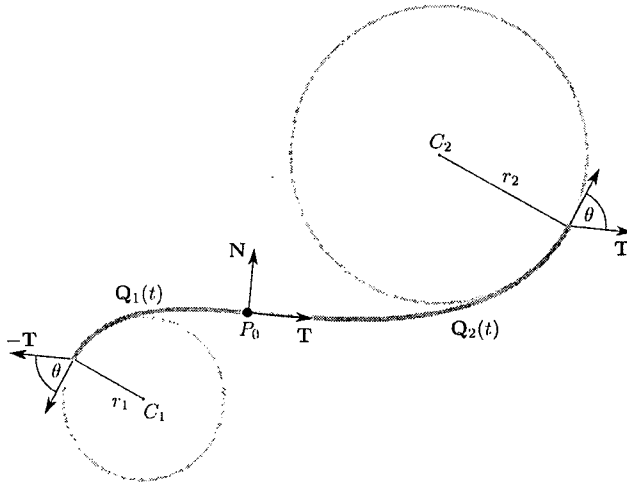


Figure 1: Definition of points and angles as used by Walton and Meek.

Without the loss of generality, we can consider the case where the curve starts from the origin and the x axis is tangential to it, as detailed in Habib and Sakai [6]. The user defined values for the actual starting point and tangent for the curve then can be translated, rotated, and if needed mirrored to align with this special case. The control points are computed in this space, then they are transformed back to the original using the inverse transformations of the applied ones.

We use these curves to define spiral segments with monotonically changing curvature. These spiral segments are used to create our circle-in-circle and circle-inside-circle transitions as detailed in the following section.

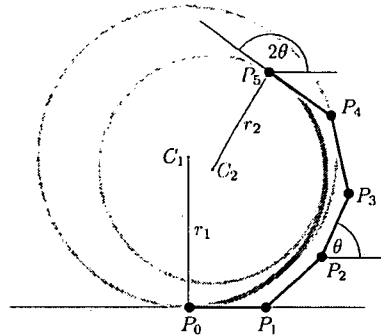


Figure 2: Definition of points and angles as used by Habib and Sakai.

4 Control circle hierarchies

To create the desired pattern, in our tool the user defines curves through a hierarchy of *control circles*. These control circles define the constraints required to derive a gentle curve. The control circles form a directed tree, where each circle is potentially connected to its ancestor. These connections are automatically created by our system as an appropriate curve.

Using the control circle hierarchy we create smooth transition curves between each child and the ancestor circle. Depending on their geometric relation we consider two cases: the *circle-to-circle transition curve* between two non-touching and non-overlapping circles, and the *circle-in-circle transition curve*, when the descendant circle is fully contained within its ancestor. For the circle-to-circle case we only support S-shaped transition curves, i.e. curves with an inflection point in the middle.

To realize these curves, the previously introduced PH curves are used. These curves have been defined to have G^2 contact to their control circle. Therefore, each incoming curve can have G^2 contact with any outgoing curve of the same circle if their endpoints coincide.

The typical workflow of curve creation in our prototype application using control circles is as follows:

1. The user places an initial circle on the drawing canvas or selects an existing one.
2. Adjusts the position and radius of the circle as needed.
3. Creates a new control circle anywhere on the canvas, and sets its radius.
4. The new circle is created as a child of the previously selected one.
5. The circles define the starting and ending curvature of the curve.

6. If it is possible, a new transition curve is created by our tool, the curve will *coil* onto the control circles, touching them only at the endpoints with G^2 contact.

4.1 Circle-to-circle transitions

Two non-touching, non-overlapping control circles are connected by an S-shape, if distance centres are within a certain threshold. The derivation of the control points of the S-curves is following the work of Walton and Meek [13].

According to their work, if the radii and centres of the circles are $r_1, r_2, \mathbf{C}_1, \mathbf{C}_2$ respectively, then the condition for the distance threshold is

$$r_1 + r_2 < \|\mathbf{C}_2 - \mathbf{C}_1\| < \sqrt{\left(\frac{321}{120}\right)^2 + \left(\frac{91}{60}\right)^2} (r_1 + r_2) < 3.075(r_1 + r_2).$$

If this criteria is met, the transition curve is automatically created.

The positions and radii of the circles define the shape of the S, save for reflectional symmetry. The choice between the two possible curves is the only required additional user input.

The S-shape is produced using two distinct PH spiral segments. Let $\mathbf{Q}_1(t), \mathbf{Q}_2(t)$ denote these two curves. We derive both $\mathbf{Q}_1(t)$ and $\mathbf{Q}_2(t)$ so $\mathbf{Q}_1(0) = \mathbf{Q}_2(0)$, $\mathbf{Q}_1(1), \mathbf{Q}_2(1)$ lie on the first and second circle respectively, and $\kappa_1(0) = \kappa_2(0)$, $\kappa_1(1) = \frac{1}{r_1}$, $\kappa_2(1) = \frac{1}{r_2}$, where $\kappa_1(t), \kappa_2(t)$ denotes the curvature of $\mathbf{Q}_1(t), \mathbf{Q}_2(t)$ respectively.

4.2 Circle-in-circle transitions

A fully contained circle is joined to its ancestor with a spiral segment, if such transition is possible. The conditions and the derivation of the control points are given in Habib and Sakai's [6] work.

For a given containing circle with radius and centre r_1, \mathbf{C}_1 , only the radius r_2 of the contained circle is required to compute the control points of the new spiral segment. Using the radii of the circles the algorithm first computes the range of allowed distances between the centres. From this range our software chooses the smallest possible distance. While it would be possible to introduce this as a user parameter, practice show that this does not expand the range of possible curves significantly.

As a consequence of the derivation, from the parameters that are defined by the user the ones used are only r_1, r_2, \mathbf{C}_1 , but not \mathbf{C}_2 . The actual value of $\|\mathbf{C}_1 - \mathbf{C}_2\|$ is chosen by the software, and the final \mathbf{C}_2 centre is computed at a later point. For a given pair of radii and a distance, the spiral segment is uniquely defined, thus it is the radii and centre-to-centre distance that dictates the positioning of the smaller circle. If the creation of the transition curve is possible, only the radius of the smaller circle is used from the user input, and its centre is repositioned as defined by the algorithm. Similarly to the S-curves, a mirroring property can be defined.

An additional parameter of a spiral segment is its starting point, defined in degrees on the ancestor control circle. For new segments, this is automatically calculated, so the new curve continues the ancestor's incoming curve, if it exists.

5 Implementation

The computations of the control points in both cases are quite involved, especially for the case of *circle-in-circle* transitions, and require numerical algorithms in the implementation. Since the solution of the emergent equations differ greatly the two kinds of transition curves, we discuss them separately.

5.1 Circle-to-circle transitions

For this special case of *circle-to-circle* transitions, all the six parameter of $u(t)$ and $v(t)$ can be reduced either to zero or to the function of a single parameter θ for each PH spiral segment. This θ measures the angle between the beginning and ending tangent vectors of a spiral segment, as show on Figure 1. Because of this, the problem of finding the appropriate curve is reduced to finding an appropriate value of θ .

Unfortunately, there is no known explicit equation for θ , but it defined as the root of a rational trigonometric equation $f(\theta) = 0$. Citing [12] $f(\theta)$ is defined as

$$f(\theta) = \|C_2 - C_1\|^2 - \left(\frac{r_1 + r_2}{120}\right)^2 g_1^2(\theta) - \left(\frac{r_1 + r_2}{60}\right)^2 g_2^2(\theta),$$

where

$$g_1(\theta) = \frac{\sin(\theta)}{(1 + \cos(\theta))^2} \left(321 - 58 \cos(\theta) + (-36) \cos(\theta)^2 \right),$$

$$g_2(\theta) = \frac{1}{1 + \cos(\theta)} \left(91 + 11 \cos(\theta) + 18 \cos(\theta)^2 \right).$$

The first and second derivatives of g_1 and g_2 :

$$g_1'(\theta) = -\frac{36 \cos(\theta)^3 + 72 \cos(\theta)^2 + 365 \cos(\theta) - 700}{\cos(\theta)^2 + 2 \cos(\theta) + 1}$$

$$g_1''(\theta) = \frac{\left(36 \cos(\theta)^3 + 108 \cos(\theta)^2 - 221 \cos(\theta) + 1765 \right) \sin(\theta)}{\cos(\theta)^3 + 3 \cos(\theta)^2 + 3 \cos(\theta) + 1}$$

$$g_2'(\theta) = -\frac{\left(18 \cos(\theta)^2 + 36 \cos(\theta) - 80 \right) \sin(\theta)}{\cos(\theta)^2 + 2 \cos(\theta) + 1}$$

$$g_2''(\theta) = -\frac{18 \cos(\theta)^3 + 36 \cos(\theta)^2 + 116 \cos(\theta) - 196}{\cos(\theta)^2 + 2 \cos(\theta) + 1}$$

Using these we numerically find the root of $f(\theta) = 0$ with bisection. It is sufficient, since these computations are only required when parameters of the defining circles are modified.

5.2 Circle-in-circle transitions

When computing the control points for a *circle-in-circle* transition we built upon the work of Habib and Sakai [6]. The problem boils down to finding a specific θ value like in the previous case, but here it has a slightly different meaning. This time it is 2θ that measures the angle between the beginning and ending tangent vectors of a spiral segment. Figure 2 demonstrates θ and 2θ .

Before computing θ , a valid range for $\|\mathbf{C}_1 - \mathbf{C}_2\|$ has to be determined. Again the solution required a root solving technique. (The upper bound is simply $|\mathbf{C}_1 - \mathbf{C}_2|$.)

The equation what's root is to be found is

$$m(\lambda)(r_1 - r_2) - \|\mathbf{C}_1 - \mathbf{C}_2\| = 0,$$

where

$$\begin{aligned} m(\lambda) &= \frac{\sqrt{g_1(p_0)^2 + g_2(p_0)^2}}{\lambda^3 - 1}, \\ g_1(p) &= \frac{1}{120 \lambda^6 \sin\left(\frac{\theta}{2}\right)^2} \left\{ \lambda^2 + p(p^2 - 2\lambda^4(3 - 2\lambda + 6\lambda^2)) \cos(\theta) \right\}, \\ g_2(p) &= \frac{1}{60 \lambda^6 \sin\left(\frac{\theta}{2}\right)} \left\{ 3\lambda^2(p^2(1 + \lambda) - 20\lambda^7 \right. \\ &\quad + 2p^2 \cos(\theta) + 20\lambda^4 \cos(2\theta)) \sin\left(\frac{\theta}{2}\right) \\ &\quad + p \left((p^2 + 2\lambda^5) \cos\left(\frac{\theta}{2}\right) + 2\lambda^4(3 - \lambda) \cos\left(\frac{3\theta}{2}\right) \right. \\ &\quad \left. \left. - 6\lambda^4 \cos\left(\frac{5\theta}{2}\right) \right) \right\}, \\ p_0 &= \frac{3}{2} \lambda^2 \left(1 + \sqrt{1 + \frac{16\lambda \cos\left(\frac{\theta}{2}\right)^2}{9}} \right) \tan\left(\frac{\theta}{2}\right), \\ \lambda &= \sqrt[3]{\frac{r_1}{r_2}}. \end{aligned}$$

These definitions were given by Habib and Sakai. As it is clear from the equations above, the derivative of the concerned functions are too complex, and it would be impossible to work with it efficiently. Nevertheless, because the domain of the possible solutions is known, a simple bisection method has proven adequate.

6 Export and refinement

The patterns created in the tool can be exported as sequences of cubic Bézier curves in SVG format (the highest degree Bézier curves that SVG supports). In order to reduce the difference between the original PH curves and the Bézier curves as much as possible, each PH curve is exported as two Bézier curves. Since a quintic PH curve is a special Bézier curve, standard operations can be carried out on it. At first we split the PH curve to two quintic Bézier curve at $t = 0.5$, then reduce the degree of these curves to cubic. The resulting curve undergoes just small visual distortion.

The SVG format is widely recognized by application, thus the user can use the whole toolset of her software to apply colours, strokes, brushes or textures the curves. Figure 3 demonstrates such a process.

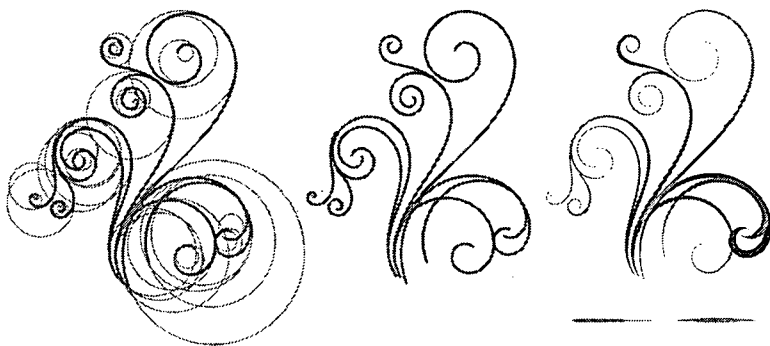


Figure 3: Refinement process example, from left to right: SVG file opened with Inkscape, Bézier segments connected, brushes and colours applied according to curve parameter (brushes shown below).

7 Results

In this paper we demonstrated a novel use of Pythagorean Hodograph curves. We presented a tool in which the user can create smooth patterns using out *control circle hierarchies*. Using this tool the user can effortlessly create curves with pleasing slopes and seamless connection thanks to the monotonically changing curvature of the elements and to the G^2 contacts. We aim this tool to the digital artist for creation of floral ornaments, swirls, and swooshes.

Along with these, we augmented the works of principal researchers of PH curves with implementation details.

Finally we showed how our tool integrates with other vector graphics applications.

8 Future work

While control circles give an intuitive way to define the curves, they lack the precision that is sometimes required even by the artist. It is not possible to explicitly define the start or endpoint of a curve, and they can be fitted only using experimenting. This behaviour of the control circles arises from the formulation of the PH curves, where only the starting point could be defined, and the position of the endpoint is dictated by the value of θ . Being able to directly define the endpoints and curvatures, much like the cubic Bézier curves being defined by endpoints and tangents in drawing applications, would give the user much better control.

Another aspect of our future work is fully integrating our tool to a vector graphics application, such as Inkscape. This way our technique would be able to reach a wider audience, and we could receive feedback from a real user base.

Further appealing properties of the PH curves can be taken advantage of. For example the arc length of PH curves can be expressed as a polynomial, allowing the application of textures and brushes depending on arc length, not just on curve parameter.

9 Acknowledgements

This research was supported by the project TÁMOP-4.2.1/B-09/1/KMR-2010-003 of Eötvös Loránd University.

References

- [1] Baass, K. G. The use of clothoid templates in highway design. *Transport. Forum* 1, 1:47–52, 1984.
- [2] Baudel, T. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, pages 185–192, New York, NY, USA, 1994. ACM.
- [3] Farin, G. *Curves and surfaces for computer aided geometric design (3rd ed.): a practical guide*. Academic Press Professional, Inc., San Diego, CA, USA, 1993.
- [4] Farouki, R. T. Pythagorean-hodograph quintic transition curves of monotone curvature. *Computer-Aided Design*, 29:601–606(6), September 1997.
- [5] Farouki, R. T. and Sakkalis, T. Pythagorean hodographs. *IBM J. Res. Dev.*, 34(5):736–752, 1990.
- [6] Habib, Z. and Sakai, M. On ph quintic spirals joining two circles with one circle inside the other. *Computer-Aided Design*, 39(2):125 – 132, 2007.

- [7] Hartman, P. The highway spiral for combining curves of different radii. *Trans. Amer. Soc. Civil Engineers*, pages 389–409, 1957., 122.
- [8] M. T. Wong, D. E. Zongker. and Salesin, D. H. Computer-generated floral ornament. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 423–434, New York, NY, USA, 1998. ACM.
- [9] Meek, D. S. and Walton, D. J. The use of cornu spirals in drawing planar curves of controlled curvature. *J. Comput. Appl. Math.*, 25:69–78, January 1989.
- [10] Prusinkiewicz, P. and Lindenmayer, A. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.
- [11] Singh, K. Interactive curve design using digital french curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 23–30, New York, NY, USA, 1999. ACM.
- [12] Walton, D. J. and Meek, D. S. A pythagorean hodograph quintic spiral. *Computer-Aided Design*, 28(12):943 – 950, 1996.
- [13] Walton, D. J. and Meek, D. S. Planar G2 transition with a fair pythagorean hodograph quintic curve. *Journal of Computational and Applied Mathematics*, 138(1):109 – 126, 2002.
- [14] Xu, L. and Mould, D. Magnetic Curves: Curvature-Controlled Aesthetic Curves Using Magnetic Fields. pages 1–8, Victoria, British Columbia, Canada, 2009. Eurographics Association.

Calculating Non-Equidistant Discretizations Generated by Blaschke Products*

Levente Lócsi[†]

Abstract

The argument functions of Blaschke products provide a very elegant way of handling non-uniformity of discretizations. In this paper we analyse the efficiency of numerical methods as the bisection method and Newton's method in the case of calculating non-equidistant discretizations generated by Blaschke products.

By taking advantage of the strictly increasing property of argument functions we may calculate the discrete points in an enhanced order—to be introduced here. The efficiency of the discrete points' sequential calculation in this order is significantly increased compared to the naive implementation.

In our research we are primarily motivated by ECG curves which usually have alternating regions of high or low variability, and therefore different degree of discretization is needed at different regions of the signals.

Keywords: non-equidistant discretization, Blaschke products

1 Introduction

In many cases non-equidistant discretizations (or non-uniform divisions) have been proven very useful. Many examples can be found from the fields of computer graphics (e.g. NURBS curves) to FFT analysis by engineering sciences.

In our research we are investigating a very elegant way of handling non-uniformity in the case of signals (e.g. ECG signals) with regions of high variability and therefore more detail, dense discretization needed, and with constant-like regions where less detail, sparse discretization is enough. The Blaschke functions, Blaschke products and their associated argument functions are used to describe a suitable non-equidistant discretization. The inverse image of an equidistant discretization according to an argument function is considered.

One can give an explicit form of the inverse of an argument function associated to a Blaschke function: the inverse can be simply calculated. But in the case

*This work was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

[†]Eötvös Loránd University, Faculty of Informatics, Department of Numerical Analysis, H117 Budapest, Pázmány Péter sétány 1/C. E-mail: locsi@inf.elte.hu

of Blaschke products, the inverse of the argument function has no explicit form, numerical methods are needed to solve the arising non-linear equations. We have as many equations as the number of points in the discretization to generate.

In the work to be presented here we analyse the efficiency of methods like the well-known bisection method and Newton's method applied to this problem. By taking advantage of non-uniformity and the strictly increasing property of argument functions, we may solve the equations at hand in a clever order also to be explained. The advantages and disadvantages of these methods and their combinations are to be analysed.

In Section 2 we introduce the argument functions (associated to Blaschke functions and products) as they serve as the starting point for the research presented here. Then we show how we can define a non-equidistant discretization (NED) according to an argument function.

Then in Section 3 we describe the methods that can be used to calculate a NED, analyse their advantages and disadvantages and introduce the idea of a better order to calculate the discrete points. Finally in Section 4 we present our results concerning calculation step count and time.

A section exhibiting possible further research areas, and the Summary concludes this paper.

2 Non-equidistant discretizations

In this section we describe the Blaschke functions and products, define the argument function of those and show how a NED can be defined. We are focusing mainly on the properties of the argument function, therefore we do not provide a detailed description of Blaschke functions. For a proper definition, further analysis and detailed calculations see e.g. [1, 3].

Blaschke functions are a family of complex valued functions of a complex variable with one additional complex parameter¹ inside the complex unit disk, i.e.:

$$B_a: \mathbb{C} \rightarrow \mathbb{C}, \quad (a = r \cdot e^{i\varphi}, 0 \leq r < 1, \varphi \in \mathbb{R}),$$

and they are defined by the formula:

$$B_a(z) = \frac{z - a}{1 - \bar{a}z} \quad (z \in \mathbb{C} \setminus \{1/\bar{a}\}).$$

These functions have the following property which we will use later on: they are a continuous bijection from the unit circle onto itself.

Blaschke products are essentially products of Blaschke functions.

Note some basic properties of Blaschke functions. The function B_a has a zero inside the unit circle ($B_a(a) = 0$), and it has a pole of order one outside the unit circle in the point $1/\bar{a}$.

¹A second parameter may also be introduced, but is of no importance in this context.

Blaschke functions and products have many interesting properties and wide ranges of mathematical application. E.g. the isometric transformations on the Poincaré disk model of hyperbolic geometry can be written in terms of Blaschke functions; they can describe rational orthogonal systems²; UDMD systems can be formed with them, which allows the application of FFT-like algorithms; they form a group with respect to the composition of functions, so wavelet-like transforms can be defined related to them; etc. See [4, 5, 6, 7].

From the bijection property of the function B_a comes that to every $t \in \mathbb{R}$ we can assign a unique $u \in [-\pi, \pi]$ so that $\exp iu = B_a(\exp it)$. This leads us to the definition of the *argument function* associated to a Blaschke function:

$$\beta_a: \mathbb{R} \rightarrow \mathbb{R}, \quad \beta_a(t) = \arg B_a(e^{it}).$$

The β_a argument function has the explicit form

$$\beta_a(t) = (\delta + \varphi) + 2 \arctan \left(\frac{1+r}{1-r} \tan \frac{t-\varphi}{2} \right), \quad (1)$$

where $a = r \cdot \exp i\varphi$, and a further δ value is introduced³, which should be chosen so that β_a becomes a continuous bijection of $[-\pi, \pi)$ for our convenience. Note that β_a is strictly increasing and invertible.

The argument function of a Blaschke product is defined as

$$\beta_{a_1, \dots, a_m}(t) := \frac{1}{m} \arg \prod_{j=1}^m B_{a_j}(z) = \frac{1}{m} \sum_{j=1}^m \beta_{a_j}(t), \quad (2)$$

with the a_j ($j = 1, \dots, m$) parameters all inside the complex unit disk. This definition makes use of the fact that the argument of the product of two complex numbers is the sum of the arguments of each. Also the $1/m$ factor is applied to maintain the $[-\pi, \pi)$ bijection property.

A NED is defined as the inverse image of an equidistant discretization. Consider the following set of $N \in \mathbb{N}$ equidistant points:

$$D_0^N := \left\{ -\pi + k \cdot \frac{2\pi}{N} : 0 \leq k \leq N-1 \right\} \subset [-\pi, \pi).$$

Then for given $1 \leq m \in \mathbb{N}$, and a_1, \dots, a_m parameters the set

$$D_{a_1, \dots, a_m}^N := \{ \beta_{a_1, \dots, a_m}^{-1}(t) : t \in D_0^N \} \subset [-\pi, \pi)$$

is a *non-equidistant discretization (NED)* of N points on the interval $[-\pi, \pi)$.

²The Malmquist-Takenaka systems can be written in explicit form using Blaschke products.

³This δ value depends on the previously mentioned possible second parameter of the Blaschke functions, when rigorously defined. One could consider $B_{a,d} = d \cdot \frac{z-a}{1-\bar{a}z}$ and $d = \exp i\delta$.

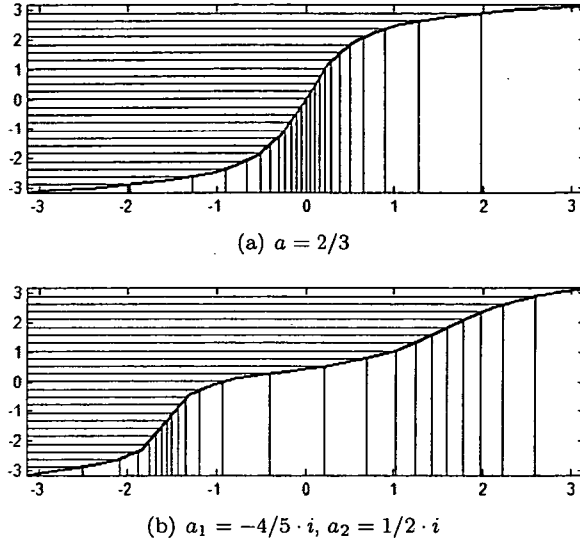


Figure 1: Argument functions and NEDs.

Figure 1 shows two example NEDs of $N = 24$ points. The first one is generated by a Blaschke function and the second one is generated by a two-factor Blaschke product with the indicated parameters. Observe that the location of the more dense areas depend on the argument of the parameter, and the degree of density corresponds to the absolute value of the parameters.⁴

Therefore we gain high control over the distribution of discrete points and so our discretization can be flexibly adapted to different problems, signals.

Non-equidistant discretizations play an important role in mathematics and applications. E.g. when $m = N$, the Malmquist–Takenaka system corresponding to the parameters a_1, a_2, \dots, a_m forms a discrete orthogonal system with respect to a scalar product defined on NED points; representation of signals and systems (Nyquist and Bode diagrams) can be improved using NEDs; etc. See works of authors of [1, 2, 3].

3 Calculating a NED

In this section we discuss the methods available for the numerical calculation of a NED, and describe an enhancement of the bisection method for this problem.

⁴Now we can see, that δ does not influence the density in any relevant way, it was really introduced only for convenience. (In this context.)

3.1 General observations and notation

Given $1 < N \in \mathbb{N}$, the number of discrete points and a_1, \dots, a_m ($1 \leq m \in \mathbb{N}$) set of parameters, denote the discrete points of an equidistant discretization by

$$d_k := -\pi + k \cdot \frac{2\pi}{N} \quad (0 \leq k \leq N-1),$$

and the corresponding points of the NED by

$$e_k := \beta_{a_1, \dots, a_m}^{-1}(d_k) \quad (0 \leq k \leq N-1). \quad (3)$$

To calculate the NED of N points with these parameters, basically we must find the e_k values for all $0 \leq k \leq N-1$, as the calculation of the d_k values is quite straightforward. When we have only $m = 1$ parameter, then the inverse formula of the argument function in (3) can be explicitly given, recall the definition in formula (1). No further effort is needed. But having $m > 1$ parameters, the inverse has no explicit form, numerical methods are needed to solve the N arising non-linear equations.

The most simple methods at hand are the well-known bisection method and Newton's method. Both have their own advantages and problems. The bisection method surely converges because of the strictly increasing property of the argument functions (the first derivative is greater than zero), but it requires a lot of calculation and converges only in first order. Newton's method would converge in order two (the first derivative is non-zero), but it needs to be initialized close enough to the solution, therefore we still need e.g. the bisection method to determine a suitable initialization point.

The derivative of an argument function according to (1) and (2) has the form:

$$\beta'_a(t) = \frac{1 - r^2}{1 + r^2 - 2r \cos(t - \varphi)}, \text{ and } \beta'_{a_1, \dots, a_m}(t) := \frac{1}{m} \sum_{j=1}^m \beta'_{a_j}(t).$$

The positivity follows from the detailed calculation, see [1].

3.2 Applying the bisection method

When applying the bisection method to find all the e_k values with the prescribed precision $\varepsilon > 0$, we should solve $N-1$ (for $1 \leq k \leq N-1$) non-linear equations one-by-one, i.e. apply the bisection method $N-1$ times. (Note that the solution $e_0 = d_0 = -\pi$ is trivial.) So the naive implementation would go as written in Algorithm 1. Denote simply by β the argument function at hand.

This algorithm needs $(N-1) \cdot \lceil \log_2(2\pi/\varepsilon) \rceil$ steps to reach the prescribed precision. (The case in Line 11 is very unlikely to happen.)

3.3 Enhancing the order of calculation

We may find that the naive implementation does not take any advantage of previously calculated solutions. But when calculating e.g. e_3 , we might make use of e_2

Algorithm 1 Naive implementation of calculating all e_k values using the bisection method.

```

1:  $e_0 \leftarrow -\pi$ 
2: for  $k = 1, \dots, N - 1$  do for the rest of the points
3:    $a \leftarrow -\pi, b \leftarrow \pi$ 
4:    $c \leftarrow (a + b)/2$ 
5:   while  $b - a > \varepsilon$  do do standard bisection iteration
6:     if  $\beta(c) < d_k$  then
7:        $a \leftarrow c$ 
8:     else if  $\beta(c) > d_k$  then
9:        $b \leftarrow c$ 
10:    else
11:      break
12:    end if
13:     $c \leftarrow (a + b)/2$ 
14:  end while
15:   $e_k \leftarrow c$ 
16: end for

```

and e_4 if these have been already found, and initialize the bisection iteration using these two values. The strictly increasing property of the argument function ensures that e_3 lies between e_2 and e_4 .

Consider the example when 8 points should be calculated. Having found e_0 and e_4 , the calculation of e_2 can be initialized using these values, therefore the bisection method could save 1 step (in average), since we have started with an interval of half the length of the original. Continuing similarly with e_1 and e_3 we may find that 1 more step can be saved for each.

Generally with the order given by the preorder traversal of a balanced binary search tree containing the values 1 through $N - 1$, we may save considerable amount of steps the bisection method to take.

Now we shall give the algorithm of calculating this order in an effective way (see Algorithm 2) with an example of a filled table in the case $N = 12$ (see Table 1).

Table 1: The appropriate calculation order of points and its generation.

j	1	2	3	4	5	6	7	8	9	10	11	12	13
$n(j)$	0	12	6	3	9	1	4	7	10	2	5	8	11
$p_1(j)$	-	-	0	0	6	0	3	6	9	1	4	7	10
$p_2(j)$	-	-	12	6	12	3	6	9	12	3	6	9	12

The table to be filled to generate this order of point contains the j indices, the points in the appropriate $n(j)$ order and $p_1(j), p_2(j)$ 'parents' of the points. In the generating algorithm we are going along the columns of this table starting with

column 3, and put the 'children' of the current column in the next free columns of the table. The variable name w denotes the column currently *watched* and f the next column to be *filled*.

Algorithm 2 The generation of the enhanced order of points.

```

1:  $n(1) \leftarrow 0, n(2) \leftarrow N$ 
2:  $n(3) \leftarrow \lfloor N/2 \rfloor, p_1(3) \leftarrow 0, p_2(3) \leftarrow N$ 
3:  $w \leftarrow 3, f \leftarrow 4$ 
4: while  $f \leq N + 1$  do                                     while table is not filled
5:   if  $n(w) - p_1(w) > 1$  then                               if there are points in between
6:      $p_1(f) \leftarrow p_1(w), p_2(f) \leftarrow n(w)$ 
7:      $n(f) \leftarrow \lfloor (p_1(f) + p_2(f))/2 \rfloor$ 
8:      $f \leftarrow f + 1$ 
9:   end if
10:  if  $p_2(w) - n(w) > 1$  and  $f \leq N + 1$  then             the other side
11:     $p_1(f) \leftarrow n(w), p_2(f) \leftarrow p_2(w)$ 
12:     $n(f) \leftarrow \lfloor (p_1(f) + p_2(f))/2 \rfloor$ 
13:     $f \leftarrow f + 1$ 
14:  end if
15:   $w \leftarrow w + 1$ 
16: end while

```

And finally we show the algorithm of the bisection method enhanced with this order of calculation: see Algorithm 3. For convenience we define $e_N := \pi$. This algorithm uses the table generated by Algorithm 2.

Algorithm 3 Enhanced implementation of calculating all e_k values using the bisection method and the order generated by Algorithm 2.

```

1:  $e_0 \leftarrow -\pi, e_N \leftarrow \pi$ 
2: for  $j = 3, \dots, N + 1$  do                                for the rest of the points
3:    $k \leftarrow n(j)$ 
4:    $a \leftarrow e_{p_1(j)}, b \leftarrow e_{p_2(j)}$ 
5:    $c \leftarrow (a + b)/2$ 
6:   while  $b - a > \varepsilon$  do                                     do standard bisection iteration
7:     the same iteration step
8:   end while
9:    $e_k \leftarrow c$ 
10: end for

```

3.4 Applying Newton's method

We have already noted that Newton's method should be initialized *close enough* to the solution to ensure (quadratic) convergence. The starting points required must be calculated with e.g. the bisection method. To make a proper statement

about what 'close enough' precisely means requires more detailed analysis and goes beyond the extent of this paper. So the application of Newton's method can be thought of as an improvement of the results (e_k values) calculated by the bisection method.

The iteration goes as usual according to the formula

$$e_k^{(next)} := e_k - \frac{\beta(e_k) - d_k}{\beta'(e_k)}.$$

Note that the function values are given by $\beta(e_k) - d_k$, since this is the expression whose zero is to be found. The iteration terminates when $|e_k^{(next)} - e_k|$ falls beneath a prescribed $\varepsilon > 0$ precision.

4 Results

In this section we describe and analyse our measurements and results concerning step counts of the bisection method, execution time, and the application of Newton's method. Both theoretical and measured values are to be displayed.

4.1 Step counts of the bisection method

Figure 2 shows for each e_k ($0 \leq k < N = 32$) point how many steps the bisection method takes to reach the precision $\varepsilon = 10^{-3}$.

Light columns show that the calculation of every e_k value (except e_0) needed 13 steps using the naive implementation. This corresponds to the theoretical number $\lceil \log_2(2\pi/10^{-3}) \rceil = 13$. Dark columns show the number of steps for each e_k using the enhanced order we defined earlier. By comparing the dark region with the whole diagram, we find that globally we can save many steps. Our gain can be visualized as the region that actually seems light gray.

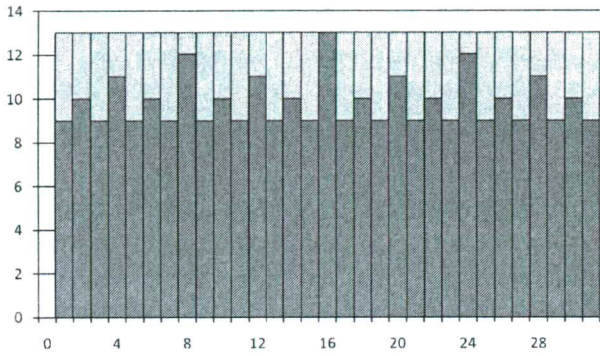
Figure 2(a) shows the theoretical number of steps⁵ as described in Section 3.3. Figure 2(b) shows the case of 1 parameter. We may see that we do not exactly save 1 step every time: sometimes none, but in some cases even 2 or more. We did not actually needed numerical methods for the above, but did in the case of 2 parameters (shown on 2(c)). The step gain also varies with k . One may observe the close relationship of these values to the parameters.

These images provide an elegant visualization of how many steps can be saved, and suggest that the theoretical estimation of global step saving is close to the real.

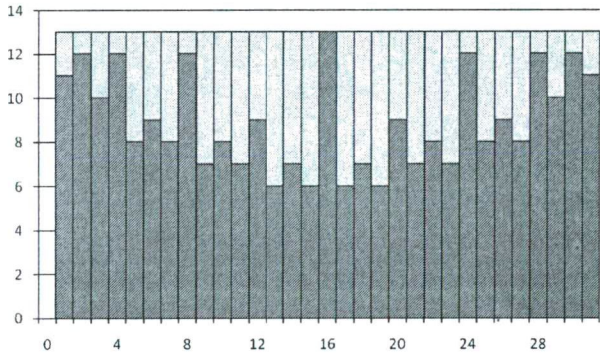
4.2 Step count and execution time ratios

Our further inspections target the fact, that the number of points and the required precision firmly affects the step count saving of the enhanced implementation of the bisection method compared to the naive one.

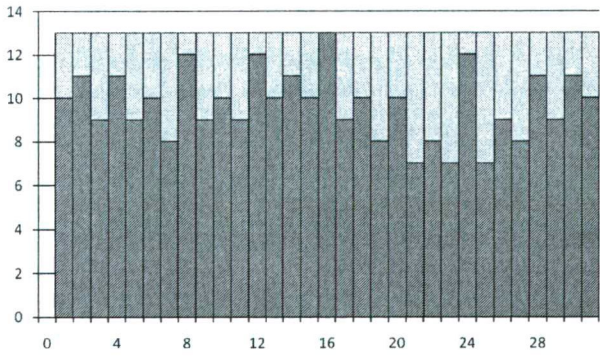
⁵Note that this theoretical step count can be approximated in the case of 1 parameter while this parameter tends to zero.



(a) Theoretical.



(b) 1 parameter ($a_1 = 3/4$).



(c) 2 parameters ($a_1 = -1/2 \cdot i, a_2 = 3/4 \cdot i$).

Figure 2: Theoretical and real step counts.

When the precision becomes greater (i.e. ε becomes smaller) the required step counts grows, but the saving does not. And as the number of points grow, the saving also increases. (Because—roughly speaking—the new points usually lie between previously calculated ones, therefore require even less steps.)

Figure 3 shows the step count ratios of the enhanced and the naive implementation of the bisection algorithm corresponding to various number of points (2^l , $l = 4, 5, 6, 7, 8$) and precisions (10^{-p} , $p = 3, 4, 5, 6, 7$). Figure 3(a) show the theoretic estimation of these ratios according using calculations similar to which resulted in Figure 2(a). Figure 3(b) shows the measured step count ratios in the case of 2 parameters randomly chosen inside the unit disk. Each value is the average of 10 independent measurements.

One can see that the theoretic estimations are very close to the real measurements, and the enhanced method requires about 50–90% of the steps of the naive implementation depending on the number of points and the precision to reach. The least saving can be seen when there are few points and high precision required, and the most saving can be observed when we have lots of points and require low precision.

We have obtained similar results concerning execution time.

4.3 Applying Newton's method

As described in Section 3.4, Newton's method can be considered mainly as an improvement of the results calculated by the bisection method.

To analyse the efficiency of Newton's method compared to the bisection method (the enhanced version) we measured the execution time (in seconds) of the bisection method to precision 10^{-12} (dashed line), and of Newton's method to the same precision after initialized with precision 10^{-2} (thin solid line) and 10^{-3} (thick solid line) in the case of 2^l ($l = 4, 5, 6, 7, 8$) points. The results can be seen on Figure 4.

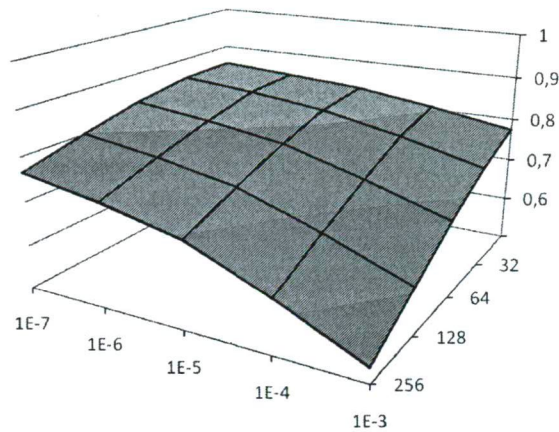
One may observe the quadratic convergence (it is much faster than the linear), and that 10^{-2} precision is close enough for the Newton's method to start. Also a near linear dependency can be seen between the number of points and the execution time.

5 Further research

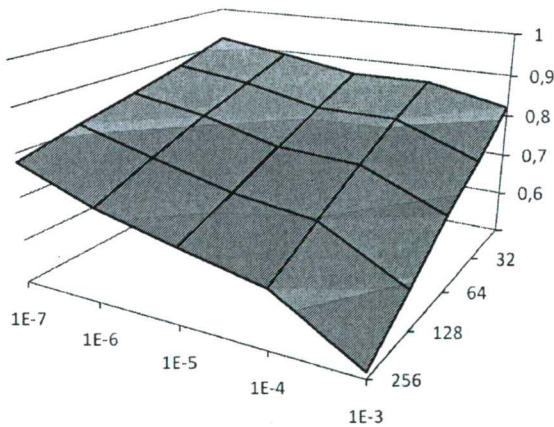
Related to the experiments described in the previous sections we can find many aspects that may be further clarified or elaborated.

Parallel implementation is a very obvious intention since the discrete points can be independently calculated. Experimenting with the emerging paradigm of general purpose GPU computing may be valuable.

The deeper mathematical analysis of the Newton method's initialization needs seems very interesting. Which bounds could be deduced (from known theorems) at each point, from where Newton's method would surely converge? Could these results be efficiently applied to further improve the speed of the calculation?



(a) Theoretical.



(b) Measured.

Figure 3: Step count ratios of the enhanced and the naive implementation of the bisection algorithm.

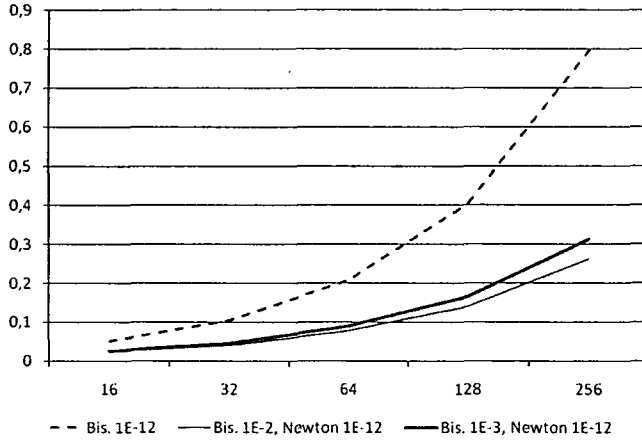


Figure 4: Execution times using Newton's method.

May the inverse of an argument function associated to a Blaschke *product* be approximated (or found explicitly)?

Multidimensional generalizations of non-equidistant discretizations of this kind could also be investigated. Possible application areas may be found e.g. at the fields of image processing or sampling methods.

6 Summary

We have made several experiments concerning the calculation of non-equidistant discretizations generated by Blaschke products and the associated argument functions. To our knowledge, no effort has been made before in this area.

We have managed to reduce the time needed for the calculation (using the bisection method) to about 50–70% of the original time by introducing a better order of the discrete points to calculate.

Further improvements have been measured by applying Newton's algorithm combined with the above.

7 Acknowledgements

This work was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

The author also wishes to thank the organizers of the 7th Conference of PhD Students in Computer Science (CSCS 2010) at the Institute of Informatics, University of Szeged.

References

- [1] Bokor, József and Schipp, Ferenc. Approximate linear h^∞ identification in Laguerre and Kautz basis. *IFAC Automatica J.*, 34:463–468, 1998.
- [2] Bokor, József, Schipp, Ferenc, and Soumelidis, Alexandros. Frequency domain representation of signals in rational orthogonal bases. In *Proceedings of the 10th Mediterranean Conference on Control and Automation*, Lisbon, Portugal, 2002.
- [3] Pap, Margit and Schipp, Ferenc. Malmquist–Takenaka systems and equilibrium conditions. *Mathematica Pannonica*, 12:185–194, 2001.
- [4] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group I. *Pure Mathematics and Applications (Pu.M.A.)*, 17(3–4):387–395, 2006.
- [5] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group II. *Annales Universitatis Scientarium Budapestinensis Sectio Computatorica*, 29:157–173, 2008.
- [6] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group III. *Publicationes Mathematicae Debrecen*, 75(1–2):263–283, 2009.
- [7] Schipp, Ferenc. Fast Fourier transform for rational systems. *Mathematica Pannonica*, 13(2):265–275, 2002.

2D Parallel Thinning and Shrinking Based on Sufficient Conditions for Topology Preservation*

Gábor Németh[†], Péter Kardos[†], and Kálmán Palágyi[†]

Abstract

Thinning and shrinking algorithms, respectively, are capable of extracting medial lines and topological kernels from digital binary objects in a topology preserving way. These topological algorithms are composed of reduction operations: object points that satisfy some topological and geometrical constraints are removed until stability is reached. In this work we present some new sufficient conditions for topology preserving parallel reductions and fifty-four new 2D parallel thinning and shrinking algorithms that are based on our conditions. The proposed thinning algorithms use five characterizations of endpoints.

Keywords: thinning, shrinking, parallel reductions, digital topology, topology preservation

1 Introduction

Shape- and topological analysis of discrete patterns play an important role in image processing and computer vision [17]. Considering the efficiency, several applications use iterative object reduction, like thinning and shrinking. Reduction algorithms are composed of reduction operations that delete object points.

Thinning is a frequently applied skeletonization technique [17, 18], which provides the relevant geometric and topological properties of the shapes. Thinning algorithms are to produce medial lines from 2-dimensional digital objects in a topology preserving way [8]. They preserve endpoints that provide important geometrical information relative to the shape of the objects.

Shrinking algorithms are to extract topological kernels [6]. A topological kernel is a minimal set of points that is topologically equivalent [8, 9, 16] to the original

*This work was supported by TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency, the European Union and the European Regional Development Fund under the grant agreement TÁMOP-4.2.1/B-09/1/KONV-2010-0005, and the grant CNK80370 of the National Office for Research and Technology (NKTH), and the Hungarian Scientific Research Fund (OTKA).

[†]Institute of Informatics, University of Szeged, Hungary,
E-mail: {gnemeth,pkardos,palagyi}@inf.u-szeged.hu

object (i.e., if we remove any further point from it, then the topology is not preserved) [6, 8, 9, 16]. Shrinking algorithms preserve the topology [6, 8, 9, 16] of the objects, however, they do not take the object geometry into consideration.

Parallel thinning and shrinking algorithms are composed of parallel reduction operations which delete a set of object points simultaneously [6, 7]. Note that there exists some “*shrinking to a residue*” algorithms, in which every object is transformed into a single point by eliminating cavities [6]. In this paper our attention is focused on the topology-oriented shrinking, and we consider shrinking algorithms as thinning ones with no endpoint preservation.

All thinning and shrinking algorithms need to preserve the topology [8]. Despite of this topological constraint, Couprie found five existing 2D parallel thinning algorithms that do not satisfy it [5]. In order to verify that a parallel reduction preserves topology, Ronse introduced the minimal non-simple sets in [16], and Kong gave some sufficient conditions [9]. Bertrand introduced the P-simple points [1] and the critical kernels [2] that provide methodologies to design topology preserving parallel thinning algorithms. Bertrand and Couprie proposed various parallel thinning algorithms based on critical kernels [3], and they linked the critical kernels to minimal non-simple sets and P-simple points in [4]. However critical kernels constitute a general framework in the category of abstract complexes in any dimension, designing parallel thinning algorithms working on discrete grids might be difficult. That is why we introduced modified versions of Kong’s sufficient conditions [9] and combined them with the known parallel thinning approaches and endpoint characterizations to generate a family of topology preserving thinning and shrinking algorithms [13, 15]. In our opinion, one can implement these algorithms easily.

We use the fundamental concepts of digital topology as reviewed by Kong and Rosenfeld [8].

A 2D (8,4) binary digital picture $\mathcal{P} = (\mathbb{Z}^2, 8, 4, B)$ is a quadruple [8], where \mathbb{Z}^2 is the set of 2D discrete points. The elements of $B \subseteq \mathbb{Z}^2$ are the black points, having the value of “1”, form the 8-connected objects, while points in $\mathbb{Z}^2 \setminus B$ are white ones, having the value of “0”, and are assigned to the 4-connected background and cavities. Let $N_8(p)$ and $N_4(p)$ denote the set of 8- and 4-adjacent points to p , respectively. Furthermore, we use the notations $N_4^*(p)$ and $N_8^*(p)$, where $N_8^*(p) = N_8(p) \setminus \{p\}$ and $N_4^*(p) = N_4(p) \setminus \{p\}$. The lexicographical order relation \prec between two distinct points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ is defined as follows:

$$p \prec q \iff p_y < q_y \vee (p_y = q_y \wedge p_x < q_x).$$

A black point is called a *border point* if it is 4-adjacent to at least one white point. The other black points are called *interior points*.

There are three major strategies for parallel thinning and shrinking algorithms: fully parallel, subiteration-based, and subfield-based [6, 7, 10, 18]. Németh and Palágyi studied a number of parallel thinning algorithms that are based on some sufficient conditions for topology preservation, and the three conventional types of endpoints were considered [15].

In this paper we introduce some advanced sufficient conditions for topology preservation. Then, we propose forty-five new parallel thinning algorithms and nine

shrinking ones that are based on these new conditions. Our thinning algorithms take five endpoint characterizations into consideration.

The rest of this paper is organized as follows. In Section 2, we propose our new sufficient conditions for topology preserving parallel reductions. Section 3 reviews the proposed parallel thinning and shrinking algorithms and presents some illustrative results. In Section 4, some properties of our algorithms are discussed. Finally, we round off the paper with some concluding remarks.

2 Topology Preserving Parallel Reductions

In this section, some results concerning topology preserving parallel reduction operations are reviewed.

A reduction operation may change some black points to white ones, which is referred to as deletion, while white points remain unchanged. A parallel reduction operation deletes a set of black points simultaneously.

A 2D reduction operation does *not* preserve topology [8, 9, 16] if any object in the input picture is split (into several objects) or is completely deleted, any cavity in the input picture is merged with the background or another cavity, or any cavity is created where there was none in the input picture.

A black point is *simple* in a picture if its deletion is a topology preserving reduction [8]. The simplicity of a point is a local property, since it can be decided by investigating its 3×3 neighborhood [7].

Although the deletion of a simple point preserves the topology, simultaneous deletion of a set of simple points may disconnect or eliminate objects, merge cavities with each other or with the background, or create new cavities. To ensure topology preservation for parallel reductions, Kong and Ronse gave some sufficient conditions [9, 16].

Theorem 1. *A parallel reduction operation is topology preserving for $(8,4)$ pictures if all of the following conditions hold:*

1. *Only simple points are deleted.*
2. *For any two 4-adjacent points p and q that are deleted, p is simple after deletion of q , or q is simple after p is deleted.*
3. *No black component contained in a 2×2 square is deleted completely.*

Theorem 1 is generally used to verify the topological correctness of the thinning and the shrinking algorithms. Németh and Palágyi proposed alternative sufficient conditions for topology preservation that can be applied to generate deletion conditions for various thinning algorithms [15].

Theorem 2. *Let \mathcal{O} be a parallel reduction operation. The operation \mathcal{O} is topology preserving for $(8,4)$ pictures if all of the following conditions hold, for any black point p in picture $(\mathbb{Z}^2, 8, 4, B)$ deleted by \mathcal{O} :*

1. Point p is simple in $(\mathbb{Z}^2, 8, 4, B)$.
2. For any simple point $q \in N_4^*(p) \cap B$, p is simple in picture $(\mathbb{Z}^2, 8, 4, B \setminus \{q\})$, or q is simple in picture $(\mathbb{Z}^2, 8, 4, B \setminus \{p\})$.
3. Point p does not coincide with the points marked “ \star ” in the seven black components depicted in Fig. 1(d)–(j).

By nature of the sufficient conditions of Theorem 2, any parallel reduction operations derived from them can not alter some 2-point wide segments (see Fig. 2). To extract the topological kernel, the algorithm should remove all the simple points, since topological kernels do not contain any simple points. That is why we propose some improved sufficient conditions.

Theorem 3. *Let \mathcal{O} be a parallel reduction operation. The operation \mathcal{O} is topology preserving for $(8, 4)$ pictures if all of the following conditions hold, for any black point p in picture $(\mathbb{Z}^2, 8, 4, B)$ deleted by \mathcal{O} :*

1. Point p is simple in $(\mathbb{Z}^2, 8, 4; B)$.
2. For any simple point $q \in N_4^*(p) \cap B$, p is simple in picture $(\mathbb{Z}^2, 8, 4, B \setminus \{q\})$ or q is simple in the picture $(\mathbb{Z}^2, 8, 4, B \setminus \{p\})$, or $q \prec p$.
3. Point p does not coincide with the points marked “ \star ” in the seven black components depicted in Fig. 1(d)–(j).

Proof. To prove this theorem, it is sufficient to show that all conditions of Theorem 1 are satisfied.

- Condition 1 of Theorem 3 corresponds to Condition 1 of Theorem 1.
- Let p', q' be two 4-adjacent simple points in B such that p' is not simple in $B \setminus \{q'\}$ and q' is not simple in $B \setminus \{p'\}$. If $p' \prec q'$ (hence $q' \not\prec p'$) then set $p = p'$ and $q = q'$ otherwise set $p = p'$ and $q = q'$ by Condition 2 of Theorem 2, p is not deleted by \mathcal{O} , thus Condition 2 of Theorem 1 is satisfied, for the pair p', q' is not deleted.
- The black component in Fig. 1(a) is an isolated and non-simple point, hence it can not be deleted by Condition 1 of Theorem 3. The next two black components (see Fig. 1(b) and (c)) are formed by two 4-adjacent black points. One of them (that comes lexicographically first) can not be deleted by Condition 2 of Theorem 3. The remaining seven black components depicted in Fig. 1(d)–(j) can not be deleted completely by Condition 3 of Theorem 3 (the points marked “ \star ” are protected in these cases). Hence, Condition 3 of Theorem 1 holds.

□

Note that the general sufficient conditions of Theorem 3 can be simplified if we consider a given parallel reduction strategy.

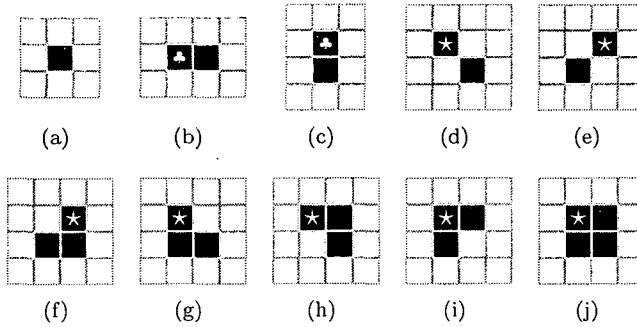


Figure 1: The ten possible black components contained in a 2×2 square. Black points marked “ \star ” are designated to be preserved by Condition 3 of Theorems 2 and 3. Both black components in (b) and (c) must remain unchanged by Theorem 2, but only points marked “ \clubsuit ” are to be preserved by Theorem 3.

3 Thinning and Shrinking Algorithms

In this section, forty-five new parallel thinning algorithms and nine shrinking ones are presented. These topological algorithms are composed of parallel reductions derived from our new sufficient conditions for topology preservation (see Theorem 3).

Thinning algorithms preserve endpoints, some border points that provide relevant geometrical information with respect to the shape of the object. During the shrinking process no endpoint criterion is considered.

Definition 1. *There is no endpoint of type E0.*

To standardize the notations, a shrinking algorithm can be considered a special case of a thinning one, where no endpoint is preserved, hence we use endpoint of type E0 (i.e., the empty set of the endpoints) for it. There exist three conventional types of endpoints E1, E2, and E3 [7].

Definition 2. *A border point p is an endpoint of type E1 if there is exactly one black point in $N_g^*(p)$.*

Definition 3. *A border point p is an endpoint of type E2 if there are at most two 4-adjacent black points in $N_g^*(p)$.*

Definition 4. *A border point p is an endpoint of type E3 if there are at most two 8-adjacent black points in $N_g^*(p)$.*

We consider two additional endpoint criteria [3, 12].

Definition 5. *A border point p is an endpoint of type E4 if there is no interior point in $N_g^*(p)$.*

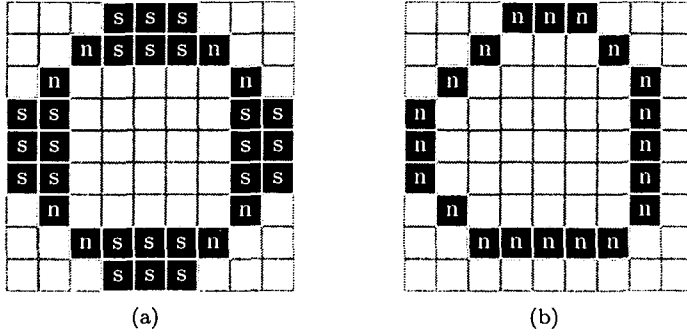


Figure 2: An example to compare the effects of reduction operations derived from Theorems 2 and 3. The points marked “s” are simple, while “n” denotes the non-simple ones. No reduction operation derived from Theorem 2 can remove any point from picture (a), but there are some reduction operations derived from Theorem 3 are capable of removing some additional points (b). Note that there is no simple point in picture (b), hence it is a topological kernel.

Definition 6. A border point p is an endpoint of type E5 if there is no interior point in $N_4^*(p)$.

Let \mathcal{E}_i denote the set of endpoints of type E_i ($i = 1, \dots, 5$) for an arbitrarily chosen binary image. It is easy to see that

$$\mathcal{E}_1 \subset \mathcal{E}_2 \subset \mathcal{E}_3 \subset \mathcal{E}_4 \subset \mathcal{E}_5.$$

Some examples of these characterizations of endpoints are depicted in Fig. 3.



Figure 3: Examples of endpoints. Points marked “ k ” are endpoints of type E_i ($k = 1, \dots, 5$; $i = k, \dots, 5$). Points marked “i” are interior points, while points marked “b” are border points that are not endpoints.

It is easy to see that all points in all possible black components contained in a 2×2 square are endpoints of types E4 and E5, since there is no interior point in these components (see Fig. 1). Consequently, Condition 3 of Theorem 3 can be omitted in the case of parallel reductions that preserve endpoints of type E4 or E5. Hence, we can state the following:

Theorem 4. Let \mathcal{O}^ε be a parallel reduction operation that preserves endpoints of type ε , where $\varepsilon \in \{E_0, \dots, E_5\}$. The operation \mathcal{O}^ε is topology preserving for $(8, 4)$

pictures if all of the following conditions hold, for any black point p in picture $(\mathbb{Z}^2, 8, 4, B)$ deleted by \mathcal{O}^ε :

1. Point p is simple and not an endpoint of type ε .
2. For any simple point $q \in N_4^*(p) \cap B$ that is not an endpoint of type ε , p is simple in $(\mathbb{Z}^2, 8, 4, B \setminus \{q\})$ or q is simple in $(\mathbb{Z}^2, 8, 4, B \setminus \{p\})$, or $q \prec p$.
3. Depending on a given endpoint characterization ε , the following conditions are to be satisfied:
 - If $\varepsilon = E0$, then point p does not coincide with the point marked “ \star ” depicted in Fig. 1(d)-(j).
 - If $\varepsilon = E1$, then point p does not coincide with the point marked “ \star ” depicted in Fig. 1(f)-(j).
 - If $\varepsilon \in \{E2, E3\}$, then point p does not coincide with the point marked “ \star ” depicted in Fig. 1(j).

Proof. Conditions 1, 2, and 3 fulfill the Conditions 1, 2, and 3 of Theorem 3, respectively. In the case of Conditions 3, black points in Fig. 1(d) and (e) are endpoints of type $E1$. In Fig. 1(d) – (i), there is at least one $E2$ or $E3$ endpoint in the component, hence it is not deletable completely. If $\varepsilon \in \{E4, E5\}$, then no black component contained in a 2×2 square can be deleted completely by \mathcal{O}^ε since all of their elements are endpoints to be preserved. □

In the rest of this section thinning and shrinking algorithms composed of parallel reduction operations that satisfy Theorem 4 are reported. The properties of these algorithms are discussed in Section 4.

The proposed algorithms were tested on objects of different shapes. Here we can present their results superimposed on just one 120×45 picture with 2572 object points, see Figs. 4, 6, 7, 11, 12. The pairs of numbers in parentheses are the count of object points in the produced pictures and the parallel speed (i.e., the number of the required parallel reduction operations [7]).

3.1 Fully Parallel Algorithms

In fully parallel algorithms, the same parallel reduction operation is applied in each iteration step [7].

The general scheme of the fully parallel thinning algorithms $FP\text{-}\varepsilon$ using endpoint of type ε are sketched by Alg. 1 ($\varepsilon \in \{E0, \dots, E5\}$).

The $FP\text{-}\varepsilon$ -deletable points are defined as follows:

Definition 7. Let $\varepsilon \in \{E0, \dots, E5\}$ be a characterization of endpoints. Black point p is $FP\text{-}\varepsilon$ -deletable if all the conditions of Theorem 4 hold.

Algorithm 1 Algorithm FP- ε

```

1: Input: picture  $(\mathbb{Z}^2, 8, 4, X)$ 
2: Output: picture  $(\mathbb{Z}^2, 8, 4, Y)$ 
3:  $Y = X$ 
4: repeat
5:    $D = \{p \mid p \text{ is FP-}\varepsilon\text{-deletable in } Y\}$ 
6:    $Y = Y \setminus D$ 
7: until  $D = \emptyset$ 

```

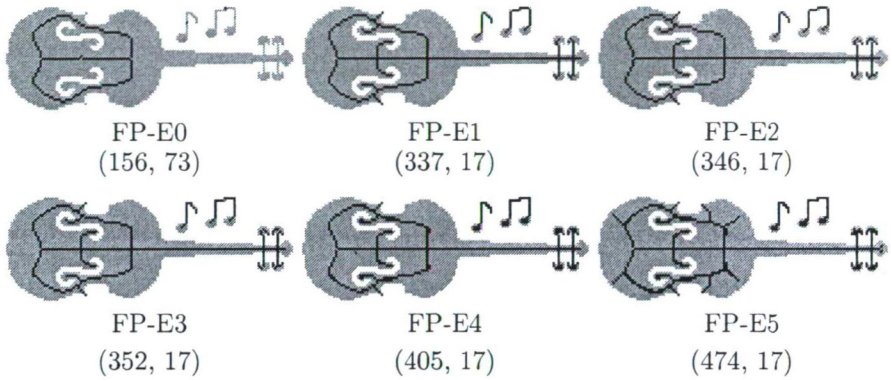


Figure 4: A topological kernel and five medial lines produced by the proposed fully parallel algorithms.

Figure 4 presents illustrative examples for topological kernels and medial lines produced by algorithms FP- ε ($\varepsilon \in \{E0, \dots, E5\}$).

It can readily be seen that deletable points of the proposed fully parallel algorithms (see Def. 7) are derived directly from conditions of Theorem 4. Hence, all of the six algorithms are topology preserving.

3.2 Subiteration-based Algorithms

In subiteration-based (or frequently referred to as directional) thinning algorithms, an iteration step is decomposed into k successive parallel reduction operations according to the k deletion directions. If direction d is the current deletion direction, then a set of d -border points can be deleted by the parallel reduction operation assigned to it [7].

A black point p is an N -border point if point p_N (see Fig. 5) is white. The W -, S -, E -border points can be defined similarly. In addition, a black point p is an NE -border point if p_N or p_E is white. Considering another pairs of directions, we can likewise talk about NW -, SW -, SE -border points (see Fig. 5).

Let ε be a type of endpoint ($\varepsilon \in \{E0, \dots, E5\}$) and $Q = \langle d_1, \dots, d_k \rangle$ be a sequence of the deletion directions. For existing 2-subiteration algorithms, the

p_{NW}	p_N	p_{NE}
p_W	p	p_E
p_{SW}	p_S	p_{SE}

Figure 5: Notations for the 3×3 neighborhood of point p .

two deletion directions NE and SW are generally applied [7, 10, 18]. Of course, applying the intermediate directions SE and NW , we get algorithms $SI-\langle NW, SE \rangle-\epsilon$. In the case of the existing 4-subiteration algorithms, cardinal deletion directions N , E , S , and W are considered [7, 10, 18].

Note that these subiteration-based algorithms with intermediate deletion directions produce distorted results for symmetric objects. In order to improve the medialness property of the 2-subiteration algorithms, we propose a new 4-subiteration scheme with the sequence of intermediate deletion directions $\langle NE, SW, NW, SE \rangle$. Note that this scheme is capable of removing the two outmost layers from the objects at an iteration step.

Directional thinning algorithms using endpoint of type ϵ and a sequence of deletion directions Q are sketched by algorithm $SI-\langle Q \rangle-\epsilon$ (see Alg. 2), ($\epsilon \in \{E0, \dots, E5\}$; $Q = \langle NE, SW \rangle, \langle NW, SE \rangle, \langle N, E, S, W \rangle, \langle NE, SW, NW, SE \rangle$).

Algorithm 2 Algorithm $SI-\langle Q \rangle-\epsilon$

```

1: Input: picture  $(\mathbb{Z}^2, 8, 4, X)$ 
2: Output: picture  $(\mathbb{Z}^2, 8, 4, Y)$ 
3:  $Y = X$ 
4: repeat
5:    $D = \emptyset$ 
6:   for all  $d \in Q$  do
7:      $D_d \stackrel{\text{def}}{=} \{p \mid p \text{ SI-}d\text{-}\epsilon\text{-deletable in } Y\}$ 
8:      $Y = Y \setminus D_d$ 
9:      $D = D \cup D_d$ 
10:  end for
11: until  $D = \emptyset$ 
```

The $SI-d-\epsilon$ -deletable points are defined as follows:

Definition 8. Black point p is called $SI-d-\epsilon$ -deletable if all the following conditions hold ($\epsilon \in \{E0, \dots, E5\}$; $d \in \{N, E, S, W, NE, SW, NW, SE\}$):

1. Point p is a d -border point, simple, and not an endpoint of type ϵ ,
2. For any simple d -border point $q \in N_4^*(p)$ that is not an endpoint of type ϵ , p is simple in $N_8^*(p) \setminus \{q\}$ or q is simple in $N_8^*(q) \setminus \{p\}$, or $q \prec p$,

3. Depending on a given endpoint characterization ε , the following conditions are to be satisfied:

- a) If $\varepsilon = E0$, then the following cases have to be taken into consideration:
 - if $d = NE$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d), (e), (f), (h), and (i),
 - if $d = SW$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d), (e), (f), (g), and (i),
 - if $d = NW$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d), (e), (g), (h), and (i),
 - if $d = SE$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d), (e), (f), (g), and (h),
 - if $d \in \{N, E, S, W\}$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d) and (e).
- b) If $\varepsilon = E1$, then the following cases have to be checked:
 - if $d = NE$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(f), (h), (i),
 - if $d = SW$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(f), (g), (i),
 - if $d = NW$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(g), (h), (i),
 - if $d = SE$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(f), (g), (h).

Some topological kernels and medial lines proposed 2- and 4-subiteration algorithms are presented in Figs. 6 and 7, respectively.

It can readily be seen that deletable points of the proposed subiteration-based algorithms (see Def. 8) are derived from the conditions of Theorem 4. Hence, all of the twenty-four algorithms are topology preserving.

3.3 Subfield-based Algorithms

Subfield-based algorithms partition the digital space into k subfields. During an iteration step, the subfields are alternatively activated, and a set of border points in the active subfield can be deleted by a parallel reduction operation [7].

The existing subfield-based thinning algorithms partition the 2-dimensional digital space into two and four subfields, see Fig. 8. In the case of k subfields, the i -th subfield denoted by $S_k(i)$ is defined as follows ($k = 2, 4$; $i = 0, \dots, k - 1$):

$$S_2(i) = \{p = (x, y) \mid (x + y) \equiv i \pmod{2}\}, \quad (1)$$

$$S_4(i) = \{p = (x, y) \mid 2 \cdot (y \bmod 2) + (x \bmod 2) = i\}. \quad (2)$$

It is easy to see that there is no 4-adjacent pair of points in the same subfield, hence Theorem 4 can be simplified in the following way.

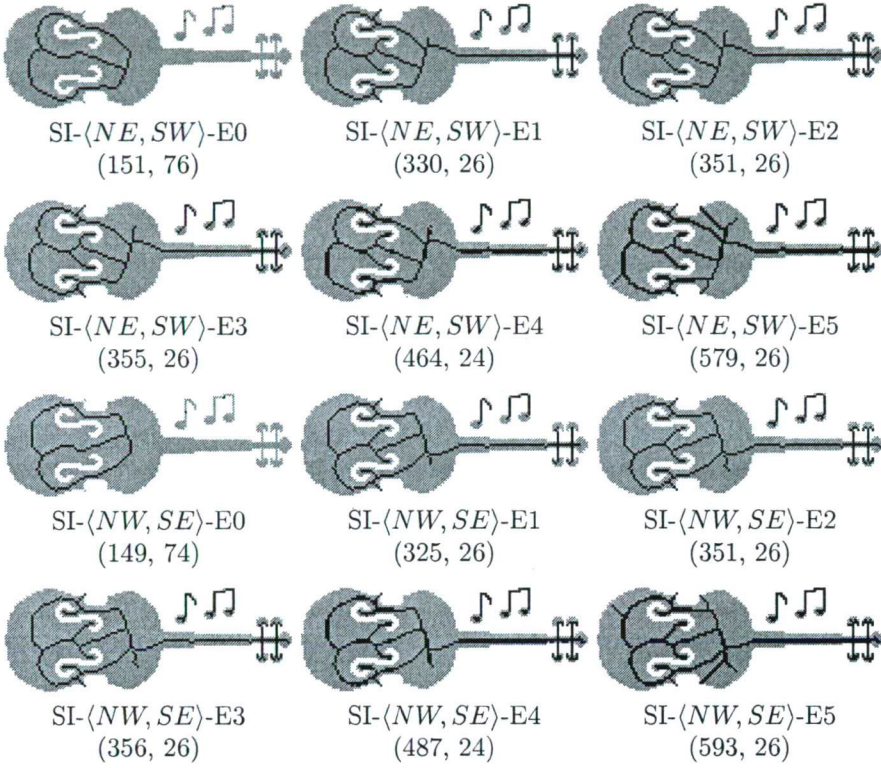


Figure 6: Two topological kernels and ten medial lines produced by the proposed 2-subiteration algorithms.

Theorem 5. Let a picture $\mathcal{P} = (\mathbb{Z}^2, 8, 4, B)$ be partitioned into k subfields, and let ε be a type of endpoints ($\varepsilon \in E0, \dots, E5$). Let $p \in B$ be a black point in the active subfield $S_k(i)$, and $\mathcal{O}_{SF_k}^\varepsilon$ be a parallel reduction operation such that $\mathcal{O}_{SF_k}^\varepsilon$ deletes p . The parallel reduction operation $\mathcal{O}_{SF_k}^\varepsilon$ is topology preserving if all of the following conditions hold:

1. Point $p \in S_k(i) \cap B$ is simple in \mathcal{P} and not an endpoint of type ε .
2. If $k = 2$ and $\varepsilon = E0$, then p does not coincide with the point marked “ \star ” depicted in Fig. 1(d) and (e).

Proof. The proof of this theorem is very easy, since it is sufficient to see that it is a special case of Theorem 3. If Condition 1 of Theorem 5 is fulfilled, then Condition 1 of Theorem 1 is satisfied. Condition 2 of Theorem 3 is not necessary to be checked, since there is no 4-adjacent pair of points in the same subfield, and operation $\mathcal{O}_{SF_k}^\varepsilon$ can delete a set of points in the active subfield. Finally, it is obvious that only two black components depicted in Fig. 1(d) and (e) contain simple points belonging to

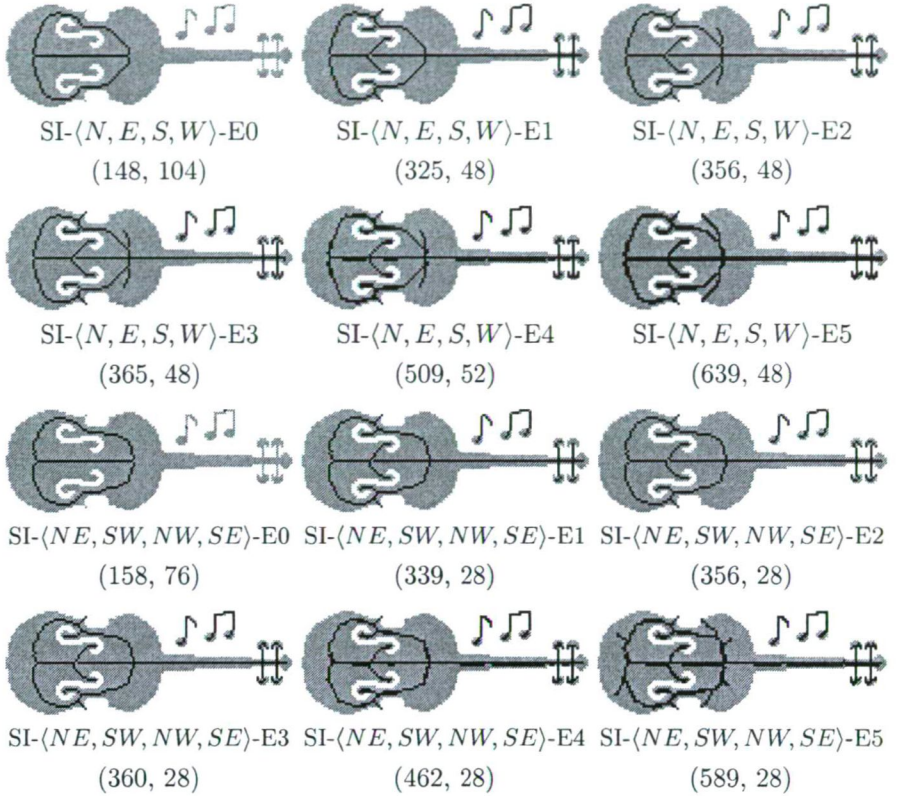


Figure 7: Two topological kernels and ten medial lines produced by the proposed 4-subiteration algorithms.

the same subfield if $k = 2$. Therefore, if Condition 2 of Theorem 5 holds, then Condition 3 of Theorem 1 is satisfied. \square

Our subfield-based thinning algorithms SF- k - ε ($k = 2, 4$; $\varepsilon \in \{E0, \dots, E5\}$) derived from Theorem 5 are sketched by Alg. 3.

In the case of subfield-based algorithms, deletable points are defined as follows.

Definition 9. *Black point p is SF- k - i - ε -deletable if the following conditions hold, ($k = 2, 4$; $i = 0, \dots, k - 1$; $\varepsilon \in \{E0, \dots, E5\}$):*

1. *Point p is simple in subfield $S_k(i)$ and not an endpoint of type ε ,*
2. *If $k = 2$ and $\varepsilon = E0$, then p does not coincide with the points marked “ \star ” in Fig. 1(d) and 1(e).*

Some topological kernels and medial lines produced by our subfield-based algorithms are presented in Figs. 11 and 12.

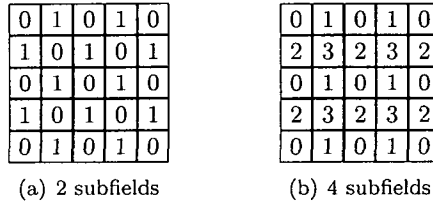


Figure 8: Partitions of \mathbb{Z}^2 into two (a) and four (b) subfields. For the k -subfield case, the points marked i are in the subfield $S_k(i)$ ($k = 2, 4$, $i = 0, \dots, k-1$).

Algorithm 3 Algorithm SF- k - ε

```

1: Input: picture  $(\mathbb{Z}^2, 8, 4, X)$ 
2: Output: picture  $(\mathbb{Z}^2, 8, 4, Y)$ 
3:  $Y = X$ 
4: repeat
5:    $D = \emptyset$ 
6:   for  $i = 0$  to  $k-1$  do
7:      $D_i = \{p \mid p \text{ is SF-}k\text{-}i\text{-}\varepsilon\text{-deletable in } Y\}$ 
8:      $Y = Y \setminus D_i$ 
9:      $D = D \cup D_i$ 
10:  end for
11: until  $D = \emptyset$ 

```

A drawback of the conventional subfield-based thinning algorithms is that they may produce several unwanted skeletal branches or the thinning process can be blocked, since endpoints preserved in a subiteration inhibit the deletion of some further points in the next iteration steps. In order to overcome this problem, we proposed a new scheme with iteration-level endpoint checking [14, 15]. According to this strategy, endpoints are marked in the beginning of each iteration step. In addition, this new strategy allows deletion of a set of border points, which were in the outmost layer in the beginning of the current iteration step.

Our new subfield-based scheme produces different residues in the case of shrinking algorithms as well, see Fig. 9. It can be stated that the new strategy produces less unwanted side branches than the conventional subfield-based thinning scheme (see Figs. 10, 11, and 12).

Our subfield-based thinning algorithms SF- k -IL- ε ($k = 2, 4$; $\varepsilon \in \{E0, \dots, E5\}$) with iteration-level endpoint checking are sketched by Alg. 4.

It can be seen that deletable points of the proposed subfield-based algorithms (see Def. 9) are derived directly from conditions of Theorem 4. Hence, all of the twenty-four algorithms are topology preserving.

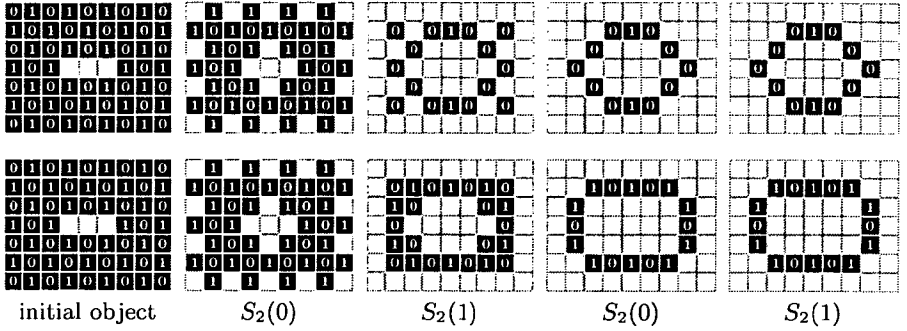


Figure 9: Two iteration steps of the conventional 2-subfield shrinking algorithm SF-2-E0 (upper row) and algorithm SF-2-IL-E0 with iteration-level endpoint checking (lower row). Subfield $S_2(i)$ is activated in the i -th phase ($i = 0, 1$). The numbers indicate the subfield indices.

Algorithm 4 Algorithm SF- k -IL- ε

```

1: Input: picture  $(\mathbb{Z}^2, 8, 4, X)$ 
2: Output: picture  $(\mathbb{Z}^2, 8, 4, Y)$ 
3:  $Y = X$ 
4: repeat
5:    $D = \emptyset$ 
6:    $E = \{p \mid p \text{ is a border point but not an endpoint of type } \varepsilon \text{ in } (\mathbb{Z}^2, 8, 4, Y)\}$ 
7:   for  $i = 0$  to  $k - 1$  do
8:      $D_i = \{p \mid p \text{ is SF-}k\text{-}i\text{-E0-deletable in } E \cap S_k(i)\}$ 
9:      $Y = Y \setminus D_i$ 
10:     $D = D \cup D_i$ 
11:   end for
12: until  $D = \emptyset$ 

```

4 Discussion

This section is to discuss some important properties of the proposed algorithms.

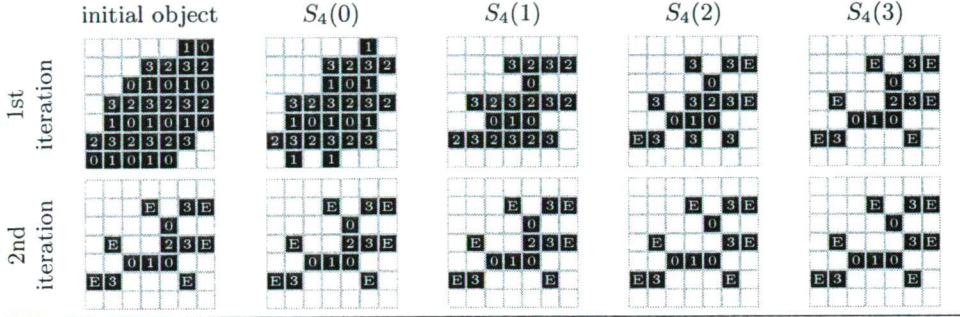
Definition 10. A result of a thinning algorithm is minimal if it does not contain any simple point except the type of endpoint taken into consideration.

Definition 11. A result of a shrinking algorithm is minimal if it there is no simple point in it.

Proposition 1. The results of the proposed fully parallel algorithms FP- ε are minimal for any pictures ($\varepsilon \in \{E0, \dots, E5\}$).

Proof. Let us suppose that a black and non-end point p remains simple after the last iteration step.

Algorithm SF-4-E1



Algorithm SF-4-IL-E1

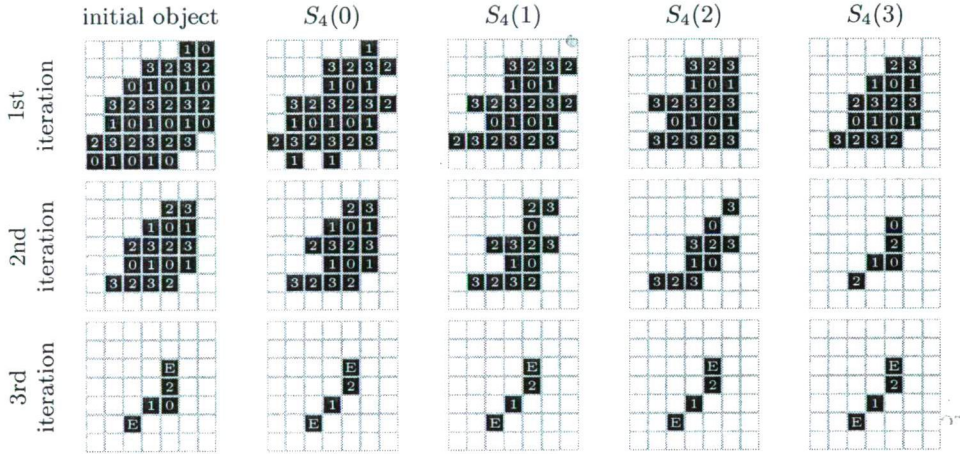


Figure 10: Phases of algorithms SF-4-E1 and SF-4-IL-E1. Subfield $S_4(i)$ is activated in the i -th phase ($i = 0, 1, 2, 3$). Points marked E are the E1 endpoints to be preserved and numbers indicate the subfield indices.

Let \mathcal{R} be the set of all remaining simple points that are not endpoints in the output picture. There is a $p \in \mathcal{R}$ such that, for any $q \in \mathcal{R} \setminus \{p\}$, $q \prec p$. This means that p satisfies all conditions of Theorem 4:

- each point in \mathcal{R} satisfies Condition 1 of Theorem 4,
- since $q \prec p$ for each $q \in \mathcal{R} \setminus \{p\}$, thus p satisfies Conditions 2 and 3 of Theorem 4.

Therefore, p is FP - ε -deletable. We come to a contradiction with our assumption, as the algorithm should have deleted p in the last iteration. \square

Proposition 2. *The results of the proposed subiteration-based algorithms $SI(Q)\text{-}\varepsilon$*

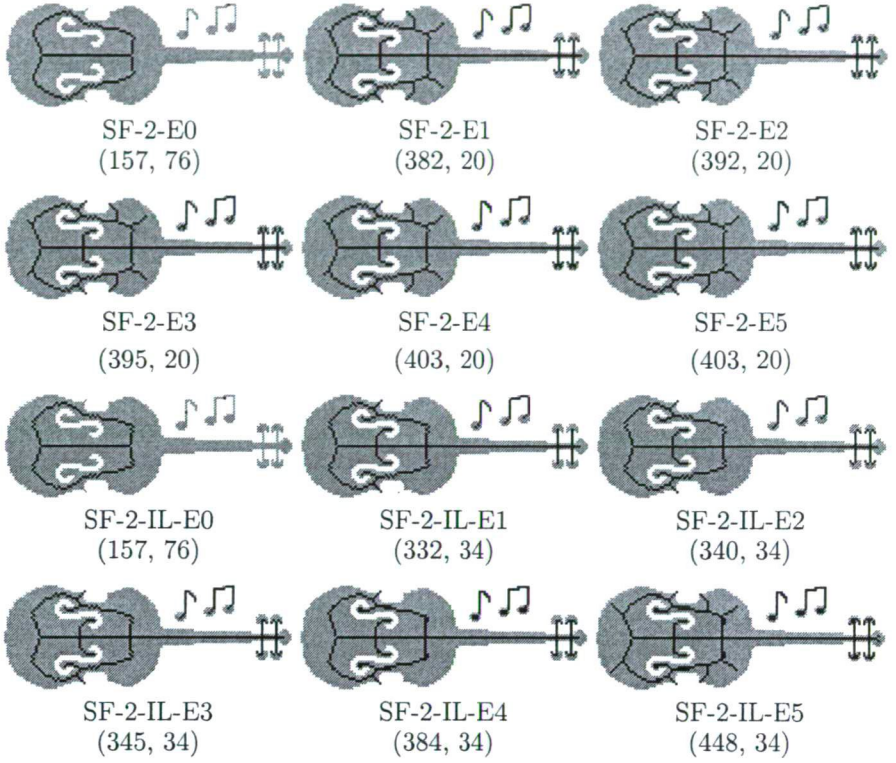


Figure 11: Two topological kernels and ten topological kernels produced by the proposed 2-subfield algorithms.

are minimal for any picture $(Q \in \{\langle NE, SW \rangle, \langle NW, SE \rangle, \langle N, E, S, W \rangle, \langle NE, SW, NW, SE \rangle\}; \varepsilon \in \{E0, \dots, E5\})$.

Proof. The proof goes similarly, as in the fully parallel case. Let us suppose that a black point p remains simple after the last iteration step.

- If $\varepsilon = E0$, then if p is a d -border point (considering any deletion direction d) and there is no other d -border simple point in $N_4^*(p)$, then p is deletable. Otherwise, let $q \in N_4^*(p)$ be a simple d -border point. Then, p or q can be deletable according to Condition 2 of Theorem 4, when the lexicographically first remains simple after the deletion of the other one. Both of these two cases lead to a contradiction.
- If $\varepsilon \in \{E1, \dots, E5\}$, then p must fulfill the considered endpoint criterion, otherwise it should have been deleted.

□

Proposition 3. *The results of the proposed subfield-based reduction algorithms $SF-k-\varepsilon$, $SF-k-IL-\varepsilon$ are minimal for any pictures $(k = 2, 4; \varepsilon \in \{E0, \dots, E5\})$.*

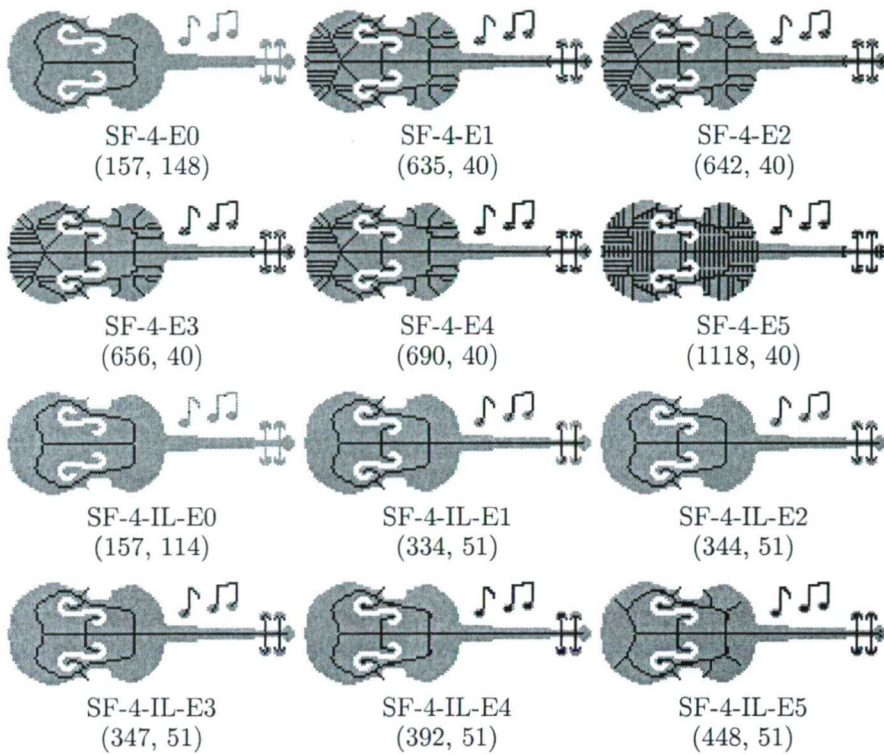


Figure 12: Two topological kernels and ten medial lines produced by the proposed 4-subfield algorithms.

Proof. If the 2-dimensional digital space is partitioned into 2 or 4 subfields as presented in Fig. 8, then it is easy to see that no 4-adjacent pair of simple points belongs to the same subfield. Hence, it is evident that if two simple points are 4-adjacent in the picture, then either of them can be deleted when the subfield is activated. If two 4-adjacent points remain simple after the final iteration step, then they must be endpoints of type E1, E2, E3, E4, or E5. \square

To summarize the properties of the presented algorithms, we state the followings:

- All the fifty-four algorithms are different from each other (see Figs. 4, 6, 7, 11, 12).
- All thinning algorithms with endpoint characterizations E4 and E5 may produce 2-point wide medial curves.
- The 2-subiteration algorithms (i.e., $SI-\langle NE, SW \rangle-\varepsilon$ and $SI-\langle NW, SE \rangle-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$) may produce a “distorted” asymmetric medial curves for symmetric objects (see Fig. 6).

- The 4-subiteration algorithms (i.e., $SI-\langle NE, SW, NW, SE \rangle-\varepsilon$ and $SI-\langle N, E, S, W \rangle-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$) can produce “almost symmetric” results for symmetric objects.
- The 4-subfield thinning algorithms $SF-4-\varepsilon$ ($\varepsilon \in \{E1, \dots, E5\}$) may produce numerous unwanted side branches. (That is why we proposed the thinning scheme with iteration-level endpoint checking.)
- Subfield-based algorithms $SF-k-IL-\varepsilon$ ($k=2, 4$; $\varepsilon \in \{E0, \dots, E5\}$) with iteration-level endpoint checking produce much less unwanted side branches than algorithms $SF-k-\varepsilon$ ($k=2, 4$; $\varepsilon \in \{E0, \dots, E5\}$) that use the conventional scheme (see Figs. 11 and 12).
- The fully parallel algorithms $FP-\varepsilon$ ($\varepsilon \in \{E0, \dots, E5\}$) require the least numbers of parallel reductions.
- The 4-subiteration algorithms that are taken the intermediate deletion direction into consideration (i.e., $SI-\langle NE, SW, NW, SE \rangle-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$) require less numbers of parallel reductions than the 4-subiteration ones considering the cardinal deletion directions (i.e., $SI-\langle N, W, S, E \rangle-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$).
- The 2-subfield algorithms (i.e., $SF-2-\varepsilon$ and $SF-2-IL-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$) require less numbers of parallel reductions than the 4-subfield ones (i.e., $SF-4-\varepsilon$ and $SF-4-IL-\varepsilon$; $\varepsilon \in \{E0, \dots, E5\}$).

In order to illustrate that our algorithms differ from the algorithms based on critical kernels, Fig. 13 presents three skeletons produced by algorithms AK^2 , MK^2 , and NK^2 [3].

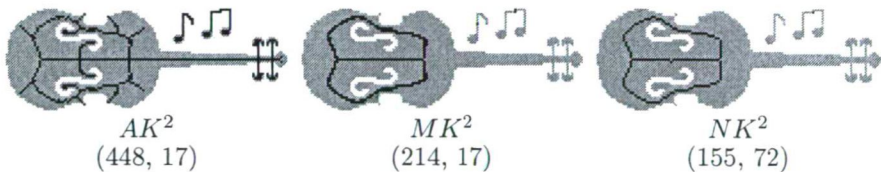


Figure 13: Skeletons produced by algorithms AK^2 , MK^2 , and NK^2 [3].

Unfortunately, there is no room to present here more examples, hence we invite the reader to visit the website at

http://www.inf.u-szeged.hu/~gnemeth/localweb/skeleton_alg2d.php, where skeletons produced by various existing algorithms are also presented.

5 Conclusions

This paper presents fifty-four topological algorithms for extracting skeleton-like shape features (i.e., topological kernels and medial lines) from binary objects. The major contributions of this work are:

- We proposed new sufficient conditions for topology preserving parallel reductions that are suitable for generating deletion rules for parallel topological algorithms.
- Fifty-four variations for parallel thinning and shrinking algorithms were constructed (each algorithm differ from the other ones). Deletion rules of the proposed algorithms were not given by matching templates (as it is usual), they were derived from our conditions.
- We introduced the 4-subiteration scheme with intermediate deletion directions, and the iteration-level endpoint checking in subfield-based algorithms.
- We proved that all the fifty-four algorithms produce minimal results for any pictures.

Acknowledgement

The authors would like to thank Prof. Michel Couprie (Département Informatique, ESIEE Paris, et Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est ESIEE 2, France) for providing the source code for algorithms based on critical kernels (<http://www.esiee.fr/~info/ck/>).

Finally, the authors would like to thank the reviewers for the comments and remarks that improved the quality of the paper.

References

- [1] Bertrand, G. On P-simple points. *Comptes-rendus de l'Académie des Sciences, Série Math.*, 1:1077-1084, 1995.
- [2] Bertrand, G. On Critical Kernels. *Comptes-rendus de l'Académie des Sciences, Série math.*, 1:363-367, 2007.
- [3] Bertrand, G. and Couprie, M. Two-dimensional Thinning Algorithms Based on Critical Kernels. *Journal of Mathematical Imaging and Vision*, 31:35-56, 2008.
- [4] Bertrand, G. and Couprie, M. On Parallel Thinning Algorithms: Minimal Non-simple Sets, P-simple Points and Critical Kernels. *Journal of Mathematical Imaging and Vision*, 35:23-35, 2009.
- [5] Couprie, M. Note on Fifteen 2D Parallel Thinning Algorithms. *Internal Report, Université de Marne-la-Vallée, IMG2006-01*, 2009.
- [6] Hall, R. W., Kong, T. Y., Rosenfeld, A. Shrinking Binary Images. In Kong, T. Y. and Rosenfeld, A., editors, *Topological Algorithms for Digital Image Processing*, pages 31-98, Elsevier Science B. V., 1996.

- [7] Hall, R. Parallel Connectivity-Preserving Thinning Algorithms. In Kong, T. Y. and Rosenfeld, A., editors, *Topological Algorithms for Digital Image Processing*, pages 145–179, Elsevier Science B. V., 1996.
- [8] Kong, T. Y. and Rosenfeld, A. Digital Topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48:357–393, 1989.
- [9] Kong, T. Y. *On Topology Preservation in 2-d and 3-d Thinning*. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 9:813–844, 1995.
- [10] Lam, L., Lee, S.-W., and Suen, C. Y. *Thinning Methodologies – A Comprehensive Survey*. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14:869–885, 1992.
- [11] Manzanera, A., Bertrand, T. M., Prêteux, F., and Longuet, B. nD Skeletonization: a Unified Mathematical Framework. *Journal of Electronic Imaging*, 11:25–37, 2002.
- [12] Manzanera, A. and Bertrand, T. M. *Metrical properties of a collection of 2D parallel thinning algorithms*. *Electronic Notes in Discrete Mathematics*, 12:255–266, 2003.
- [13] Németh, G. and Palágyi, K. Parallel Thinning Algorithms Based on Ronse’s Sufficient Conditions for Topology Preservation. Wiederhold, P. and Barneva, R. P., editors, *Progress in Combinatorial Image Analysis*, pages 181–192, Research Publishing Services, 2009.
- [14] Németh, G., Kardos, P., and Palágyi, K. Topology Preserving 3D Thinning Algorithms Using Four and Eight Subfields. In Campilho, A. and Kamel, M., editors, *Image Analysis and Recognition 2010, Part I, LNCS 6111*, pages 316–325, Heidelberg, 2010, Springer-Verlag.
- [15] Németh, G. and Palágyi, K. Topology Preserving Parallel Thinning Algorithms. *International Journal of Imaging Systems and Technology*, 21:3744, 2011.
- [16] Ronse, C. Minimal Test Patterns for Connectivity Preservation in Parallel Thinning Algorithms for Binary Digital Images. *Discrete Applied Mathematics*, 21:67–79, 1988.
- [17] Siddiqi, K. and Pizer, S. M., editors. *Medial Representations — Mathematics, Algorithms, and Applications*, Series in Computational Imaging, 2008, Springer.
- [18] Suen, C. Y. and Wang, P. S. P., editors. *Thinning Methodologies for Pattern Recognition*. Series in Machine Perception and Artificial Intelligence (8), World Scientific, 1994.

A Knowledge-Based Approach to Raster-Vector Conversion of Large Scale Topographic Maps*

Rudolf Szendrei[†], István Elek[‡] and Mátyás Márton[§]

Abstract

Paper-based raster maps are primarily for human consumption, and their interpretation always requires some level of human expertise. Today's computer services in geoinformatics usually require vectorized topographic maps. The usual method of the conversion has been an error-prone, manual process.

In this article, the possibilities, methods and difficulties of the conversion are discussed. The results described here are partially implemented in the IRIS project, but further work remains. This emphasizes the tools of digital image processing and knowledge-based approach.

The system in development separates the recognition of point-like, line-like and surface-like objects, and the most successful approach appears to be the recognition of these objects in a reversed order with respect to their printing. During the recognition of surfaces, homogeneous and textured surfaces must be distinguished. The most diverse and complicated group constitute the line-like objects.

The IRIS project realises a moderate, but significant step towards the automatization of map recognition process, bearing in mind that full automatization is unlikely. It is reasonable to assume that human experts will always be required for high quality interpretation, but it is an exciting challenge to decrease the burden of manual work.

Keywords: Geoinformatics, topographic maps, raster-vector conversion, artificial intelligence, knowledge representation

1 Introduction

Paper-based raster maps are primarily appropriate for human usage. They always require a certain level of intelligent interpretation. In GIS applications vectorized

*This research was supported by the project TÁMOP-4.2.1/B-09/1/KMR-2010-003 of Eötvös Loránd University.

[†]PhD student, E-mail: swap@inf.elte.hu

[‡]Department of Cartography and Geoinformatics, E-mail: elek@map.elte.hu

[§]Department of Cartography and Geoinformatics, E-mail: matyi@map.elte.hu

The authors are at the Faculty of Informatics, Eötvös Loránd University of Budapest, and they are members of the university research group in geoinformatics T.E.A.M. (<http://team.elte.hu/>)

maps are preferred. Especially, government, local authorities and service providers tend to use topographic maps in vectorized form. It is a serious challenge in every country to vectorize maps that are available in raster format. This task has been accomplished in most countries — often with the use of uncomfortable, “manual” tools, taking several years. However, it is worth dealing with the topic of raster-vector conversion. On one hand, some results of vectorization need improvement or modification. On the other hand, new maps are created that need vectorization. This is valid not only for topographic maps, but also for remote sensing images reflecting the status of agricultural areas.

The theoretical background of an intelligent raster-vector conversion system has been studied in the IRIS project [5]. Several components of a prototype system has been elaborated. It became clear very early that the computer support of conversion steps can be supported at quite different levels. For example, a map symbol can be identified by a human interpreter, but the recognition can be attempted with a software, using the tools of image processing. Therefore it is also valid that the level of intelligence of the raster-vector conversion system can be various. A computer system can be fairly valuable and usable even if every important decision of interpretation is made by the expert user. However, the system designed and developed by the authors is aimed at to automatize the raster-vector conversion as much as possible. This aim gives an emphasis to the knowledge-based approach.

Two types of expert knowledge in connection with maps are distinguished (see Fig. 1). First type consists of professional knowledge needed the interpretation of maps. This is required to derive an equivalent in vector format from a paper-based scanned topographic map. The basic level information is contained in a standard file. More sophisticated relationships are usually stored in a relational data base, connected to vectorized data. The other type of expertise consists of the knowledge needed to draw conclusions based on vector data model and the related data base contents.

There are several research results on the latter topic [1, 4]. Examples of questions to be answered are the area that will be inundated by water in case of a flood of a river, or how economic can the exploitation of an oil field. Beyond answers consisting of numerical data, expert systems often use visualization for better understanding [2].

It is important to realize that the adequacy of answers given by an expert system does not only depend on the inference rules applied [3], but also on the quality of stored data used as input of these rules [6, 10]. This also emphasizes the importance of map interpretation knowledge. In this paper knowledge-based approach means the first type of knowledge mentioned above, i.e. raster-vector conversion.

This paper deals with a part of raster-vector conversion applied in cartography, with knowledge-based approach. The types of map symbols used in topographical maps will be introduced, together with the algorithms used to recognize them. The organization of expertise into knowledge base will also be presented.

The following must be considered in connection with good quality and automated vectorization. Raster maps contain numbers, inscriptions and other kinds of data, but the majority of information is contained in a special cartographic

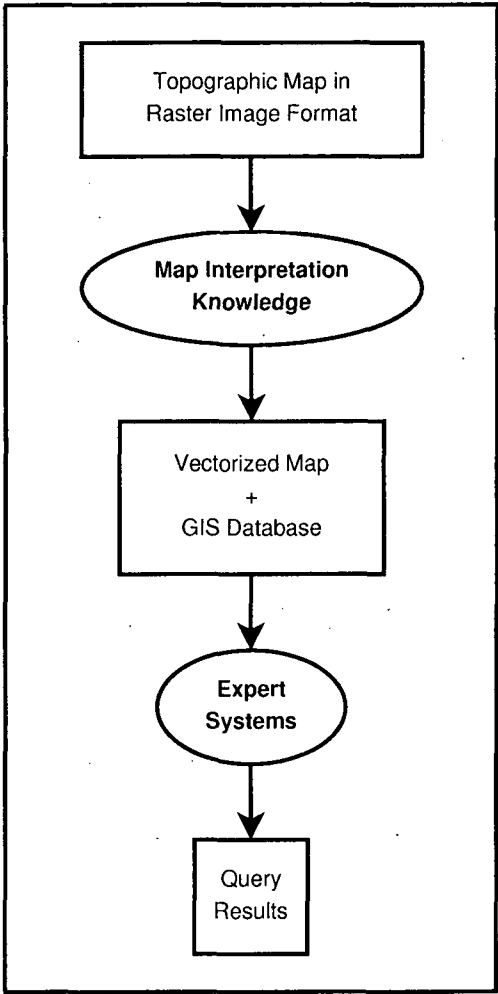


Figure 1: Knowledge representation in geoinformatics

context that can be adequately understood only by human expert. These relationships used for interpretation are no more contained in the vectorized map — it consists only of numerical and descriptive data. Vectorization necessarily involves some loss of information, this is why the depth of conversion must be carefully defined. Automatic interpretation of image contents requires sophisticated image processing tools, which are not comparable to human perception in the majority of cases. Therefore, the level of automatic recognition must also be appropriately determined.

2 Map symbols

The topic of this article is how to interpret printed variant of maps and how to represent them in computer systems. This process is considered basically as the result of interpretation and processing of map symbols. To accomplish this task it is very important to understand maps, and specifically, map symbols. To gain a comprehensive survey, refer to [8]. Although human cognition can not be completely understood, it is necessary to know to a certain extent how the human expert interprets graphical information. Regarding human perception, primarily points, lines and textured surfaces are sought and distinguished (see Fig. 2). It must be realized that human perception may reveal finer or hidden information, for example how roads crossing at different levels hide each other. Human mind is also capable of abstraction, for example when it disregards the actual texture of surface, and investigates only its shape. Human eye can make some corrections, for example in the determination of shades of color layers printed over each other.

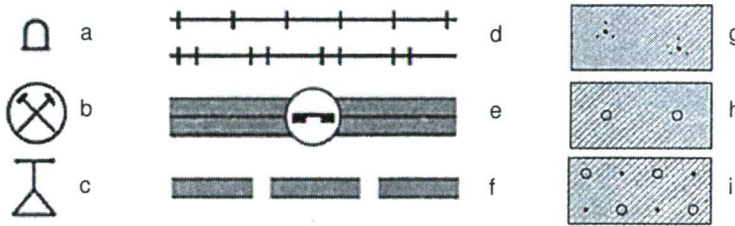


Figure 2: Point types: a) statue, b) mine, c) thunderhead. Line types: d) railway, e) highway with emergency phone, f) highway under construction. Area types: g) scrub, h) orchard, i) orchard with bushes.

Map interpretation process and the complexity of knowledge based object recognition can be visualized via any example of the four different object types — that is, point, line surface and inscription. IT tools seem to be capable of accomplishing map interpretation steps, but the extent that can be reached with human perception can not be approached by the IT technology of present days.

Point-like elements are usually graphical map symbols of small size, which often covers relatively larger area on map than the real object it represents. Nevertheless, its reference point can always be identified. Similarly, line-like elements can also cover larger area on map than their real size. For instance in the case of a highway, a zero-width center line can represent the theoretical position of the road in the database. Beyond the graphical properties of lines the database may contain real physical parameters, such as road width, carrying capacity, coating (concrete, asphalt) etc.

Hiding is a very inherent phenomenon in maps when line-like objects, landmarks (typically roads, railways and wires) located at different elevations intersect. This results in discontinuity of objects in map visualization. However, in map interpretation continuity must be assumed. Interpretation is not so straightforward in the

case of surface elements. The discontinuity of surface in map does not generally mean discontinuity of the real object it represents, e.g. in the case of a bridge over a river represented by water surface. However, a dam in the map may actually mean the separation of different levels of water.

The recognition and interpretation of inscriptions take place at two levels. Basically, the name represented by the inscription must be recognized, which may be accomplished with the support of a name data base. Interpretation means the establishment of appropriate connection between landmarks and their names. The recognition of inscriptions goes beyond the goals of this article.

3 Knowledge-based approach

One of the main tasks of geoinformatics is the conversion of available raster maps to vector format. Nowadays this is dominantly a manual task; it involves the tracing of polygon boundaries. This work is supported by the currently used software products, but the map interpretation is out of their scope. This is considered as a complex task in geoinformatics, which needs the intelligent application of various image processing algorithms. Our aim is to implement such a system that takes the majority of expert work, but it makes possible user interaction when it is necessary (see Fig. 3).

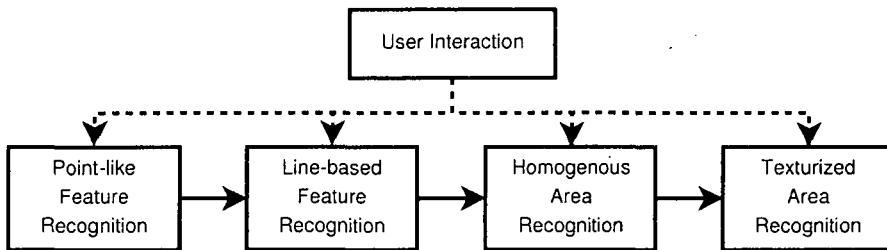


Figure 3: The flow-chart of the thematic map interpretation system.

In our approach, map symbols can be categorized into three groups according to their recognition algorithm. *Point-like* map symbols are used to mark objects that can be bound to a certain location (see Fig. 2 a – c) or land cover types, e.g. vineyard symbol. *Line-like* map symbols usually mean different types of roads and railroads, see Fig. 2 d – f. *Texturized surface* map symbols mean a texture covering a given type of land cover, see Fig. 2 g – i). Beyond these map symbols, extra care should be taken to recognize polygons delimiting areas.

The map representation of some objects involves several classes. For example, a lake is basically represented by a homogeneous blue surface, but its boundary is described by a line-type element. The case of buildings is similar, but their boundaries are more dominant than their (usually pink) surface.

One of the most difficult problems is the detection of rivers, as they can be represented either as line-like or surface element, depending on their width (e.g., in

case of a delta). In Table 3 below the categorization of some typical map objects can be seen.

Object	Point-like	Line-based	Texturized surface
Letter	X	–	–
Vineyard symbol	X	–	–
Road	–	X	–
Railway	–	X	–
House polygon	–	X*	–
River	–	X*	X
Lake	–	X*	X
Delta	–	X*	X
Field	–	X*	X
Forest	–	X*	X

Table 1: X - feature type can be recognized, * - boundary feature

In the following sections the knowledge-based algorithms used to recognize different map symbols will be presented.

In order to compile a map interpretation system from these algorithms, the order used during printing the map must be known, and the way of thinking of human interpreting the map must also be taken into account. As the first step an overview will be given on rules assembling maps, each resulting in an individual layer.

1. The boundaries of polygons are drawn.
2. Polygons are filled with a solid color or covered by a texture.
3. The road and railroad network is drawn, using their respective map symbols.
4. The map symbols of point-like objects are put onto the map.
5. Inscriptions belonging to point-like objects are printed.
6. Inscriptions belonging to line-like objects are printed following the arc of line — either next to the line or directly onto the line, omitting the section covered by the inscription.

During vectorization, these steps are executed in reverse order so as to collect map components. Map layers are apparently printed onto each other; top layers may hide elements belonging to layers beneath them. Within cartography, a more complex set of rules is used. Especially, conflicts may arise within one layer as well. For example, some characters may be omitted from inscriptions that intersect each other to avoid overlaps (map generalization).

4 Recognition of Point-Like Symbols

Point-like symbols are small objects (see Fig. 2 *a – c*), which usually also appear in the legend. They appear on the map undistorted, though they may be rotated.

In a previous article [11] the authors introduced an efficient, linear time algorithm for the recognition of point-like symbols (see Fig. 4), which is also capable of recognizing surface textures, since texture is nothing more than a point-like object repeated a number of times. We assume that for each map scale, an individual symbolset is used.

The following algorithm attempts to recognize all point-like objects.

1. Do edge detection (Canny or Laplace) on the whole map m .
2. For all possible symbols (appearing in the database or requested by the user) do the following:
 - a) Let s_{edge} the result of the edge detection applied on the current symbol s of size $s_x \times s_y$.
 - b) Let s_{otsu} the Otsu-thresholded image of s_{edge} .
 - c) Find the first pixel (u, v) of s_{otsu} , searching top-down, left to right, and determine the corresponding direction angle $s_{\Theta}(u, v)$ by the gradients calculated on s at (u, v) .
 - d) For all edge pixel $m(x, y)$ of the transformed map image determine the corresponding direction angle $m_{\Theta}(x, y)$ by the gradients calculated on m at (x, y) and perform the following steps:
 - i. Rotate the original raster image of the point-like object around point $s(u, v)$, by the angle $m_{\Theta}(x, y) - s_{\Theta}(u, v)$ to get S_{mat} , that is the matrix representation of the symbol rotated.
 - ii. From the values of S_{mat} subtract the values of the underlying map pixels considering $m(x, y)$ as the origin, to get the difference matrix D_{mat} .
 - iii. Calculate the standard deviation σ of D_{mat} .
 - iv. If σ is smaller than a given threshold, then the point-like symbol is recognized with coordinates $(x - u + s_x/2, y - v + s_y/2)$. The coordinates and the type of symbol are exported into a file or database.

	Laplace-I	Laplace-II	Prewitt	Sobel
Map	23.05%	29.07%	20.72%	27.89%
Symbol	15.63%	21.88%	32.81%	35.94%

Table 2: Ratio of edge pixels after an Otsu-thresholding is applied.

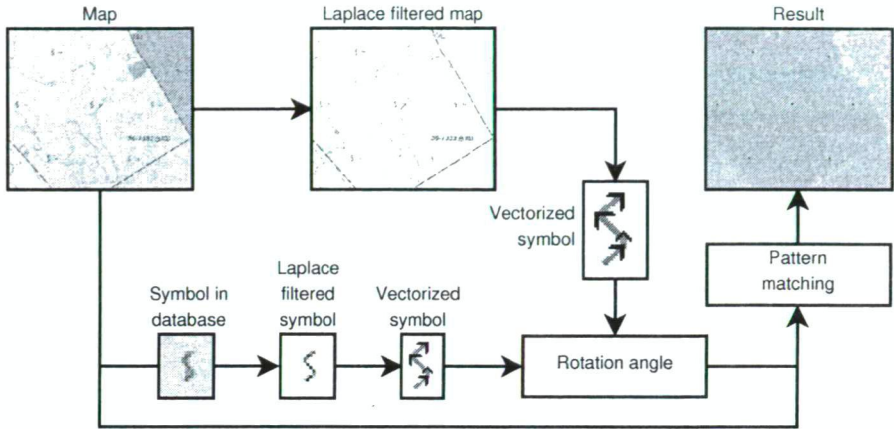


Figure 4: The flowchart of the recognition algorithm for a given symbol. The result image and the filtered images are in negative for better visibility. Each black point on the result image represents a recognized symbol.

The algorithm does edge detection to omit those map points, where pattern matching is unnecessary. The edge detection is also needed to determine which pixel of a symbol should be matched to an edge pixel of the map. We made Otsu-thresholding on the results of four different edge detectors and counted the remaining edge pixels (see Table 2) to decide which edge detector is the most useful. We chose Laplace-I, because it gives less points where pattern matching has to be made. It is also important to choose the interpolation method to symbol rotation. Simple point sampling interpolation produces unneeded distortion, since it only chooses the closest neighbour to a pixel. Bilinear and bicubic interpolation give better results (see Fig. 5). These interpolation methods are also considering the surrounding pixels. We found that bicubic interpolation keeps more image details, like edge features.

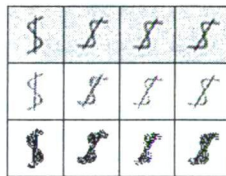


Figure 5: Top row: Original image, and its *Point sampling*, *Bilinear* and *Bicubic* interpolated rotations. Middle row: Results of Laplace-I. Bottom row: Results of Otsu-thresholding.

The method is compared to a rotation-invariant pattern matching method [12] based upon color ring-projection. That gives a high recognition rate (about 95%).

The authors wrote that “Computation time of the proposed color ring-projection matching is 6 s on a Pentium 300 MHz personal computer for an arbitrarily rotated image of size 256 x 256 pixels and a circular window of radius 25 pixels.” Nowadays, the algorithm may run on a Core i7 920 processor (without specific instruction sets, like SSE, etc.) approx. 35 times faster, but we have 100 megapixels resolution topographic maps. This gives us a runtime of 261 seconds in the case of a multi-threaded implementation. Because of the topographic symbols are simple graphics (versus natural images), we can perform the recognition with our parallelized algorithm in approx. 2-3 s on the same machine with a recognition rate > 99%. In practice, the point-like symbols of a high detail topographic map can be vectorized manually in approx. 1 hour. We note that a great similarity of a point-like symbol and a map region can lead to a false positive match. These false positives can be easily removed manually after vectorization, as we did. In the future, we try to automatically remove the false positives based on line filtering methods, e.g. Hough transformation, Canny etc.

5 Recognition of Line-Like Symbols

Line-like symbols are usually the trace of a road like object, or edge of a polygon with a given texture/attribute.

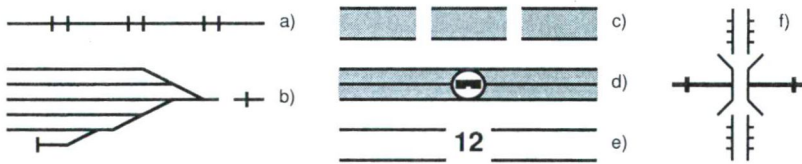


Figure 6: Examples for line-like symbols: a) railway b) railway network at a railway station, c) highway under construction d) highway with emergency phone. e) road with a specified width f) bridge above a canal

Recognition of line-like symbols is one of the most difficult tasks of raster-vector conversion. These symbols are often complex, and it is permitted for two symbols to differ only in their size, to cross each other or to join to form a single object (see Fig. 6 a, b, respectively). Difficulties are posed by parallel lines belonging to the same object (see Fig. 6 d, e) versus lines running in parallel which belong to separate objects. Further difficulties are the discontinuous symbols (see Fig. 6 c, e, f).

It is beyond the aim of the current article to solve all the difficulties mentioned above, so for the purpose of this paper we assume that line-like symbols

1. do not cross each other, and
2. do not join to form a single object, and
3. are continuous.

A classic way of line-like symbol vectorization is introduced in [7], where cadastral maps in binary raster image format are vectorized. The additional features of color topographic maps, like road width, capacity, coating etc. can not be recognized the classical way [9]. Each of these features are represented by a corresponding graphics, color and structure. The following method is able to recognize the trace of the line-like symbols of a topographic map.

1. Do image segmentation and classify each pixel.
2. Create a binary map, where each black pixel belong to eg. road.
3. Apply thinning and morphological thinning on the binary map.
4. Vectorize the one pixel thin skeletons.

The first step is the segmentation, which works as follows. Define an object color set O and a surface color set S . The amount of colors in each color set is approx. 5-7 in the case of topographic maps. We assume that on a printed map each pixel color can be defined as a linear combination of a surface and an object color. In optimal case, this can be written as a $c = \alpha * c_o + (1 - \alpha) * c_s$ equation, where c is the value of the current pixel, and c_o, c_s are the respective object and surface colors, so the segmentation can be done by solving the minimalization task $\min_{o \in O, s \in S} |c - \alpha * c_o + (1 - \alpha) * c_s|$ for each pixel.

As the second step is a simple selection on the segmented pixels, it can be done easily. The third step consists of two different thinning methods. A general thinning method is used first to avoid creating unneeded short lines by morphological thinning. The general thinning can be described as it iteratively deletes pixels inside the shape to shrink it without shortening it or breaking it apart.

P ₉	P ₂	P ₃
P ₈	P ₁	P ₃
P ₇	P ₆	P ₅

Table 3: 3×3 binary matrix, and the indices of its elements.

To decide whether an edge pixel P_1 should be deleted, sider its 8 neighbors in the 3 by 3 neighborhood (see Table 3), P_2, P_3, \dots, P_8 and P_9 and define:

- $N(P_1)$: number of non-zero neighbors ($N(P_1) = P_2 + P_3 + \dots + P_9$).
- $S(P_1)$: number of 0 to 1 (or 1 to 0) transitions in the sequence P_2, P_3, \dots, P_9 .

The meaning of the values:

- $N(P_1) = 0$ (an isolated point)
- $N(P_1) = 1$ (tip of a line)

- $N(P_1) = 7$ (located in concavity)
- $N(P_1) = 8$ (not a boundary point)
- $S(P_1) \geq 2$ (on a bridge connecting two or more edge pieces)

Repeat the following steps, until no more change can be made

1. Mark all pixels satisfying all of the following: ($P_1 = 1$) and ($2 \leq N(P_1) \leq 6$) and ($S(P_1) = 1$) and ($P_2 * P_4 * P_6 = 0$) and ($P_4 * P_6 * P_8 = 0$) and ($P_7 \neq 0$).
2. Delete all marked pixels.
3. Mark all pixels satisfying all of the following: ($P_1 = 1$) and ($2 \leq N(P_1) \leq 6$) and ($S(P_1) = 1$) and ($P_2 * P_4 * P_8 = 0$) and ($P_2 * P_6 * P_8 = 0$) and ($P_3 \neq 0$).
4. Delete all marked pixels.

Because the result of the above algorithm may contain small pixel groups, a morphological thinning should be performed. This morphological thinning can be done by using the structuring elements shown in Table 4. At each iteration, the image is first thinned by the left hand structuring element (see 1st element of Table 4), and then by the right hand one (see 2nd element of Table 4), and then with the remaining six 90° rotations of the two elements. The process is repeated in cyclic fashion until none of the thinnings produces any further change. As usual, the origin of the structuring element is at the center.

0	0	0
	1	
1	1	1

	0	0
1	1	0
	1	

	1	
1	1	1

	1	
	1	1

Table 4: The first and second structuring elements are used to morphological thinning based skeletonization, while the third and fourth structuring elements are used to morphological fork detection on binary images. Values of the elements are: 0 - background, 1 - foreground. Empty places can be either 0 or 1.

The skeletonized binary image can be vectorized in the following way. Mark all object pixels *black* and surface pixels *white*. Mark those *black* pixels *red*, where $N(P_1) > 2$, and then mark the remaining *black* fork points *blue* by using the 3rd and 4th structuring elements of Table 4 in the same way as structuring elements are used in morphological thinning. The *red* fork points are connecting lines, while *blue* fork points are connecting forks. Mark *green* each *black* pixel, if at most one neighbour of it is *black* (tip of a *black* line). It can be seen that a priority is defined over the colors as *white* < *black* < *green* < *red* < *blue*. The following steps vectorize the object pixels

1. Select a *green* point, mark *white* and create a new line segment list, which contains that point.
2. Select a *black* neighbour if it exists and if the current point is also *black*. Otherwise select a higher priority point. Mark *white* the point and add to the end of the list.
3. Go to Step 2, while a corresponding neighbour exists.
4. Go back to the place of the first element of the list and go to Step 2. Be careful that new points should be added now to the front of the list. (This step processes points in the opposite direction.)
5. Go to Step 1, while a *green* point exists.
6. Select a *black* point, mark *white*, and create a new line segment list, which contains that point.
7. Select a *black* neighbour of the current point, mark *white*, and put it at the end of the list.
8. Go to Step 7, while a *black* neighbour exists.
9. Select a *red* point p , mark *white* and create a new line segment list, which contains that point. Let $NeighbourSelect = RedSelect = Counter = 0$, $BlueFirst = false$, $where = back$, $q = p$.
10. Let $PrevPoint = q$.
11. If the $NeighbourSelect$ th neighbour r of q exists, let $q = r$, let $BlueFirst = (Steps = 0 \text{ and } where=back)$, let $n = q$, and increment $NeighbourSelect$ by 1. Put q into the list at $where$ and go to Step 13.
12. If the $RedSelect$ th neighbour r of q exists,
 - a) If q and n are neighbours and $where = front$, then let $q = PrevPoint$ and increment $RedSelect$ by 1. Go to Step 10.
 - b) Put q into the list at $where$, mark q *white*, let $NeighbourSelect = 0$ and increment $Counter$ by 1. Go to Step 10.
13. If $where=back$, then let $where=front$, $q = p$ and go to Step 10.
14. Go to Step 9, while a *red* point exists.

Although, the algorithm above vectorizes all the objects, it merges the several object types and colors. Hence, pixels of a given object color are copied onto a separate binary image before they are vectorized.

We introduce an approach, which is able to recognize the features of line-like objects, so the corresponding attributes can be assigned to them. This assumes

that the path of each object exists in the corresponding vector layer. In order to recognize a specific feature, its properties should be defined for identification.

Two properties of vectorised symbols are recognized: forks ($F \sim Fork$), and end-points ($E \sim End$). Both are well known in fingerprint recognition where they are called *minutiae*. In the case of fingerprints, a fork means an end-point in the complement-pattern, so only one of them is used for identification. In our case, we can not define a complement-pattern, so both forks and end-points are used.

Representation of line-like symbols is based on weighted, undirected graphs. An *EF*-graph is an undirected graph with the following properties:

- Nodes are either of type *E* or *F*. The color of a node is determined by the corresponding vector layer.
- Two nodes are connected if the line-segment sequence connecting the nodes in the corresponding vector layer does not contain additional nodes. Edges running between nodes of different colors can be defined by the user (in case of multicolor objects). The weight of the edge is equal to the length of the road connecting the two nodes, and it has the color of the corresponding symbol part.
- There are special nodes, denoted by an index *P*, which occur on the trace of a line object. These will be used to produce the final vector model.

An *EF*-graph can also be assigned to the vectorised map, not only to the vectorised symbols, where line-like symbols are not separated to their kernels. For recognition we use the smallest units of the symbol, called the kernel. The smallest unit is defined as the one which can be used to produce the entire symbol by iteration. In the *EF*-graph there is usually only two nodes participating in the iteration; these are type *F* with only a single edge, so become the entry and exit points to the graph. In the very few cases, where the entry and exit points of the smallest unit can not be identified, the kernel of the line-like object is itself. Smallest unit can not be defined for the whole vectorised map.

Figure 7 shows how a symbol is built up from its smallest units by iteration. Weights represent proportions and depend on the scale of the map. Beside weights, we can assign another attribute to edges, their color. In the figure almost all edges are coloured black.

The recognition of line-like objects is reduced to an extended subgraph isomorphism problem: we try to identify all the occurrences of the *EF* graph of the symbol (subgraph) in the *EF* graph of the entire map. The weights of the *EF* graphs are normalized with respect to the scale of the map, and the collection is sorted in decreasing order of node degrees. Call this collection of sorted *EF* graphs *S*. Since the *EF* graphs created to maps do not contain the edges those connecting nodes with different colors, this case should be handled. In this article, the potential edges are identified by searching the corresponding neighbour on its own layer in the given distance of the node. The validity of a found potential edge is verified by

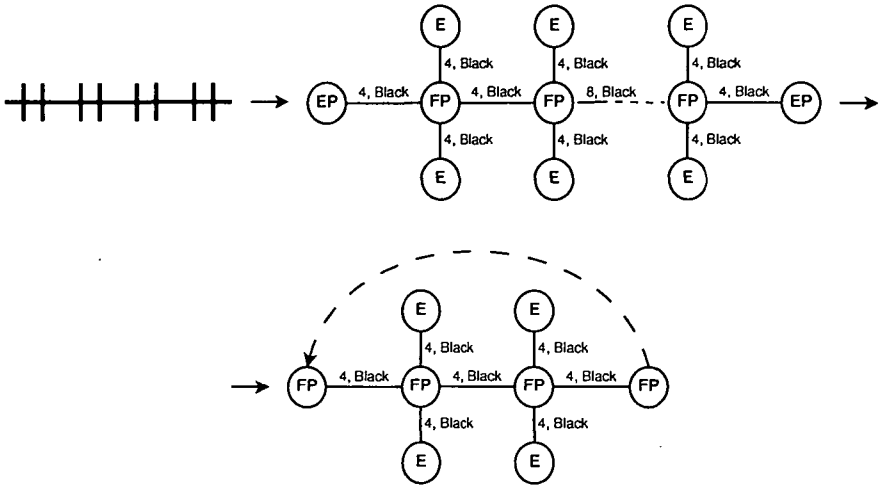


Figure 7: The EF graph and the elementary EF graph of a double railway line. Distances are relative values and refer to the scale of the map. The dashed line in the elementary EF graph represents its cyclical property.

comparing the color of the edge and the color of the segmented image pixels lying *under* the edge.

Subject to the conditions above it is possible to design an algorithm for the recognition of subgraphs. While processing the map, recognized objects are removed, by recoloring the corresponding subgraph. Two colors, say blue and red, can be used to keep track of the progress of the algorithm and to ensure termination.

The following algorithm stops when there are no more red nodes left in the graph.

1. Choose an arbitrary red node U with the highest degree from the EF graph of the map.
2. Attempt to match the subgraph at node U against S , that is the sorted collection of EF graphs, until the first successful match in the following way:
 - a) Perform a “parallel” breadth-first search on the EF graph of the map and the EF graph of kernel of the current symbol with origin U . This is called successful if both the degree of all nodes match, and weights are the same approximately.
 - b) In the case of success, all matching nodes become blue, otherwise they remain red.

Upon successful matching the EF -graph of the symbol is deleted from the EF -graph of the map. Entry and exit points must not be deleted unless they are marked as E , and the degree of remaining F nodes must be decreased accordingly. The

equality of edge weights can only be approximate, due to the curvature of symbols. The algorithm above can be implemented, as it keeps track the given object by using the line segments as paths in vector data. The other difficulty is that edges with differently colored nodes are not directly defined by the vector layers. In practice, we have created a spatial database, which has contained the vectorized line segments and their color attribute. The potential edges was determined by a query, looking for existence of a neighbour with a corrspring color in a given distance from the corresponding node.

6 Recognition of Homogeneous Surfaces

Although, some surface vectorization methods [5, 6] are working well on homogeneous surfaces, these methods are not able to deal with visually separated surface regions. During the recognition of surfaces, the most frequent tasks are the unification of areas which logically belong together, but are visually separated, and correction of map errors. For this purpose we use a mask, which identifies the points in the map which can be considered part of a surface. The mask can be defined as the area which remains after the removal of point and line-like symbols. Another possibility is to consider a mask to be the pixels of the appropriate color that remain after color segmentation on the original map. Pixels belonging to the mask get the color of the surface, others are set to be UNDEFINED. For best results, we use both approaches combined. For recognition and high-quality polygonization we defined a heuristic rule system, which we discuss next.

6.1 Removal of the Pixels of False Surfaces

According to the rule, we remove all pixels, and groups of pixels of a given surface type if they occur less frequently then a given treshhold. Many misclassified false surfaces are removed by this rule.

Surface type	Searching region size	Minimal pixel occurence
e.g. Grass	$n \text{ px} \times n \text{ px}$	$m \text{ pcs}$

Table 5: An example to remove false surface pixels.

In the example (see Table 5), a given pixel is classified as e.g. grass, if and only if, in the surrounding $n \times n$ square there are at least m pixels that can be classified as grass. At the boundaries of the image, where less pixels exist, the treshhold is decreased appropriately. The rule is applied top-down, left to right. The process is not adaptive in the sense that mask is developed as a new image, while the original map remains unchanged.

6.2 Classification of Delimited Surfaces

On topographical maps it is usual practice that certain areas are delimited with a line, which forms a polygon. This helps recognition if we know that the polygon exclusively surrounds a homogeneous area of given type. In such cases, as soon as we determined a dominant surface we can classify the whole surface within the polygon. For example, in the case of a house, the color of the delimiting polygon is always black, and the surface within the polygon is pink. To determine if the polygon surrounds the given area we use the usual “flooding” technique from any inner point. In practice, this needs a threshold (see Table 6) to limit “flooding”.

Border	Blob type	Flood limit
e.g. Black	e.g. House	n px

Table 6: An example of the classification rule.

If flooding does not stop under the specified threshold n , we terminate and classify the area as UNDEFINED.

6.3 Joining Surfaces

The removal of point and line-like objects result in discontinuity of surfaces. We need a separate rule to join these (see Table 7), which describes what to do if there is an unexpected pixel during the top-down, left-right processing of the mask. From the given point, that is the occurrence of the first unexpected pixel, it searches within a predetermined limit n , in opposing directions for surfaces which have been classified as of the same type. If search succeed in finding such surfaces, then the color of the original pixel is changed to that of the surrounding surface. This process can be repeated m times or until no further change has made.

Blob type	Searching radius	Iterations
e.g. Grass	n px	m

Table 7: Rule to fill gaps between surfaces.

6.4 Removal of Small Damaged Areas and Holes

Small errors can be often found on maps. In a clearly defined area, there might be spots which apparently do not belong there. Forests, for example, often contain empty areas yet the whole should be classified as one. Another source of errors, is when the type of a small area is clearly identified, yet, due to its small size or its context this classification can not be accepted. In these cases, the anomalous pixels are removed. Table 8 gives further details.

The rules given above are mostly complete, in the sense that most pixels and areas are classified into one of the known types. Yet, unclassified areas may remain.

Blob type	Bounding type	Blob size limit	Decision
Non-forest	Forest	n px	$Type := Forest$
Forest	Non-forest	m px	Eliminate

Table 8: Eliminate blobs and holes on a forest layer.

Most of the time, those unclassified areas are surrounded by many areas of *different* types, so the techniques above fail. Table 9 shows a rule set which helps under such circumstances, but interestingly it also helps to reclassify previously classified areas. The table specifies for which pixels or areas should the rules apply, which surrounding areas are dominant, which areas are sub-dominant, and what other types must occur nearby. In other words, Table 9 is a higher level rule set, providing a sort of context sensitivity to the classification process. The distance limit can be interpreted as the maximum of the smallest diameter of the discontinuity, and the size limit determines the largest discontinuity which can be filled. If at least one of the conditions of the rule is satisfied the area can be reclassified to the dominant type.

Type	Dominant type	Sub-dominant types	Dist. limit	Size limit
Unknown	Grass	Set(Forest, Vineyard)	n px	m px

Table 9: Rule to fill multi-bounded blobs, pixels and holes.

The rules specified in 9 specify for which areas a rule should apply. Often what is needed is to exclude a rule to be applied to a given surface. Such a case is when we are attempting to eliminate discontinuities between two areas, but we want to avoid considering a river as a discontinuity. To handle such situations, we need to provide a mask for the areas we wish to retain. Only the areas of interest get an appropriate type, others get UNDEFINED. Then we unite the results of the original and the temporary mask by copying the defined types from the later onto the former. Table 10 provides the details.

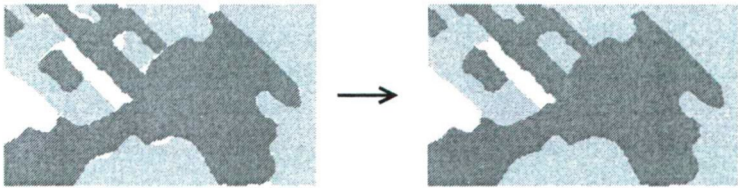


Figure 8: Result of using rules for filling multi-bounded blobs, pixels and holes.

While a human expert is able to manually vectorize map anomalies, like discontinuity and additional hidden information, the raw segmentation is unable to recognize them.

Surface type	Type of dominant surface	Type of sub-dominant surface
House	House	Grass

Table 10: A. Surface type selection. B. Combine grass and house surfaces, preferring houses to “draw” onto the grass layer.

	Surface	Correct	False ⁺	False ⁻	Time
Segmentation results	Field	88.75%	0.93%	11.25%	0ms
	Forest	84.59%	3.53%	15.44%	2460ms
	Vineyard	51.23%	443.01%	48.76%	2460ms
	Meadow	85.67%	1.87%	14.34%	3100ms
Reconstruction results	Field	98.17%	3.77%	1.83%	57ms
	Forest	96.96%	0.78%	3.04%	4927ms
	Vineyard	63.01%	25.93%	36.99%	5044ms
	Meadow	97.31%	1.19%	2.69%	8964ms

Table 11: Segmentation is done by the method described in section Recognition of Line-Like Symbols. Runtime and quality were measured on a topographic map of scale 1:10000. Results were compared to the result of the manual vectorization.

As it can be seen on Table 11, the reconstruction algorithms decreased the ratio of false positives, improved the vectorization accuracy and “recognized” some hidden information, which could be recognized earlier only by a human expert.

7 Recognition of Texturized Surfaces

Texturized surfaces are made up of repetition of smaller images, which are never rotated. The algorithm we described for the recognition of point-like symbols can be generalized (and at the same time specialized by removing the rotation steps) to simultaneously recognize multiple point-like symbols.

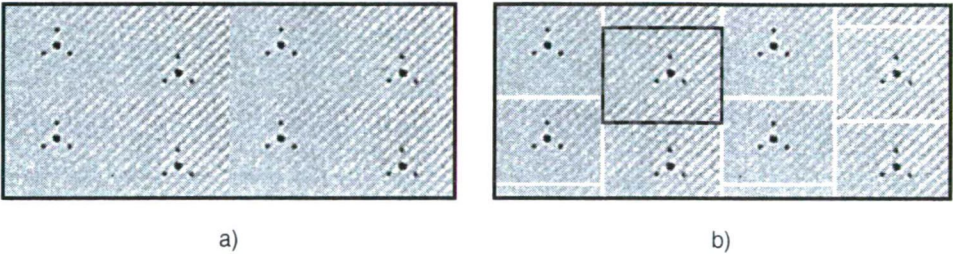


Figure 9: Texture, representing a region covered by scrub: a) texture, b) a texture tiled with subtextures showing the kernel.

It is assumed that, besides the point-like symbols, raster data for all textured surfaces, with the appropriate scale, are available and can be uniquely identified by their index. The algorithm of Section 4 will now be applied to recognize texture kernels instead of point-like objects. The modifications required are as follows:

1. Prepare a binary mask with the same size as the original map, and initialize all elements of it to FALSE.
2. Prepare a surface mask with the same size as the original map and initialize all elements of it to UNDEFINED. Each element of the surface mask is an index, which uniquely identifies a surface type or remains UNDEFINED.
3. Perform thinning on the map (for example Laplace).
4. Order the texture kernels according to their standard deviation, and call it TKA.
5. Process those pixels of the map which are currently unidentified (their index is UNDEFINED) in a top-down left to right fashion, which means performing the following steps:
 - a) Assign TRUE to all elements of TKA (meaning no matching has been attempted).
 - b) Calculate the standard deviation σ_c of the current map segment, that is the image starting at the pixel identified in step 5.
 - c) In TKA find the closest match to σ_c , for which its boolean entry is TRUE.
 - d) Match the two pieces of images within the given standard deviation threshold.
 - e) If they match, assign them matching texture type to each element of the surface mask for the area under consideration, and set the corresponding elements of the binary mask to TRUE. Then continue from step 5.
 - f) If there is no match, mark the texture kernel with FALSE and continue from step 5.c.

On the edges of textured surfaces, the kernels are usually not complete. In such cases, the algorithm as it stands is not capable of successful recognition. Those fractions can usually be identified by surrounding texture rectangles.

8 Conclusion

In this article, the results of the IRIS projects are discussed. The project aims to automate and support the recognition of raster images of topographic maps, with the combination of digital image processing and a knowledge-based approach. The developed system contains the basic knowledge of the classification of symbols to point-like, line-like and surface-like. From the point of recognition, there is an algorithmic difference between the recognition of homogeneous surfaces and textured surfaces.

The robustness of IRIS is based on the insight that layers of symbols are recognized (and “removed”) in reverse order of their printing. The interpretation of line-like symbols is the most difficult, and it can not be surprising that the task requires the heaviest mathematical machinery. *EF* graphs are used for the recognition of curved, line-like objects with a systematic pattern.

Rules of knowledge-based systems also appear in IRIS, and numerous rules are used for the recognition of homogeneous surfaces.

It is widely accepted, that knowledge-based systems do not match the quality of human experts, yet, there are areas where automation is desirable and produces better results. Within its own limits a knowledge-based system is reliable, it is neither superficial nor does it makes mistakes. Both of these qualities have exceptional value in the case of maps with rich structure. While human interpretation and manual vectorization are error-prone, an automated system can process vast amount of details.

Experiences with IRIS clearly demonstrates that human experts will always be required, yet it is an exciting challenge to decrease the amount of repetitive, and error-prone tasks.

While most point-like objects are recognized ($> 99\%$), some works remain to eliminate the false positives. We experienced the same difficulties when we tried to recognize the texturized surfaces. We have shown also a feature recognition method to line-like symbols. We found that these features could be recognized well ($> 80\%$), which speeds up the later manual postprocessing. It removes the recognized features and assigns the attributes of the feature to the corresponding object. Furthermore, we developed a method to deal with the hidden surface objects. Formerly, these objects could be identified only by a human expert. As it can be seen on Table 11, our knowledge-based method is able to make deductions by predefined rules to refine the segmentation results and recognize surface discontinuities and hidden surface regions.

We tested our methods on hungarian topographic maps at scale 1:10000. We have scanned the maps at 300dpi resolution, which produced ca. 100 megapixels bitmaps. The recognition time for each point-like symbol was 3-4 seconds, while the recognition of the line-like object features on the whole map took 15-18 seconds. The surface reconstruction took 5-6 seconds for each surface layer. The total vectorization time was less than 2 minutes, compared to a vectorization made by a human expert, which takes many hours.

References

- [1] Baik, S., et al. A naive geography analyst system with cognitive support of imagery exploitation. In Monroy, R., et al., editor, *Lecture Notes in Artificial Intelligence*, number 2974, pages 40–48, 2004.
- [2] Buttenfield, B. P. and McMaster, R. B., editors. *Map generalization: Making rules for knowledge representation*. Longman Scientific & Technical, 1991.
- [3] Chen, H., et al. A geographic knowledge representation system for multimedia geospatial retrieval and analysis. *International Journal on Digital Libraries*, 1(2):132–152, September 1997.
- [4] Corner, R. J. *Knowledge Representation in Geographic Information Systems*. PhD thesis, Curtin University of Technology, December 1999.
- [5] Dezső, B., Elek, I., and Máriás, Zs. IRIS, Development of Automatized Raster-Vector Conversion System. Technical report, Eötvös Loránd University and IKKK, November 2007. in Hungarian.
- [6] Dezső, B., Elek, I., and Máriás, Zs. Image processing methods in raster-vector conversion of topographic maps. In Karras, A. D., et al., editor, *Proceedings of the 2009 International Conference on Artificial Intelligence and Pattern Recognition*, pages 83–86, July 2009.
- [7] Janssen, R. D. T. and Vossepoel, A. M. Adaptive vectorization of line drawing images. *Computer Vision and Image Understanding*, 65(1):38–56, January 1997.
- [8] Klinghammer, I. and Papp-Váry, Á. *Földünk tükre a térkép (Map, mirror of the Earth)*. Gondolat, 1983.
- [9] Liang, SC. and Chen, WJ. Extraction of line feature in binary images. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E91A:1890–1897, August 2008. Issue 8.
- [10] Samet, H. *The Design and Analysis of Spatial Data Structures*. Addison - Wesley, 1990.
- [11] Szendrei, R., Fekete, I., and Elek, I. Texture based recognition of topographic map symbols. In Karras, A. D., et al., editor, *Proceedings of the 2009 International Conference on Artificial Intelligence and Pattern Recognition*, pages 7–10, July 2009.
- [12] Tsai, D. and Tsai, Y. Rotation-invariant pattern matching with color ring-projection. *Pattern Recognition*, 35(1):131–141, January 2002.

Projection Selection Dependency in Binary Tomography*

László Varga^{†‡}, Péter Balázs[‡] and Antal Nagy[†]

Abstract

It has already been shown that the choice of projection angles can significantly influence the quality of reconstructions in discrete tomography. In this contribution we summarize and extend the previous results by explaining and demonstrating the effects of projection selection dependency, in a set of experimental software tests. We perform reconstructions of software phantoms, by using different binary tomography reconstruction algorithms, from different equiangular and non-equiangular projections sets, under various conditions (i.e., when the objects to be reconstructed undergo slight topological changes, or the projection data is affected by noise) and compare the results with suitable approaches. Based on our observations, we reveal regularities in the resulting data and discuss possible consequences of such projection selection dependency in binary tomography.

Keywords: discrete tomography, reconstruction, adaptive projection acquisition, GPU-accelerated computing, non-destructive testing

1 Introduction

The main goal of transmission tomography is to reconstruct the inner structure of given objects from a set of their projections. This is usually done by exposing the object of study to some electromagnetic or particle radiation at one side and measuring the amount of received energy on the other end. After the projections have been gathered, one can apply certain reconstruction algorithms for discovering the linear attenuation coefficients of the object at its different points.

The reconstruction can be performed in many ways. In the ideal case (when we have hundreds of projections available) one can use, e.g., the filtered backprojection

*This research was in part supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency, by the TÁMOP-4.2.1/B-09/1/KONV-2010-0005 project co-financed by the European Union and the European Regional Development Fund, and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

[†]Department of Image Processing and Computer Graphics, University of Szeged, Szeged, Hungary, E-mail: {varga.l, pbalazs, nagy.a}@inf.u-szeged.hu

[‡]Corresponding author.

method or other continuous techniques for finding the reconstruction of arbitrary objects [7, 12]. Unfortunately, acquiring hundreds of projections is sometimes impossible, since taking too many of them can be expensive or can even damage the object of study. In this case, one can try to improve the quality of the reconstruction basically in two ways. One approach is to take the projections having the highest information content in order to get sufficient information from fewer projections [15]. Another common approach is to develop more accurate reconstruction algorithms by using some prior information of the objects of interest.

In discrete tomography [8, 9] we assume that the object to be reconstructed consists of only a few (usually 2 to 4) materials, having known attenuation coefficients. With such a strong prior information, algorithms have been developed capable of producing accurate reconstructions from a limited amount of (say, up to 10) projections. However, the low number of projections gives a relatively high freedom in choosing the directions to take projections with.

Our previous studies [14, 17, 18, 19] revealed that different projection sets of an object can have entirely different information content, some holding more or less information than others. Despite their good performance, discrete reconstruction algorithms still require a certain amount of information to produce an accurate result, otherwise there can be numerous possible solutions, and among them just one is considered to be correct. This makes finding the proper angles essentially important in the case of discrete tomography, since we can get entirely different reconstructions from projection sets with even the same number of projections.

There are also theoretical results, giving upper bounds to the number of required projections in case of reconstructing convex objects [6]. Also, one of our long-term goals is to discover a more general description on how determined a binary reconstruction can be by a given set of projections.

The main aim of our current study is to determine, whether the result of a reconstruction can be improved by finding the correct directions to take projections with. If the result can be improved in such a way, one can develop more accurate reconstruction techniques, capable of providing better reconstructions exclusively by finding the appropriate projection angles.

In this paper, we summarize and extend the previous results connected to the angle selection dependency in binary tomography. We conduct experimental software tests on phantom images by reconstructing them from different equiangular and non-equiangular sets of their projections and compare the resulting reconstructions. We do experiments by applying three different binary reconstruction algorithms, and examine the effects of modifications of data, i.e., when the projections are corrupted by noise, or the object to be reconstructed is slightly altered (for example it has fractures or unwanted holes in it). The novelty of our studies lies in examining the case, when we allow the usage of non-equiangular projection sets and in the same time assume presence of distortions of the data. Furthermore, we will give a brief description of the nature of the problem, and describe some of its possible consequences.

Although, we do mention some angle selection strategies in this paper, we must highlight that our goal is not to propose a new reconstruction algorithm, but only

to evaluate the direction-dependency of three currently existing ones. We must also note that – although our results contain an explicit evaluation of the performance of the three selected reconstruction algorithms – we do not intend to compare them and decide which one is the best.

The paper is structured as follows. In Section 2 we introduce a formulation of the transmission tomography problem, and in Section 3 we present algorithms for solving it in the binary case. In Section 4 we describe the test frameset used for our experiments. In Section 5 we give some of our results, and provide an explanation of them. Finally, in Section 6 we conclude our work, and suggest some possible extensions of it.

2 Transmission Tomography

In this chapter we will provide a model of the continuous two-dimensional tomographic reconstruction problem, that will serve as the basis of describing the formulation of the reconstruction problem in the discrete environment.

In a common representation of two-dimensional transmission tomography there is an unknown $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ function we want to reconstruct (usually, because it represents the two-dimensional cross-section of a real-world object). The only data we can measure about this unknown function is a set of its line integrals given by the Radon-transform as

$$[Rf](\alpha, t) = \int_{-\infty}^{\infty} f(t \cos(\alpha) - q \sin(\alpha), t \sin(\alpha) + q \cos(\alpha)) dq, \quad (1)$$

where the α and t value, respectively, describes the direction and the position of a line in the two-dimensional space, with its points parameterized by q . In transmission tomography the task is to reconstruct an f' function that has the same projections as the original $f(u, v)$ function, in a set of predefined directions. Theoretically, this problem can be solved by exact mathematical methods when all possible $[Rf](\alpha, p)$ values are available [7, 12].

Unfortunately, in a practical application we usually can only deal with a finite number of values therefore we have to discretize the model applied for both the projection data and the function to be reconstructed. In the followings, we will assume that the function to be reconstructed has constant values on each unit square shaped region determined by the two-dimensional integer lattice, that is

$$f(u + a, v + b) = f(u + c, v + d); \quad u, v \in \mathbb{Z}; \quad a, b, c, d \in [0, 1). \quad (2)$$

We will further assume that the function f has a bounded support, therefore without the loss of generality we can say that in (2) $u, v \in \{[-\frac{n}{2}, \frac{n}{2}] \cap \mathbb{Z}\}$ with a constant n value. This restriction does not affect the applicability of the model, since in a real-world application we do not have infinitely large objects to deal with. This way, the task can be regarded as the reconstruction of an $n \times n$ pixel-sized image, from its projections.

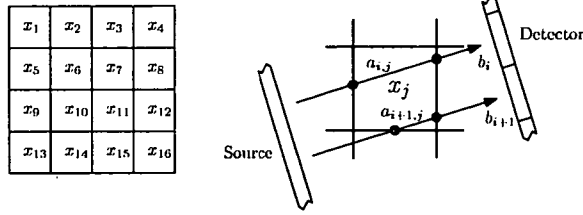


Figure 1: Representation of the ordering of the pixels and the parallel-beam geometry used.

In our experiments a projection was defined as a set of $[Rf](\alpha, t)$ values with the same α angles and

$$t \in \left\{ k + 0.5 \mid k \in \mathbb{N}, |k + 0.5| \leq n/\sqrt{2} \right\} \quad (3)$$

values, assuming that the origin of the coordinate system was placed into the center of the image to be reconstructed. In the defined projection geometry, a projection is composed of integrals taken along a set of equidistantly placed parallel lines. The distance between the neighboring projection lines was set to be 1 unit in the coordinate system and we used as many projection lines as needed to cover the whole image. The rotation center used for controlling the relation of the image and the projections was placed half-way between two projection lines, and in the center of the image to be reconstructed.

Using the previous restrictions the reconstruction problem can be represented by a system of equations

$$\mathbf{Ax} = \mathbf{b}; \quad \mathbf{A} = (a_{i,j})_{m \times n^2} \in \mathbb{R}^{m \times n^2}, \quad \mathbf{x} \in \mathbb{R}^{n^2}, \quad \mathbf{b} \in \mathbb{R}^m, \quad (4)$$

where \mathbf{x} denotes the ordered sequence of the pixels of the unknown image to be reconstructed, \mathbf{b} is the sequence of the measured projection values and \mathbf{A} describes the connection between \mathbf{x} and \mathbf{b} , where all $a_{i,j}$ elements give the length of the line segment of the i -th projection line through the j -th pixel. An illustration can be seen in Figure 1. Ideally, by solving equation system (4) we can get the pixel values of an image that has exactly the same projections as the measured ones.

Although, there are several general methods for solving equation systems, a direct method for finding the solutions of (4) is usually not the best approach. The resulting equation system can be too large for exact equation system solvers and might also be underdetermined, possibly yielding an infinite number of solutions. On the other hand, we cannot even guarantee that there is a solution, since – due to measurement errors and noise – the equation system can be inconsistent as well.

The different types of the algebraic reconstruction techniques [2, 7, 12] try to overcome this problem by applying iterative algorithms to approximate a solution, from a suitably chosen initial suggestion.

Another group of reconstruction methods reformulate the task as an energy minimization problem with a given

$$C(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda \cdot g(\mathbf{x}) \quad (5)$$

energy function. In the above formulation \mathbf{A} , \mathbf{b} , and \mathbf{x} are the same as defined in (4) and $g(\mathbf{x})$ is a function representing additional information about the image to be reconstructed with a given λ weight. In the ideal case, we can find a vector \mathbf{x}^* , where (5) takes its minimal value, with an appropriate general optimizer [1, 14]. This \mathbf{x}^* vector will represent the desired reconstruction. The main advantage of this approach is that it can easily incorporate some a priori information into the model via a suitable $g(\mathbf{x})$ function.

3 Binary Reconstruction Algorithms

We examined the projection selection dependency in binary tomography, i.e., in the case when the results can contain only binary values (and $\mathbf{x} \in \{0, 1\}^{n^2}$ in (4)). In our experiments we applied three suitable binary tomographic reconstruction algorithms from the corresponding literature. All three algorithms are deterministic, therefore their results are unique for each input and can easily be evaluated. The brief introduction of the algorithms can be given as follows.

3.1 Thresholded Simultaneous Iterative Reconstruction Technique (TSIRT)

The first algorithm was basically a continuous reconstruction followed by a thresholding. The continuous reconstruction was produced by the Simultaneous Iterative Reconstruction Technique (SIRT) [7, 12], which is an iterative algorithm for finding an approximate solution of (4). After obtaining the continuous result we applied a thresholding of the pixel values with a 0.5 threshold.

3.2 Discrete Algebraic Reconstruction Technique (DART)

The second algorithm was the Discrete Algebraic Reconstruction Technique [2], which is a combination of an algebraic based reconstruction method and a iterated thresholding. The DART starts by producing a continuous reconstruction using a suitable algorithm, and applies a thresholding of the result. Later, the values of the boundary pixels are fine-tuned by an iterative process.

In our experiments, we applied 10 iterations of the SIRT algorithm to obtain the continuous reconstructions, and used a threshold value of 0.5.

3.3 Energy Minimization Tomography with DC Programming

The third algorithm – that was first introduced in [16] – is based on DC programming (a numerical method for minimizing the difference of convex functions), and

performs the reconstruction by minimizing an energy function given as

$$\mathcal{J}_\lambda(\mathbf{x}) := \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \frac{\gamma}{2} \sum_{j=1}^{n^2} \sum_{l \in N_4(j)} (\mathbf{x}_j - \mathbf{x}_l)^2 - \lambda \frac{1}{2} \langle \mathbf{x}, \mathbf{x} - \mathbf{e} \rangle, \quad \mathbf{x} \in [0, 1]^{n^2}. \quad (6)$$

Here, γ is a fixed constant to control the weight of the smoothness term on the result, $N_4(j)$ is the set of pixels 4-connected to the j -th pixel, and \mathbf{e} denotes the vector with all n^2 coordinates equal to 1. This algorithm starts with approximating an optimal continuous result by minimizing the energy function with a $\lambda = 0$ value. After, an iterated process forces binary results, by proceeding with the minimization while periodically increasing λ with a λ_Δ value.

In the sequel, we will simply call this algorithm "DC". In the reconstructions the parameter settings of the DC algorithm were determined as in [17].

4 Test Frameset

We conducted experimental tests on a set of software phantoms. We reconstructed them from different sets of their projections and we evaluated the results from two different viewpoints. First, we compared the reconstructions of the same phantoms, under the same conditions but from different sets of their projections. With the result of these experiments, we could determine how dependent the reconstruction of a specific object – performed by a specific algorithm – can be on the choice of projection angles. Secondly, we compared the reconstruction of the same objects, performed by using the same projection directions, but under different conditions (original and altered versions of the objects, addition of different levels of noise, or in case when the reconstruction is performed by different algorithms), in order to see if there are regularities of the projection selection dependency of an object which can be possibly used in practical applications.

Our test database consisted of 10 phantom images, all having the same size of 128×128 pixels. The database could be divided into two parts, 5 basic phantoms with different properties (those can be seen in Figure 2) and a slightly altered version of each basic phantom (the ones shown in Figure 3), which simulate small distortions of the object of study (i.e., fractures or bubbles).

The elements of the test database were chosen based on previous studies [2, 14, 16, 17, 18, 19]. Phantoms like the ones of Figure 2a and Figure 2b, containing circles in a ring, are commonly used test images in discrete tomography, since they are easy to generate but relatively hard to reconstruct. The ones in Figure 2c and Figure 2d are phantoms similar to complex real world objects. Finally, Figure 2e shows a highly direction dependent image, that is useful for illustrating our results.

For simulating measurement errors during the projection acquisition phase of a real-world application, some of the experiments were performed using projection data corrupted by noise. In a real application the gathered projection data can be degraded by different artifacts – e.g.: beam hardening, photon scattering, imperfections of the detectors, background noise, etc. – most of which can be handled

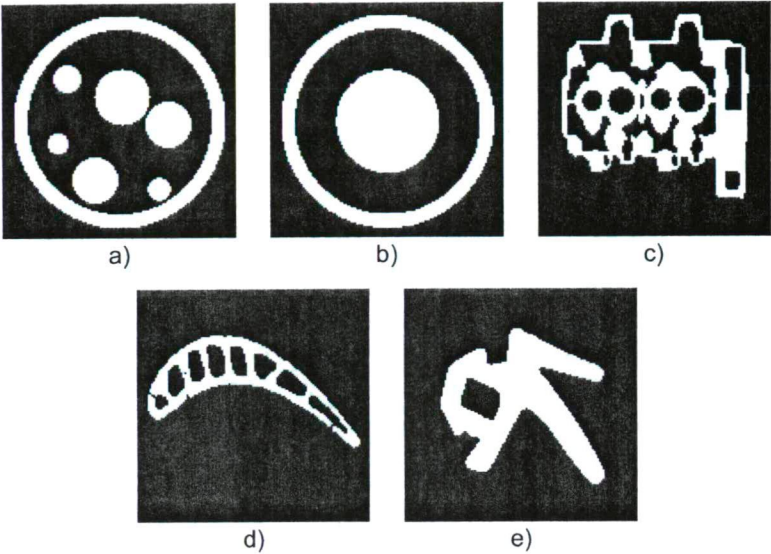


Figure 2: Basic images in the test database.

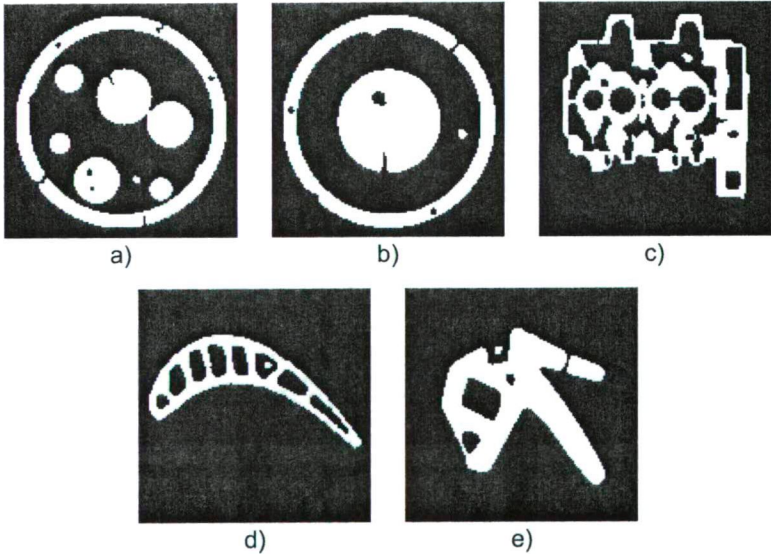


Figure 3: Altered images in the test database.

by preprocessing steps [3, 5, 10, 13]. Unfortunately, it was impossible to simulate all these effects in our experiments. Therefore, we decided to apply an additive Gaussian noise, which is a common technique for modeling noise in transmission tomography [4, 5, 13, 20].

In each case the mean value of the noise was set to $\mu = 0$, and the standard deviation was chosen from the set $\sigma \in \{0.5; 1.5; 5.0\}$. Taken, that the mean of the projection values was approximately 40, we can say that the amount of noise in the projection was respectively 0%, 1.25%, 3.75% and 12.5%, compared to the amount of useful data. Together with the noiseless case, this gave four different versions of the projection data. We generated and stored the noise in advance, before the reconstructions, in order to ensure the same conditions in all experiments.

The reconstruction algorithms were implemented with GPU acceleration using the NVIDIA CUDA programming toolkit [21]. For the computation we used a 2.66 GHz Core 2 Quad CPU, and an NVIDIA GeForce GTS250 GPU. With this highly parallel implementation the time required to perform all 1145224 reconstruction tasks was about 500 hours.

For the evaluation of the results we used a numerical error measurement called Relative Mean Error (*RME*) that was defined in [11]. The *RME* value of a reconstruction is computed as

$$RME(\mathbf{x}^*, \mathbf{y}) = \frac{\sum_{i=1}^n |x_i^* - y_i|}{\sum_{i=1}^n x_i^*} . \quad (7)$$

Here \mathbf{x}^* denotes the vector of pixel values of an expected reconstruction (in our case the pixel values of the phantom processed) and \mathbf{y} is the reconstruction provided by one of the reconstruction algorithms described in Section 3, performed under a specific set of conditions (we will refer to the phantom and its reconstruction given by \mathbf{x}^* and \mathbf{y} , in the text). Informally, the *RME* value gives the ratio of missed pixels in a binary reconstruction, normalized by the number of object pixels on the expected image. The main advantage of the *RME* measurement is that it gives the amount of error compared to the size of the object of study, and not the image itself (thus the error measurement is not affected by the zero padding of the image to be reconstructed). As a consequence the *RME* measurement can take values higher than 1, but fortunately this does not affect our evaluations, since we are interested in comparing the reconstructions to each other (and not determining their quality).

We generated the projection sets used in our experiments with two angle set selection techniques described in [18]. In the followings we shortly describe them.

4.1 Equiangular Angle Sets

The first type of angle sets were generated equiangularly along the half circle. Such projection angle sets are uniquely determined by their number of p projections and a starting angle α as

$$S(\alpha, p) = \left\{ \alpha + i \frac{180^\circ}{p} \mid i = 0, \dots, p-1 \right\} . \quad (8)$$

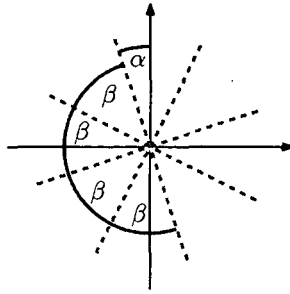


Figure 4: Example of the equiangular projection angle sets (angle set $S(\alpha, 4)$).

An example of the equiangular angle sets is given in Figure 4. With our notation 0° stands for vertical projection beams, aimed from the bottom to the top of the image.

With each image, reconstruction algorithm and specified noise, we performed reconstructions from projection sets $S(\alpha, p)$, with $p \in \{2, \dots, 18\}$ projection numbers and all integer α starting angles ranging between 0° to $\lfloor \frac{180^\circ}{p} \rfloor$.

4.2 Angle Set Selection with Greedy Angle Testing

We also conducted experiments on several non-equangular projection sets. Unfortunately, the extremely high number of such sets (even assuming only integer angles between 0° and 179° with 2 projections would produce $\binom{180}{2} = 16110$ possible angle sets) made it impossible to perform such a thorough testing that we did in the equiangular case. Therefore, we had to choose a smaller subset of all the possibilities.

In accordance to practical applications we restricted our studies to projection sets producing highly accurate results. We must note that the performance of projection sets strongly depends on the image to be reconstructed, and other conditions – like the applied reconstruction algorithm or noise – can also have influence on the information content of the projections. Therefore, we generated non-equangular angle sets for each image, algorithm and noise level, trying to find highly accurate projections sets.

For the purpose of generating the desired angle sets we used the greedy angle selection algorithm of [18], that produces an $L = \langle \alpha_1, \alpha_2, \dots, \alpha_p \rangle$ ordered list of angles as follows.

Greedy: Greedy angle selection algorithm.

Input: \mathbf{x}^* vector of image pixel values, $p \geq 2$ maximal number of angles, and $1 \leq \alpha_1 \leq 179$ predetermined integer angle.

Output: $L = \langle \alpha_1, \alpha_2, \dots, \alpha_l \rangle$ angle list so that $l \leq p$.

Step 1 Set $L_1 = \langle \alpha_1 \rangle$, $i = 1$;

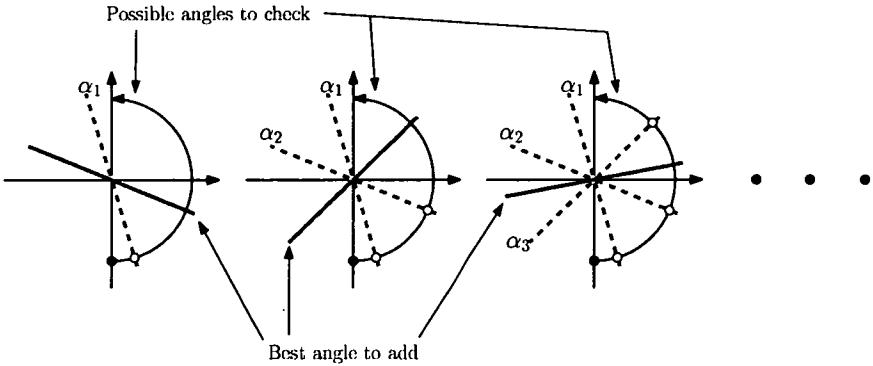


Figure 5: Steps of the Greedy algorithm.

Step 2 Let $i \leftarrow i + 1$;

Step 3 Let $0^\circ \leq \alpha^* \leq 179^\circ$ be an integer angle for which $RME(\mathbf{x}^*, \mathbf{x}_{(L_{i-1}, \alpha^*)})$ is minimal;

Step 4 Let the next list of angles be $L_i = \langle L_{i-1}, \alpha^* \rangle$;

Step 5 If $i = p$ or $RME(\mathbf{x}^*, \mathbf{x}_{L_i}) = 0$ return with L_i otherwise go to **Step 2**

Here, \mathbf{x}_L stands for the reconstruction of the \mathbf{x}^* image from the projection set specified by the angles of L .

The **Greedy** algorithm is an iterative process that takes an image, and in each iteration it tries to determine the best projection to be added to the current ones. The result of this algorithm is a list of angles ordered by decreasing significance. A demonstration of the algorithm's proceeding is plotted in Figure 5.

In each iteration, when there are i projections already chosen, the algorithm has to test $180 - i$ possible projection sets to find the best one to proceed with. This determines a total number of $\sum_{i=1}^{p-1} (180 - i)$ projection angle sets, assuming that the maximal number of projections is p . We used these angle sets for building up our test frameset. This provided reconstructions from a significant number of non-equiangular projection sets, that gave a proper base for comparing the results of the different reconstruction algorithms.

Although, in [18] other methods for non-equiangular angle set selection are also described – which could have been used for generating the desired angle sets in our studies – we decided to use the **Greedy** algorithm, due to its deterministic nature and relatively fast performance.

The parameters of the **Greedy** algorithm were set as described in [18]: we allowed to choose integer angles between 0° and 179° , and used the SIRT algorithm for producing the first α_1 projection angle. Furthermore, we set the maximal number of projections to 18.

5 Experimental Results

After performing the tests we started comparing the *RME* values of the reconstructions. As it was mentioned in Section 4 we evaluated the data from two viewpoints. First, we compared the reconstructions of the same objects, performed under the same conditions (i.e., using the same reconstruction algorithms and same noise), but from different projection sets to see how much improvement can be reached in the reconstruction, solely by finding better projections. Secondly, we compared the reconstructions of the same objects, from the same projections, but under different conditions to find out whether or not the direction dependency of the objects remain consistent under different conditions – i.e., to see if there are projection angle sets which lead to better (or worse) reconstruction results under all circumstances.

Although all the results proved our observations, due to the extremely high number of performed reconstructions we cannot present all of them in detail. Therefore, in this section we will discuss our general observations, and show only samples of the result dataset which demonstrate them the best.

5.1 Reconstruction from Equiangular Projection Sets

In the case of the equiangular angle sets the task was relatively easy. We could plot the curves of *RME* values belonging to the reconstructions performed by one of the three reconstruction algorithms, from a specific number of projections, and using a given type of noise, according to the starting angle (as it was done in [19]). Then, we could easily determine if the curves have significant differences between their minimal and maximal values (i.e., if the accuracy of the reconstructions depend on the choice of the projection angles), and if the curves on the diagrams are similar or not (i.e., if there is correspondence between the reconstructions of the same object from the same projections, but under different conditions). As an example, the *RME* values belonging to the reconstructions of the phantoms in Figure 2e and Figure 3e (the basic and altered versions of the same object) can be seen in Figure 6. The plotted graphs show statistics of all the reconstructions performed from equiangular projection sets containing 4 projections, grouped by the different types of applied random noise.

Apparently, all the graphs have significant gaps between their minimum and maximum points, showing that the binary reconstruction of an object can indeed be improved only by taking the proper projections (Figures 7 and 8 give examples of the reconstructed images preformed with different projection sets). We can also notice, that the curves of the graphs are relatively smooth, suggesting that projections with angles close to each other have similar information content. This also indicates that small changes of the projection angles may only have negligible effects on the result of the reconstruction. In addition, all the curves show some degree of similarity, i.e., the minimal and maximal values correspond to similar projection angle sets and the transitions between the extrema are also alike.

Such examination of equiangular projections has already been done in [17, 18], but we decided to reproduce these results to highlight the regularities and give a

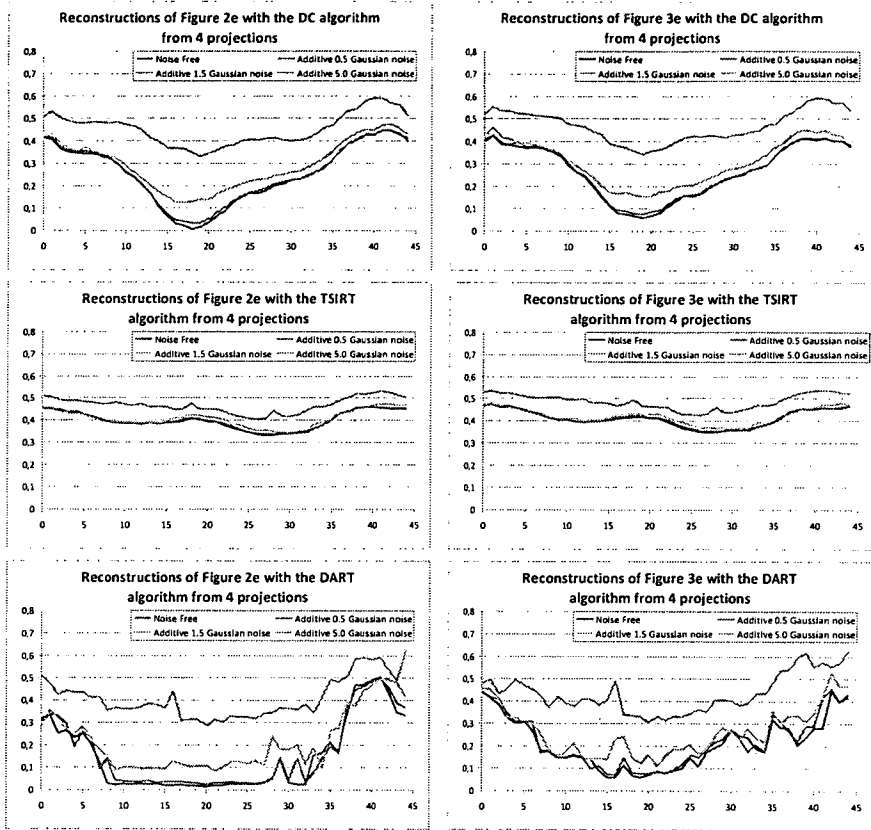


Figure 6: RME values of the reconstructions of the phantoms in Figure 2e and Figure 3e from 4 projections according to the starting angle. The three diagrams on the left hand side are the reconstructions of Figure 2e, and those on the right hand side are the reconstructions of Figure 3e. Each row shows the results of one applied reconstruction algorithm (from the top to the bottom: DC, TSIRT and DART algorithms, respectively). Each diagram shows four curves each belonging to the reconstructions affected by the four different types of noise. On the diagrams: horizontal axis stands for the starting angle, and vertical axis for the RME value.

brief explanation of the projection selection dependency of objects.

Let us consider the problem of angle selection dependency in general, i.e., without the assumption of examining any specific object, reconstruction algorithm, or angle selection strategy. As it was described in Section 2, the goal of transmission tomography is to reconstruct a function f' , that has the prescribed projections in a set of directions. We also mentioned that the limited amount of projection values may not contain enough information for a proper reconstruction. This lack of data makes it possible to have several different f' functions having the same projections, i.e., different reconstructions. Nevertheless, we are usually interested in only one specific result (the one identical to the original object), and any other solution is considered to be incorrect.

In our discretized grid based representation, the lack of information means that there can be several different binary images having the expected projections, some of which can be entirely different from the expected reconstruction. In this case the most we can expect from a reconstruction algorithm is to find one of these possible solutions. If the set of possible reconstructions is large, the probability of finding an accurate solution can be small. Naturally, if we want to increase the probability of finding an acceptable reconstruction, we have to reduce the set of possibilities, by gaining additional information for the reconstruction. This can be done by either adding some extra prior knowledge to the model (e.g. by assuming that the shape of the object of study fulfils some special property [9]), acquiring additional projections, or taking better projections with higher information content.

Since we have been experimenting on deterministic algorithms, the choice of the reconstruction method can also be regarded as a prior knowledge, i.e., to decide which strategy should be used for choosing a reconstruction out of the possible ones.

Regarding the previous discussion, the differences in the curves of Figure 6 can easily be explained. First, let us consider the effects of the additive noise. In this case changing the projection data also affects the feasible solutions the algorithms can choose from, and all the possible solutions will have some degree of error. Since we applied the same noise each time, the distortion of data is similar with every reconstruction, and the effect in Figure 6 is an approximately constant upwards shifting in the *RME* value curves. Naturally, higher noise levels result in bigger upwards shifting of the curves.

The effect is different when we compare the *RME* curves on Figure 6, belonging to the basic and altered version of the phantoms. We know, that some objects can be reconstructed easier if their projections are taken from a specific set of directions. It is also clear, that different objects can have different optimal angle sets. Therefore, we can only find the optimal set of directions for a specific object. If we alter the structure of the the object, the information content of its projections taken from specific directions can change, thus some of them can be more (or less) useful for the reconstruction. This means that the modified object can have different optimal angle sets. Our results show that this change of the optimal directions correspond to the level of modification of the object. Small modifications do not notably affect the result of reconstruction, but by increasing the distortion we reach

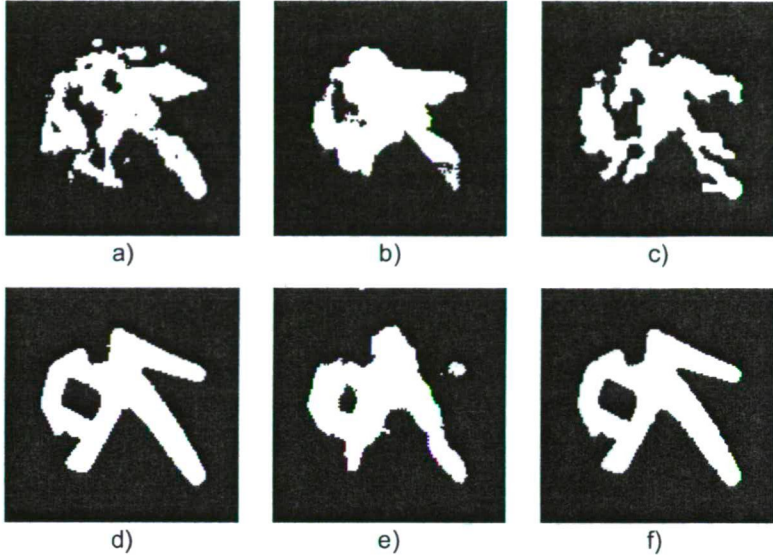


Figure 7: Reconstructions of the phantom in Figure 2e performed by the three reconstruction algorithms, from different projection sets containing 4 projections. Images a, b, c correspond to the worst, and d, e, f to the best reconstructions gained by using equiangular projection sets. The three columns contain results belonging to the different reconstruction algorithms (DC: a, d; DART: b, e; TSIRT: c, f).

a certain point, where we are dealing with an entirely different object, with different properties.

The differences in the curves of Figure 6 are the most remarkable in the case of comparing the results belonging to the different reconstruction algorithms. Since we used the same information in all the reconstructions (the projection data and the fact that we are looking for binary solutions), the choice of the reconstruction algorithm should only influence which one of the feasible results is found. This previous assumption is, however, not true, since the algorithms we used are not guaranteed to give optimal solutions. Still, having more informative projection sets can make it more likely to get better results from the same projection sets regardless of the applied reconstruction algorithm.

We should also take a look at what this projection angle dependency means regarding the reconstructed images themselves. In Figure 7 we gave the best and worst reconstructions of the phantom in Figure 2e reconstructed by the three examined algorithms from equiangular projection sets containing 4 projections. Here, we can see significant differences, especially on the reconstructions performed by the DC and the DART algorithms, but even the results of the TSIRT algorithm can notably be improved by finding the proper projections.

5.2 Reconstruction from Non-Equiangular Projection Sets

It was also shown [18] that – in the case of noiseless projections – the dissimilarity between the result of the reconstructions using different projection sets can be even more significant if we allow the acquisition through non-equiangular projection sets. Here, we extended the previous work by examining the case when the projection data is affected by random noise. A brief example this can be seen in Figure 8, showing the reconstructions of the phantom of Figure 3e, from different projection sets. Our results indicate that the previous observations still hold, even in the case of distorted data, and loosing the assumption of equiangularity can bring further improvement to the reconstruction.

Comparing the results numerically is a bit harder when using non-equiangular angle sets, than with the equiangular ones. In this case we could only compare a set of reconstruction pairs by plotting both *RME* values of the corresponding reconstructions. We used this technique for a pairwise comparison of the different reconstruction algorithms. Examples for the resulting diagrams can be seen in Figure 9.

We would expect Figure 9 to show some degree of correspondence between the reconstructions of the different algorithms. If it is so, then the projection sets producing better results for one algorithm should also produce high quality reconstructions for other ones, and the points in the diagrams should be placed along – or at least close to – a diagonal straight line. As we can see, the points of Figure 9 do satisfy our expectations so we can deduce that there is a strong correspondence between the results of the different reconstruction algorithms.

We used the same technique for comparing the reconstructions of the basic and altered versions of the images from projection sets using the same angles, in the case when we allow non-equiangular angles and the projection data can be corrupted by noise. An example of such results is given in Figure 10 by giving the *RME* value pairs of the reconstructions of Figure 2e and Figure 3e, performed with the DC reconstruction algorithm. Again, we can see that the points highlighted in Figure 10 are allocated close to a diagonal straight line indicating that small distortions of the object of study do not have a significant influence on the reconstruction. Moreover, we also can deduce that such distortions do not affect the information content of projections taken from specific angles, i.e., if a projection of an object holds high information content, then another projection taken from the same direction, but from a slightly different object would also provide similarly useful information for the reconstruction.

The results indicate that there is a strong correspondence between the binary reconstructions of an object performed with different reconstruction algorithms, even in the case when the projections are chosen non-equiangularly, and the projection data is affected by different types of random noise. Furthermore, we can say that the projection angle dependency of objects remains consistent and somewhat predictable under different conditions, and therefore it is possible to use the direction-dependency of objects in practical applications for improving the results of reconstruction (like it was proposed in [17, 19]).

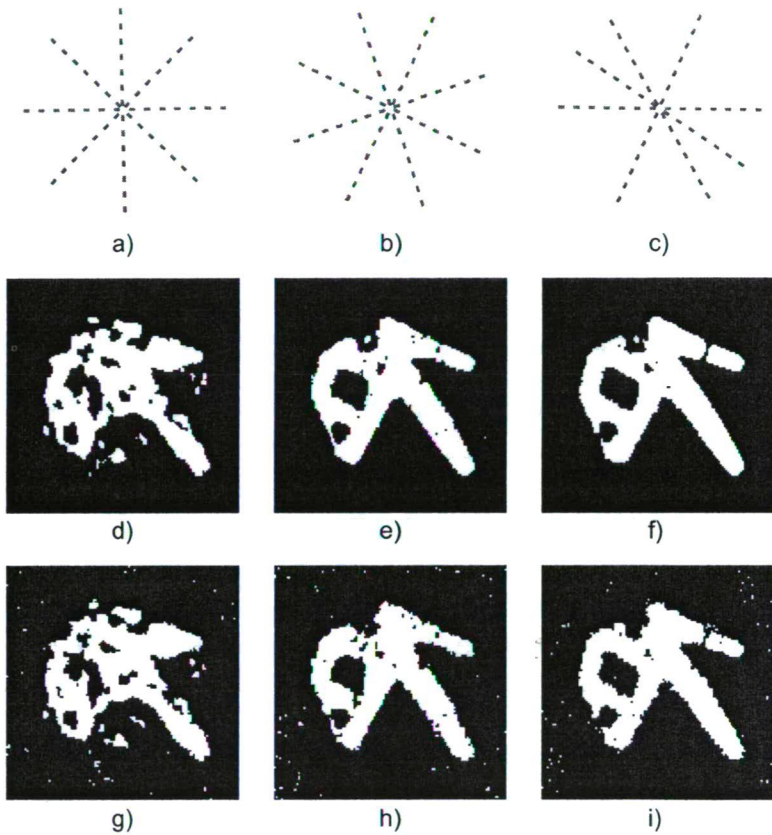


Figure 8: Comparing of the reconstructions of the phantom in Figure 3e, reconstructed by the DC algorithm, from different projection sets, with different levels of noise. Each column contains the angles of the projection sets and the corresponding reconstructions. Top row indicates the angles in the projection sets, second row gives the reconstruction in the noiseless case, bottom row contains the reconstructions from projection sets affected by random noise with 1.5 deviance.

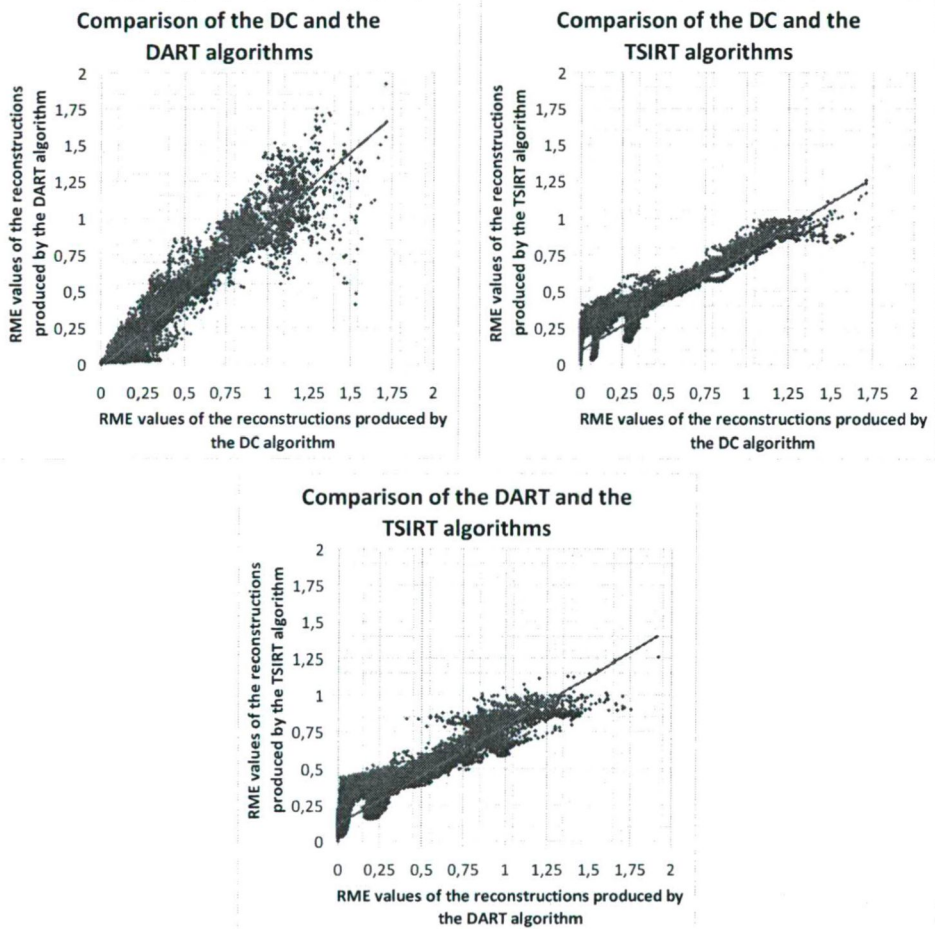


Figure 9: Cross comparison of the algorithms of Section 3 for the reconstructions of the phantom in Figure 2e. Each diagram contains points representing RME value pairs corresponding to reconstruction performed by two algorithms. The graphs contain results with projection numbers from 2 to 18, and all the additive random noise described in Section 4. The red lines are regression lines fitted to the points for better visibility.

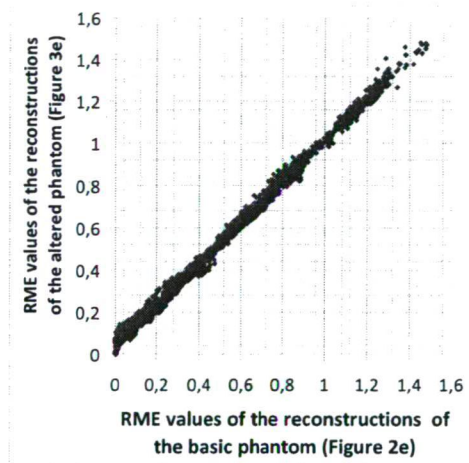


Figure 10: Reconstructions of Figure 2e and Figure 3e performed with the DC reconstruction algorithm from projection sets acquired with different angles. Each point of the diagram is positioned according to the *RME* values of the reconstructions of the two images from the projection sets having the same angles. Different points of the diagram give reconstructions corresponding to different angle sets. The diagram also contain reconstructions performed with noise-corrupted projections.

Although, we have only given a sample of the performed reconstructions, we must note that our observations seemed to hold in all our many test cases.

6 Conclusion

In this paper we studied the projection angle selection dependency of reconstructions in the field of binary tomography. We have summarized and extended previous results by showing, that there is a strict correspondence between the reconstructions performed by different binary tomography reconstruction algorithms even in the case when the projection angles are selected from arbitrary directions, the objects of study are distorted and the projection data is affected by random noise.

Our results indicate that this angle selection dependency is caused by the different information content of the different projections, which is the intrinsic property of the images themselves. In the future we intend to discover the deep connections between our experimental results and the theory of discrete tomography, and also to extend our investigations to the case of three dimensional tomography, i.e, when the objects to be reconstructed are represented in three dimensions, and we can take projections from any directions on the sphere.

Acknowledgments

We would like to thank Joost Batenburg for providing test images (Figures 2c-d and Figures 3c-d) for the studies. The phantom of Figure 2e and Figure 3e were originated from the image database of the IAPR Technical Committee on Discrete Geometry (TC18). We are also grateful to the reviewers of the manuscript for their valuable comments.

References

- [1] P. Balázs, M. Gara, *An evolutionary approach for object-based image reconstruction using learnt priors*, Lecture Notes in Computer Science 5575 (2009) 520–529.
- [2] K.J. Batenburg, J. Sijbers, *DART: a fast heuristic algebraic reconstruction algorithm for discrete tomography*, IEEE Conference on Image Processing IV (2007) 133–136.
- [3] J. Baumann, Z. Kiss, S. Krimmel, A. Kuba, A. Nagy, L. Rodek, B. Schillinger, J. Stephan, *Discrete Tomography Methods for Nondestructive Testing*, Chapter 14 of [9] (2007) 303–331.
- [4] B. Chalmond, F. Coldefy, B. Lavayssière, *Tomographic reconstruction from non-calibrated noisy projections in non-destructive evaluation*, Inverse Problems 15 (1999) 399–411.

- [5] P. Duvauchelle, N. Freud, V. Kaftandjian, D. Babot, *A computer code to simulate X-ray imaging techniques*, Nuclear Instruments and Methods in Physics Research B 170 (2000) 245–258.
- [6] R.J. Gardner, P. Gritzmann, *Discrete tomography: determination of finite sets by X-rays*, Transactions of the American Mathematical Society, 349(6) (1997) 2271–2295.
- [7] G.T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections, 2nd Edition*, Springer (2009).
- [8] G.T. Herman, A. Kuba (Eds.), *Discrete Tomography: Foundations, Algorithms and Applications*, Birkhäuser, Boston (1999).
- [9] G.T. Herman, A. Kuba (Eds.), *Advances in Discrete Tomography and Its Applications*, Birkhäuser, Boston (2007).
- [10] S. Kimmel, J. Baumann, Z. Kiss, A. Kuba, A. Nagy, J. Stephan, *Discrete tomography for reconstruction from limited view angles in non-destructive testing*, Electronic Notes in Discrete Mathematics 20 (2005) 455–474.
- [11] A. Kuba, G.T. Herman, S. Matej, A. Todd-Pokropek: *Medical applications of discrete tomography*, Discrete Mathematical Problems with Medical Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 55 (2000) 195–208.
- [12] A. C. Kak, M. Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York (1999).
- [13] N.D.A. Mascarenhas, C.A.N. Santos, P.E. Cruvinel, *Transmission tomography under Poisson noise using the Anscombe transformation and Wiener filtering of the projections*, Nuclear Instruments and Methods in Physics Research A 423 (1999) 265–271.
- [14] A. Nagy, A. Kuba, *Reconstruction of binary matrices from fan-beam projections*, Acta Cybernetica, 17(2) (2005) 359–385.
- [15] G. Placidi, M. Alecci, A. Sotgiu, *Theory of adaptive acquisition method for image reconstruction from projections and application to EPR imaging*, Journal of Magnetic Resonance, Series B, (1995) 50–57.
- [16] T. Schüle, C. Schnörr, S. Weber, J. Hornegger, *Discrete tomography by convex-concave regularization and D.C. programming*, Discrete Applied Mathematics 151 (2005) 229–243.
- [17] L. Varga, P. Balázs, A. Nagy, *Direction-dependency of a binary tomographic reconstruction algorithm*, Lecture Notes in Computer Science 6026 (2010) 242–253.

- [18] L. Varga, P. Balázs, A. Nagy, *Projection selection algorithms for discrete tomography*, Lecture Notes in Computer Science 6474 (2010) 390–401.
- [19] L. Varga, P. Balázs, A. Nagy, *Direction-dependency of binary tomographic reconstruction algorithms*, Submitted to Graphical Models (CompIMAGE 2010 special issue).
- [20] S. Weber, A. Nagy, T. Schüle, C. Schnörr, A. Kuba, *A benchmark evaluation of large-scale optimization approaches to binary tomography*, Lecture Notes in Computer Science 4245 (2006) 146–156.
- [21] NVIDIA CUDA Programming Guide, Version 2.0
http://developer.download.nvidia.com/compute/cuda/2.0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf

Parameter Estimation of Flow-Measurement in Digital Angiography*

Krisztian Veress[†] and Tibor Csendes[†]

Abstract

The purpose of angiographic procedures used in cardiovascular interventions is to classify the patient's potential of regeneration after strokes caused by dead blood cells in the main arteria. The flow of blood into heart's capillaries is measured using x-ray radiometry with contrastive fluids. One quick and reliable method for estimating this potential could save lives and would allow further treatments to be more accurately planned.

Our task was to fit a 5-parameter Gamma function to the intensity samples extracted from the x-ray angiograms. The estimation of this function's parameters is hard given that the raw data set is heavily polluted with several different types of noise.

Our complete solution has four main parts which have also been successfully verified and validated. First, we propose a solution for eliminating the noise by applying a specially designed moving window Gauss filter. Secondly, we have designed an algorithm for computing a good initial guess for the Levenberg-Marquardt optimizer in order to achieve the required precision. Third, an algorithm is proposed for selecting significant points on the smoothed data set with an interval-based classification method. Finally, we apply the LM algorithm to compute the solutions in a nonlinear least squares way.

We have also formulated an algorithm based on interval arithmetic which can be effectively used for comparing nonlinear least-squares fit results and assign goodness values based on their residuals. This method has been used for measuring improvements during the development.

We must emphasize that the proposed algorithms are distinct, they can be used in other applications together or separately since they are generally applicable, they do not depend on specialties of the presented application.

Keywords: numerical analysis, optimization, parameter estimation, gamma function

*This work was supported by the national grant TAMOP-4.2.1/B-09/1/KONV-2010-0005

[†]Institute of Informatics, University of Szeged E-mail: {verkri,csendes}@inf.u-szeged.hu

1 Introduction

The digital subtraction angiography [17] used in medical surgery is one kind of an image recording and processing method where panoramic x-ray images are taken while contrastive x-ray fluid [8] is injected into the patient's heart's main arteria. The x-ray fluid flow is similar to blood-flow [11], thus the amount of blood that can reach the critical region can be measured. The goal is to estimate the patient's survival chances who has recently survived a stroke, being dead blood cells removed by means of a surgery intervention. There is a high correlation between these and the regeneration ratio (perfusion parameter) of his/her cardiac muscle [1], [22].

X-ray panoramic images are recorded at a rate of 15 fps for 10 to 15 seconds yielding 150 to 225 frames as digital subtraction angiograms [17]. The surgerer in charge selects the critical cardiac muscle region as the Region of Interest (ROI) [21] on the first couple of frames. The intensity of the x-ray fluid can be computed on each frame [3] by calculating the average intensity of the interior pixels in that ROI. In this way, an aggregate intensity value for each frame brings on a time-series ($M(t)$) which is going to be our input. Two examples are shown in Figure 1.

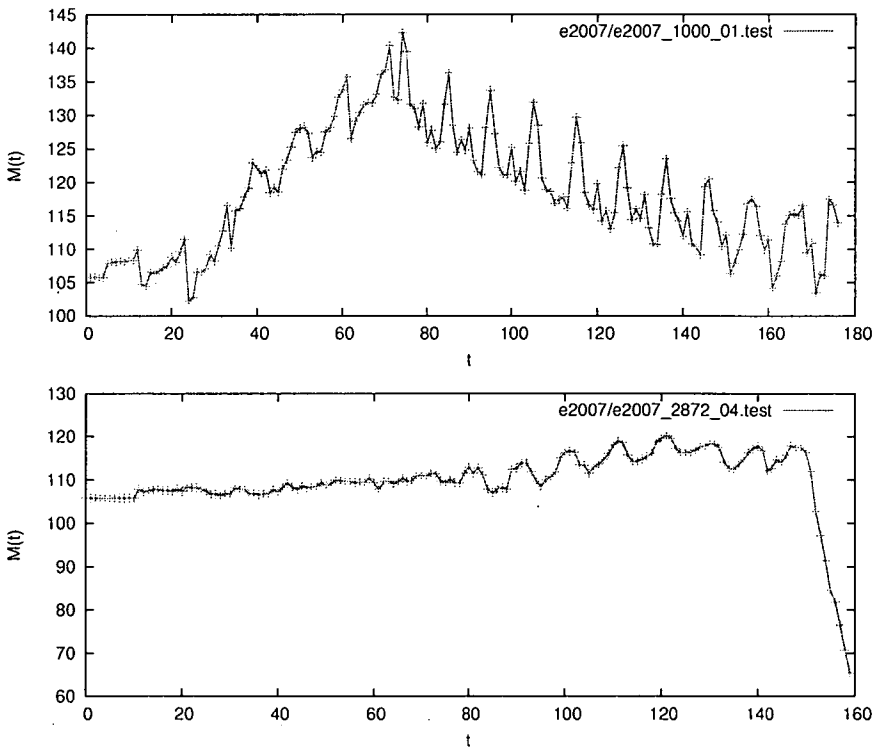


Figure 1: One valid and one erroneous input time-series of x-ray intensities

1.1 Modelling flow dynamics

Our main task was to characterize the x-ray fluid flow in blood vessels by means of numerical values. Several theoretical formulations have been proposed to explain the shape of peripheral indicator dilution curves [24], [25]. One expression proposed by Evans [5] and examined by Howard [13] has a graphical representation which bears a remarkable resemblance to indicator dilution curves without recirculation. This function can be expressed in the original form

$$H(t) = K_s(t - AT)^\alpha e^{-\frac{t - AT}{\beta}} + Z_l, \quad (1)$$

being $K_s > 0$ a constant scale factor, $AT > 0$ the appearance time, $Z_l > 0$ an offset, and $\alpha, \beta > 0$ the rising and descending slope shape parameters. To have a quick overview of this mathematical model, we have plots with different parameter sets in Figure 2.

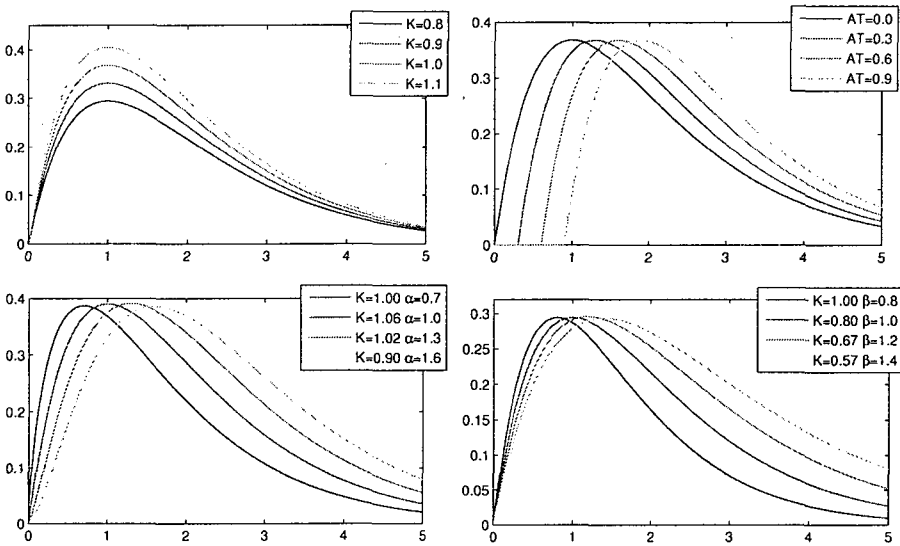


Figure 2: The effect of changing single parameters of $H(t)$

The AT parameter specifies the time when the contrastive fluid has been injected while Z_l and K_s are the base intensity and intensity scaling values of the x-ray device. The slope parameters describe the way blood can enter and exit the cardiac muscle. Since well-studied physiological meanings are to be abstracted from these numerical parameters [13], estimating them precisely and accurately is a must.

The main objective was to efficiently fit $H(t)$ to the initial samples $M(t)$ with high confidence regarding the nature of the consequences our results could introduce. Early studies showed that our inputs can be more precisely modeled with a

slightly modified version of $H(t)$:

$$G(t) = \left\{ \begin{array}{ll} Z_l & | \ AT < t \\ H(t) = K_s(t - AT)^\alpha e^{\frac{-(t - AT)}{\beta}} + Z_l & | \ AT \geq t \end{array} \right\}, \quad (2)$$

What we have now, is a formulation of a non-linear parameter estimation problem where the measured data is $M(t)$ and the model is $G(t)$. There are several methods for solving such problems [6],[14],[23] among which we have selected the Least Squares Estimation procedure. To be more precise, we are going to use the Levenberg-Marquardt algorithm [15], [19], minimizing the difference between the model and sample values in a nonlinear least squares way [7], [16]:

$$Z_{res} = \sum_{i=1}^n (G(t_i) - M(t_i))^2 \rightarrow \min. \quad (3)$$

1.2 Normalizing residual squares

After estimating the regression parameters, an essential aspect of the analysis is to test the appropriateness of the overall model. To declare a fit 'good' or 'bad', the sole sum of the residuals (Z_{res}) are not reasonably satisfactory, since a 'better' fit can have higher Z_{res} values than a 'worse' one caused by heavy noise or badly scaled sample.

Widely used techniques such as proportion of variance, chi-square and covariance matrix calculations showed nothing better. Since Z_{res} can be arbitrarily large, we propose a method for scaling these values into any chosen interval.

Let $\mathbf{f}, \mathbf{m} \in \mathbb{R}^n$, \mathbf{f} be the *vector* of our fitted values, \mathbf{m} the measurement *vector*, $g_l, m_l \in \mathbb{R}$ lower, and $g_u, m_u \in \mathbb{R}$ upper bound values. Then $\forall i \in 1, \dots, n$

$$\begin{aligned} \mathbf{m}_i &\in [m_l, m_u], & m_l &\leq m_u, & m_l, m_u &\in [0, 255], \\ \mathbf{g}_i &\in [g_l, g_u], & g_l &\leq g_u, & g_l, g_u &\in [0, 255], \end{aligned}$$

which results the following inclusion (based on interval arithmetic):

$$(\mathbf{g}_i - \mathbf{m}_i) \in [g_l, g_u] - [m_l, m_u] \in [g_l - m_u, g_u - m_l].$$

From interval arithmetic we know that for any $f : \mathbb{R}^n \mapsto \mathbb{R}$ function, its *natural interval extension* is given by the interval-valued function in the form

$$F : [\mathbb{R}^n] \mapsto [\mathbb{R}] \text{ where } F([x]) \supseteq \{f(y) \mid y \in [x]\}$$

and can be formulated by replacing each value with its thickest encasing interval:

$$x \in \mathbb{R} \mapsto [x, x] \in [\mathbb{R}].$$

Now we can easily compute the natural interval extension of Z_{res} :

$$\begin{aligned}
 Z_{res} &= \sum_{i=1}^n (G(t_i) - M(t_i))^2 = \sum_{i=1}^n (g_i - m_i)^2 \\
 &\in \sum_{i=1}^n ([g_l - m_u, g_u - m_l])^2 \in \sum_{i=1}^n ([0, \max((g_l - m_u)^2, (g_u - m_l)^2)]) \\
 &\in [0, n \max((g_l - m_u)^2, (g_u - m_l)^2)].
 \end{aligned}$$

which implies that

$$\hat{Z}_{res} = \frac{1}{n \max((g_l - m_u)^2, (g_u - m_l)^2)} Z_{res} \in [0, 1]. \quad (4)$$

The plotted original Z_{res} and normalized \hat{Z}_{res} values for different inputs are shown in Figure 3 (smaller value means better accuracy). Note that the normalized values are more scattered than the original ones meaning that normalization pushes away 'bad' and 'good' fits.

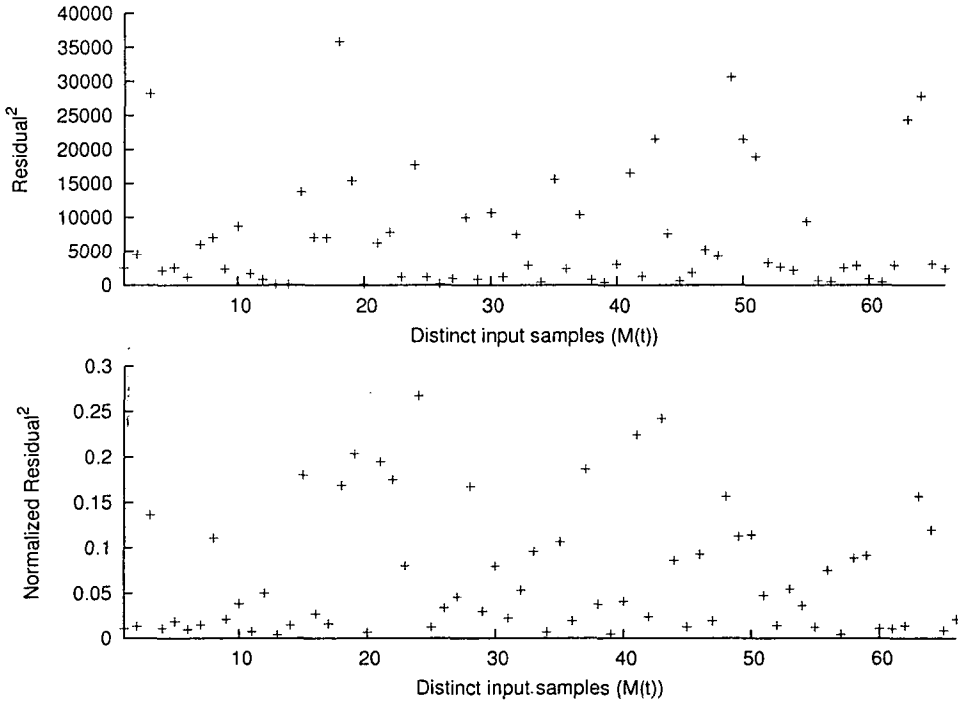


Figure 3: Original and normalized residuals on 66 different $M(t)$ samples

1.3 Results on the initial samples

To evaluate our solution we had 66 real life, anonymous medical samples at our disposal. We must emphasize that these samples contain noises from uncountable sources [10] (i.e. unprecise recording, unprecise fluid injection [8], x-ray device's auto-intensity regulation [9], image processing bugs [18]) and our effort in figuring out suitable noise models was a fool's errand.

We have implemented our solution in C++ by making the most of the Insight Toolkit [26]. On the first attempt we tried to directly fit the model (2) to our input using the mentioned LMA algorithm. Since the latter is an iterative curve-fitting method, we had to feed it with a suitable initial guess vector which was chosen for the following:

$$P_0 = (K_s, AT, \alpha, \beta, Z_l) = (0.02, 34.0, 3.3, 11, 1, 106.0)$$

Based on our statistic analyses P_0 describes a nearly-optimal estimator for a well-conducted, unnoisy, and error-free measurement sample. Since the results were really bad (concerning either performance or precision), we classified each fit in a graphical way into *appropriate* and *inappropriate fit* classes (see Figure 4), separately computed the normalized residuals and lastly matched these two properties.

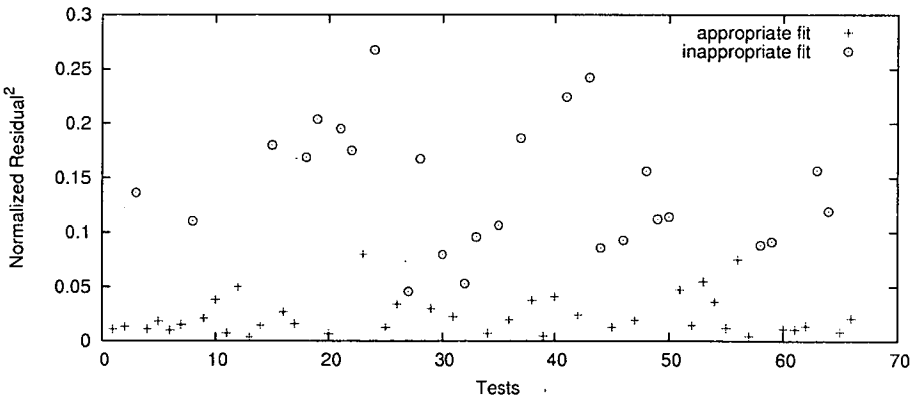


Figure 4: Normalized results on the test database (naive fit)

In Figure 4 a somewhat sharp interface showed up between appropriate and inappropriate classes from which we may conclude that there exist a high correlation between our normalized values and the optimality of the estimators.

However, we must recognize that not every inappropriately classified sample is a clearly wrong measurement which means that the LMA algorithm should be fine-tuned and other pre-, and post-processing phases should be included in our complete solution.

2 Noise filtering

In order to achieve better results, we have decided to apply a noise filtering algorithm whose primary goal was to eliminate spikes and produce a smoothed sample. Simple filters like median and arithmetic mean moving-window filters did not perform well on all types of measurements.

The chosen filtering algorithm is a general Gaussian moving window average type [12] filter with specially designed weights and variable length window size. The weights are designed to be precomputable given an initial sample, and not to introduce undesired offsets and scaling on the input values:

$$M^*(t_i) = \sum_{j=i-L_w}^{i+L_w} \frac{M(t_j)}{2L_w + 1} w_j \quad \text{if } L_w \leq t_i \leq |M(t)| - L_w,$$

where $\forall j \in [i - L_w, i + L_w]$, and the weights are:

$$w_j = e^{-(t_j - t_i)^2 / (2L_w + 1)} \frac{2L_w + 1}{\sum_{j=i-L_w}^{i+L_w} e^{-(t_j - t_i)^2 / (2L_w + 1)}}.$$

By selecting the weights in this way, it is guaranteed that $\forall i \in [1, n]$:

$$\sum_{j=i-L_w}^{i+L_w} w_j = \sum_{j=i-L_w}^{i+L_w} \left(e^{-(t_j - t_i)^2 / (2L_w + 1)} \frac{2L_w + 1}{\sum_{j=i-L_w}^{i+L_w} e^{-(t_j - t_i)^2 / (2L_w + 1)}} \right) = 2L_w + 1.$$

Generally speaking our proposed weighting method gives us a (not arithmetic) mean moving window filter with Gaussian weights. The optimal window size has been selected by generating a histogram for all possibly usable window sizes for all inputs in our database resulting the value 33.

This modification of the original smoothing filter has the same advantages as to the Savitzky-Golay [20] filter because it also tends to preserve features of the distribution such as relative maxima, minima and width, which are usually 'flattened' by other adjacent averaging techniques. However, further researches should be conducted to compare the performance, stability and applicability of these two filters and select the most appropriate one.

Our proposed filtering algorithm can successfully be used on any one-dimensional sample since weights depend only on the measurement vector and the optimal window size can also be found in the above way.

2.1 Smoothing results

In Figure 5 we present four different initial samples and the result of our proposed smoothing filter. One can see that our filter is not sensitive to radidly changing curves or spikes and keeps the dominant part of the input signal.

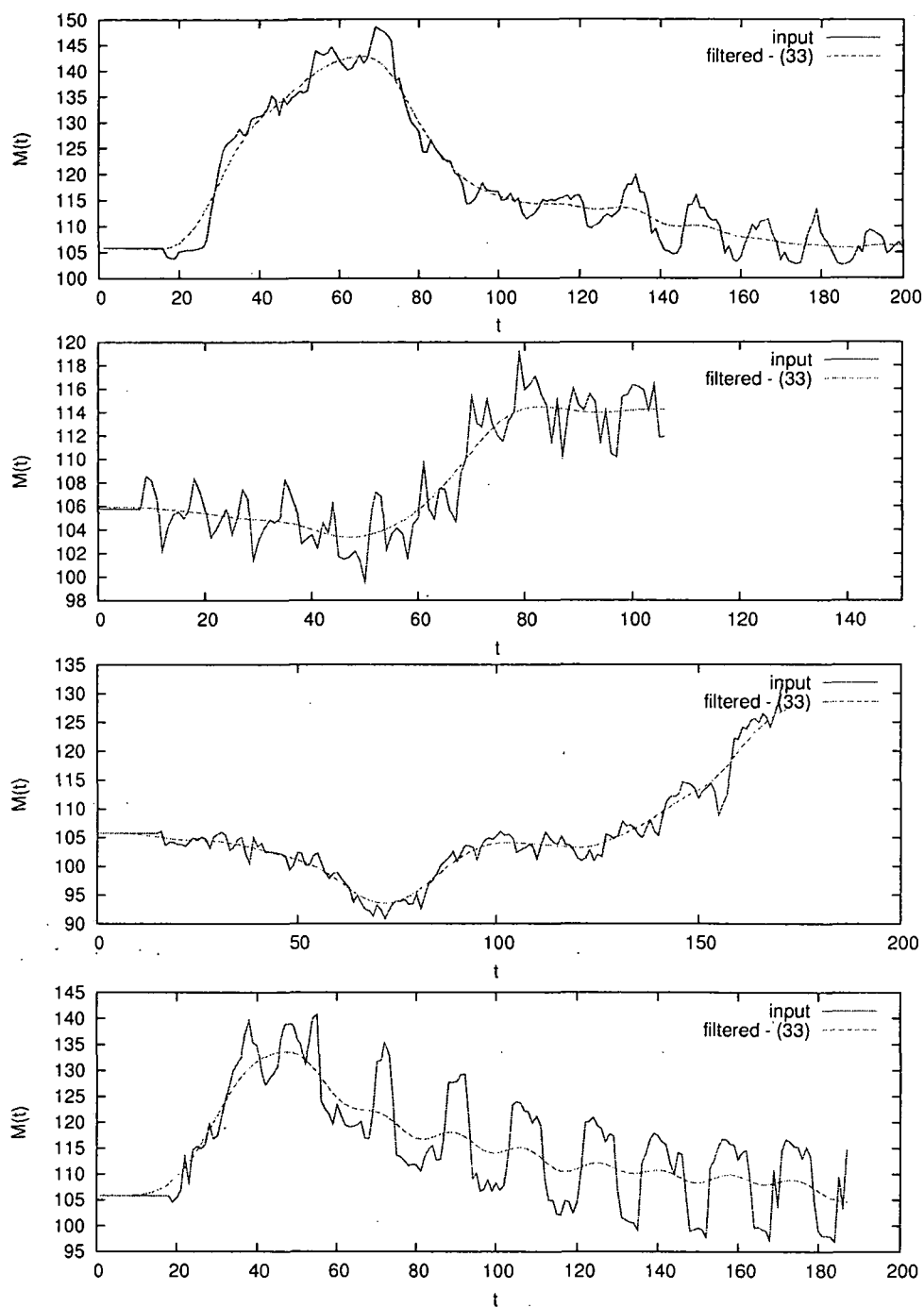


Figure 5: Smoothing filter results on real medical samples

2.2 Results on the filtered samples ($M^*(t)$)

We modified the NLLS LMA minimizer's objective function as to minimize the difference between the model ($G(t)$ and the **filtered sample** ($M^*(t)$). Table 1 shows aggregate statistics about the results generated on the 66-element test database.

Table 1: Numerical results of the LMA algorithm on $M(t)$ and $M^*(t)$ samples

$M(t)$	min	max	mean	median
iterations:	3	9999	3796.09	46
Z_{res} :	121.84	35815.86	7010.51	2811.72
\hat{Z}_{res} :	0.0034	0.267	0.069	0.037
CPU time (s):	0.01	7.22	2.03	0.21
$M^*(t)$	min	max	mean	median
iterations:	4	9998	2158.71	45
Z_{res} :	122.09	30084.45	3730.36	2480.72
\hat{Z}_{res} :	0.0034	0.255	0.048	0.034
CPU time (s):	0.01	7.34	1.13	0.07

One can see that when using $M^*(t)$ as reference, half the time is required to achieve double precision (compare Z_{res} values). The normalized values (\hat{Z}_{res}) have also significantly decreased which means better fits. As before, we evaluated all curve-fitting results, classified them, and plotted the \hat{Z}_{res} values (Figure 6).

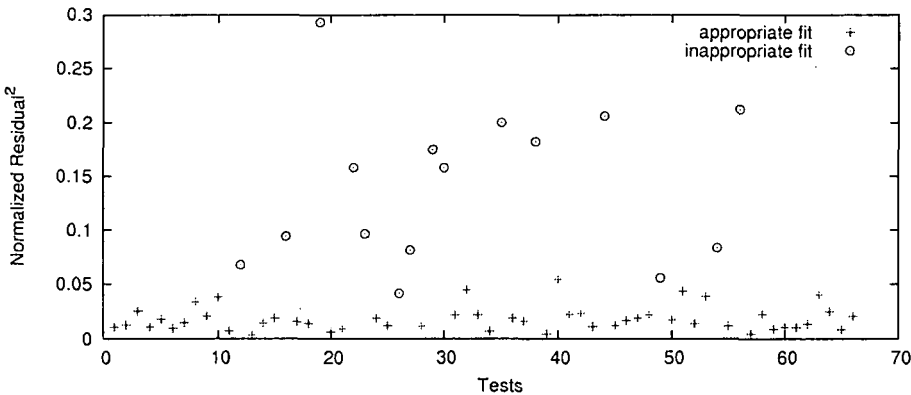


Figure 6: Normalized results on the test database (filtered fit)

Notice that the bad fit count decreased and the interface between appropriate and inappropriate classes sharpened which let us conclude that using the filtered signal is far better than using the original one.

3 Initial guess computation for LMA

Until now, we used a static, constant initial vector P_0 for the LM algorithm. Given the nature of the source experiments, real solution vectors are expected to be scattered in space. Scattering means greater search space, which indicates that a static initial vector for the LMA is in general a bad choice.

We propose an algorithm which is able to dynamically compute an excellent approximation of the estimator based on the filtered sample in $\mathcal{O}(1)$ time and $\mathcal{O}(n)$ space. Further advantage of our proposed algorithm is that it can also be used in other applications where the same (or analytically equivalent) model is applied.

It is known, that fluid injection is scheduled to be one second after the start of the recording, so the estimation of the Z_l parameter is trivial; we have to compute the arithmetical mean of the first 15 values of $M^*(t)$.

To estimate the other 4 parameters, we have computed the first order $H'(t)$ and second order $H''(t)$ derivatives of $H(t)$ and solved the $H'(t) = 0$ and $H''(t) = 0$ equations for identifying minimizer, maximizer and inflection points. The results showed that $H(t)$ has only one single maximum point (at t_{max}), and two inflection points (at $t_{i,1}$ and $t_{i,2}$) which can be expressed with α, β , and AT :

$$\begin{aligned} t_{max} &= \alpha\beta + AT, \\ t_{i,1} &= \alpha\beta - \sqrt{\alpha}\beta + AT, \\ t_{i,2} &= \alpha\beta + \sqrt{\alpha}\beta + AT. \end{aligned}$$

In this way, if we were able to produce a good estimation for t_{max} , $t_{i,1}$ and $t_{i,2}$, by solving the above nonlinear systems of equations we would acquire good estimations for AT , α , and β . However, the solution of this NLP problem is hard, complex NLP solvers are likely to introduce further errors, that is why we have chosen a simpler and faster heuristic method.

Taking the above three equations the following expressions can be derived:

$$\alpha = \frac{(t_{max} - AT)^2}{(t_{max} - t_{i,1})^2} = \frac{(t_{max} - AT)^2}{(t_{i,2} - t_{max})^2}, \quad (5)$$

$$\beta = \frac{(t_{max} - t_{i,1})^2}{t_{max} - AT} = \frac{(t_{i,2} - t_{max})^2}{t_{max} - AT}. \quad (6)$$

Since the AT parameter (the x-ray fluid appearance time) can easily be detected on $M^*(t)$, and given the t_{max} and one of the $t_{i,1}$ and $t_{i,2}$ values, α and β are directly computable using equations (5) and (6). The estimation of the AT parameter is done by combining zero and first order assumptions on the ideal model:

$$\begin{aligned} AT &\approx \frac{1}{3} \arg \max_t (M^*(t) = 0) + \frac{1}{3} \arg \max_t (M^*(t) = Z) + \\ &+ \frac{1}{3} \arg \min_t (M^*(t) > Z). \end{aligned} \quad (7)$$

Last but not least, an estimation for the K_s parameter must be given. This scaling is determined by the maxima of $G(t)$ (2). Given that

$$H(t_{max}) = (\alpha\beta)^\alpha e^{-\frac{\alpha\beta}{\beta}} + Z = (\alpha\beta)^\alpha e^{-\alpha} + Z,$$

an approximation for K_s can be formulated as

$$K_s \approx \frac{\max(M^*(t)) - Z}{(\alpha\beta)^\alpha e^{-\alpha}}. \tag{8}$$

Summarizing our method, we must

- 1. localize the maximum point t_{max} and maxima of $M^*(t)$,
- 2. localize at least one inflection point ($t_{i,1}$ and/or $t_{i,2}$) by searching discrete approximated roots of $M^{*'}(t)$,
- 3. compute an approximation for AT given the equation 7,
- 4. compute approximations for α , β , and K_s using equations 5, 6, and 8.

One may wonder how good estimation is computed by the proposed algorithm. To astonish the reader, we present some numerical and graphical results. Table 2 shows the general and normalized residuals as if the initial guess was our final solution vector.

Table 2: Computed initial guess vector evaluation

	min	max	mean	median
Z_{res} :	235.88	17085.28	4009.98	343.64
\hat{Z}_{res} :	0.0090	0.2379	0.0436	0.02719
CPU time (s):	0.01	0.26	0 .025	0.02

If compared to Table 1, it is in full view that the newly proposed algorithm performs much better than using the LMA with static starting point. Of course, being an initial vector we can achieve further improvements (Table 3).

Table 3: LMA results using computed initial guess

$M^*(t)$	min	max	mean	median
Z_{res} :	120.45	11245.8	3210.77	134.66
\hat{Z}_{res} :	0.007	0.231	0.0316	0.0187
CPU time (s):	0.01	2.54	0.42	0.05

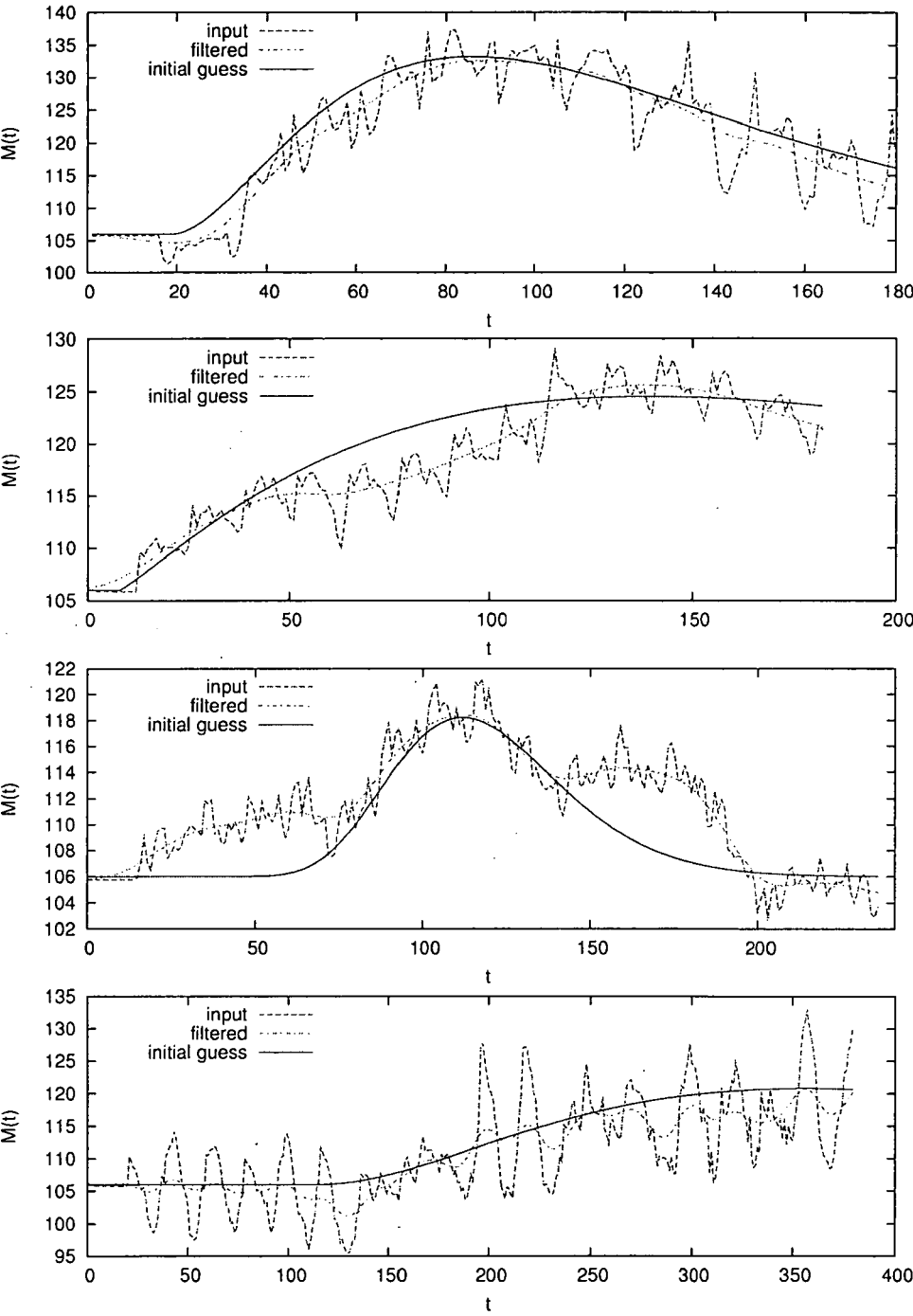


Figure 7: Initial guess computation results on the test database

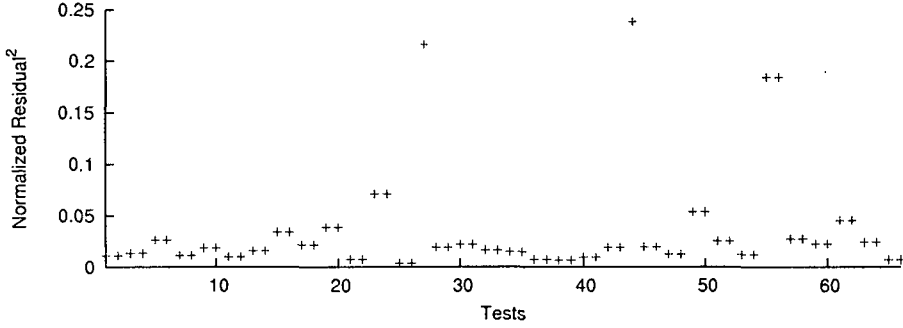


Figure 8: Normalized results with dynamic initial vector

In Figure 7 four results are shown along with the original and filtered samples. Notice that how diverse could be the shape of the input measurements. In Figure 8 showing the normalized residuals for the new approach, each fit has been classified as appropriate, higher values indicate wrong samples which must be dropped or re-recorded. At this point, the required efficiency and precision has been achieved.

4 Significant point selection

Although we could have stopped at this point, we was wondering how could we improve the physiological accuracy (not the numerical) of our solution. In order to achieve this new goal, the compression [2] of the input signal was the first step. All 200 (on average) intensity values in the filtered sample are too much for our model's 5 parameters [4]. Using less measurement values (about 20), we expected that our curve-fitting would be even more faster and accurate in a biological sense. The results showed a positive feedback.

Our basic idea was to classify the filtered sample's values as *significant* and *non-informational* points. To select the *significant* ones, we must detect those points where there are sudden changes in $M^*(t)$. Generally speaking, we want to approximates the curve with a polyline. Our point selection scheme is based on the first order discrete derivative of $M^*(t)$:

$$\partial M^*(t) = D_S(t) = \begin{cases} 0 & \text{if } t = 1 \\ M^*(t) - M^*(t-1) & \text{otherwise} \end{cases}$$

by dividing its codomain into a pre-defined number of intervals (C_I). The limits for an interval I_i can be computed using the following equations:

$$\begin{aligned} \underline{I}_i &= \min(D_S) + (i-1) \left(\frac{\max(D_S) - \min(D_S)}{C_I} \right), \\ \overline{I}_i &= \min(D_S) + i \left(\frac{\max(D_S) - \min(D_S)}{C_I} \right). \end{aligned}$$

The second part of the point selection is while scanning $D_S(t)$, we track some history on the previously seen values and note those moments where the previous point was located in another interval than the current one. This yields selected points near interval borders.

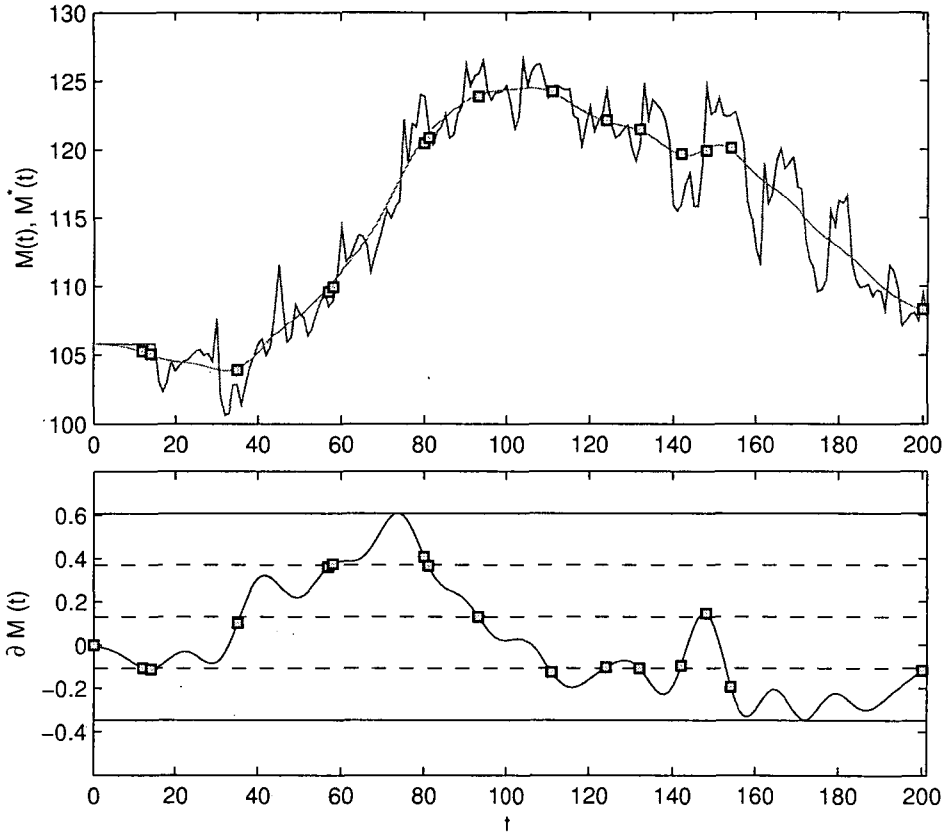


Figure 9: Significant point selection algorithm

The scheme of this method is shown in Figure 9, where the bottom graph shows the codomain of $D_S(t)$ divided into $C_I = 4$ equidistant intervals. The blue squares are the selected *significant* points which are then projected onto the top graph.

Of course, increasing C_I would increase the selected points, since the interval lengths would be smaller yielding more interval-border crossings. The proposed algorithm is designed to be driven by only one parameter – the *target significant point count*. The implementation is designed to select as many points as requested; more requested points mean more accurate approximation but less compression and vice versa. Also take note that our third proposed algorithm can also be used on any kind of a discrete sample in any dimensions.

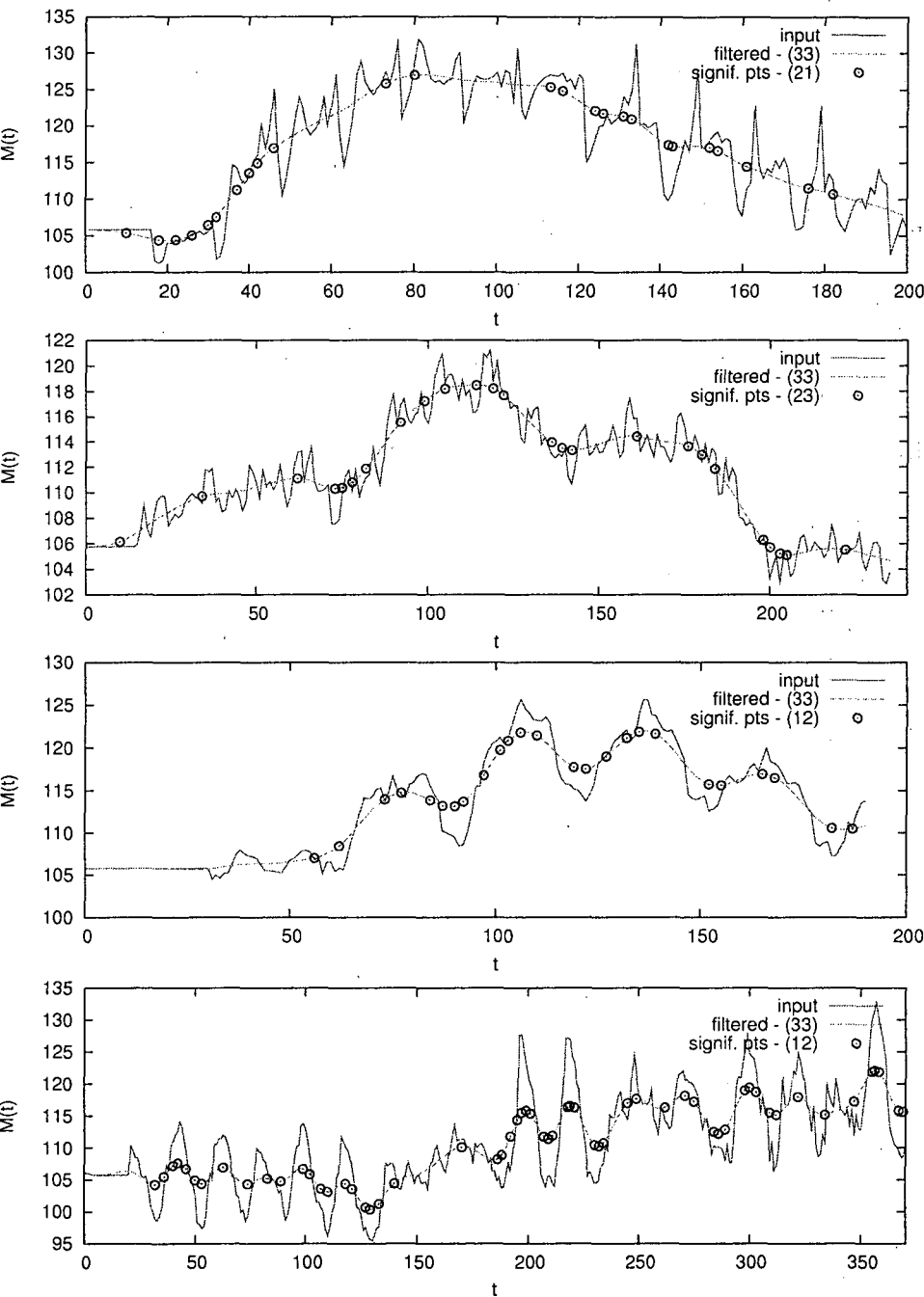


Figure 10: Significant point selection results

5 Final results

To summarize the work, our complex solution consists of a special filter, a pure mathematical initial guess computation algorithm, a measurement compression method and last but not least a NLLS Levenberg-Marquardt solver. This is also the order of their application, so after getting the initial sample, we apply our filter, compute an appropriate initial vector and select significant points using the filtered sample, and apply the LM optimizer with the pre-computed vector on the significant data points as empirical data.

We have successfully applied our solution for all the 66 real measurements at our disposal, selected 4 measurements to be re-recorded, and computed an excellent approximation of the model's parameters for the remaining 62 data sets. These have been validated by surgeons specialized in cardiovascular experiments and interventions at the Cardiovascular Research Laboratory at University of Szeged, Hungary.

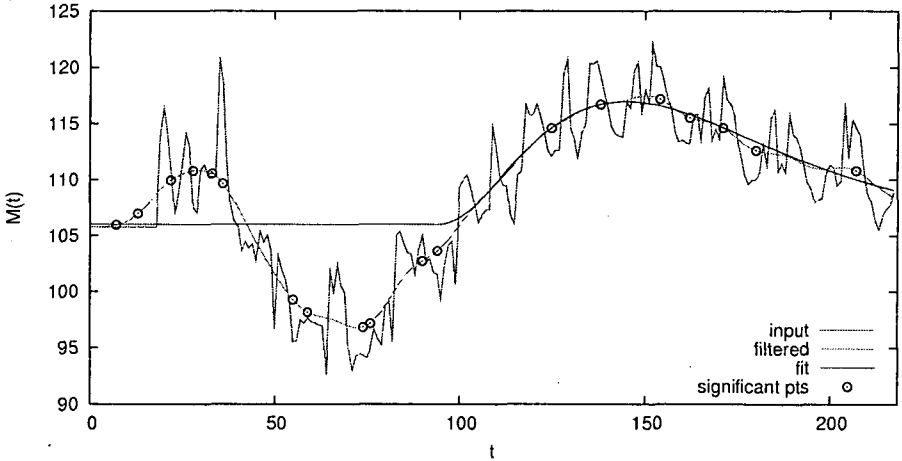


Figure 11: Composite result of a particular fit

Our composite solution technique is able to determine the validity of the measurement, then if it proves to be valid, we provide guaranteed results on any kind of input sample with high precision using no more than 2 seconds of computation time!¹

Comparing our solution with the time requirements of arranging the patient into the examination room, recording the x-ray video, image processing and ROI selection, we can surely say that our solution is really efficient and also effective enough to incorporate it into real-world devices.

¹Using an Intel Core 2 T2300, 4 GB RAM based PC

References

- [1] Abidov, Aiden, Germano, Guido, and Berman, Daniel. Transient ischemic dilation ratio: A universal high-risk diagnostic marker in myocardial perfusion imaging. *Journal of Nuclear Cardiology*, 14:497–500, 2007.
- [2] Ahmedaa, Sabah Mohamed and Abo-Zahhad, Mohammed. A new hybrid algorithm for ecg signal compression based on the wavelet transformation of the linearly predicted error. *Medical Engineering and Physics*, 23:117–126, 2001.
- [3] Brown, Lisa Gottesfeld. A survey of image registration techniques. *ACM Computing Surveys*, 24:325–376, 1992.
- [4] de Graaf, P., v. Goudoever, J., and Wesseling, K. Compressed storage of arterial pressure waveforms by selection of significant points. *Medical and Biological Engineering and Computing*, 35:510–515, 1997.
- [5] Evans, R. L. Two comments on the estimation of blood flow and central volume from dyedilution curves. *J. Appl. Physiol.*, 14(3):457, 1959.
- [6] Gao, Zhenxiao, Xiao, Tianyuan, and Fan, Wenhui. Hybrid differential evolution and neldermead algorithm with re-optimization. *Soft Computing*, pages 1–14, 2010.
- [7] Ghogho, Mounir, Swami, Ananthram, and Asoke, K.N. Non-linear least squares estimation for harmonics in multiplicative and additive noise. In *Signal Processing*, pages 43–60, 1999.
- [8] Gibson, CM., Anshelevich, M., and Murphy, S. Impact of injections during diagnostic coronary angiography on coronary patency in the setting of acute myocardial infarction from the timi trials. *Am J Cardiol.*, 86(12):1378–1379, 2000.
- [9] Gibson, CM, Kirtane, AJ, and Murphy, S. Impact of contrast agent type (ionic versus non-ionic) used for coronary angiography on angiographic, electrocardiographic and clinical outcomes following thrombolytic administration in acute mi. *Catheter Cardiovasc Interv.*, 53:6–11, 2001.
- [10] Gibson, CM. and Schmig, A. Coronary and myocardial angiography: angiographic assessment of both epicardial and myocardial perfusion. *Circulation*, 109(25):3096–3105, 2004.
- [11] Gobbel, Glenn T., Christopher, Dvm, Cann, E., and Fike John, R. 768 measurement of regional cerebral blood flow using ultrafast computed tomography theoretical aspects.
- [12] Haddad, R.A and Akansu, A.N. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 39:723–727, 1991.

- [13] Howard K. Thompson, Jr., Frank Starmer, C., Whalen, Robert E., and McIntosh, Henry D. Indicator transit time considered as a gamma variate. *Circulation Research, American Heart Association*, 15:502–515, 1964.
- [14] Lee, Sik-Yum and Zhu, Hong-Tu. Maximum likelihood estimation of nonlinear structural equation models. *Psychometrika*, 67:189–210, 2002.
- [15] Lourakis, Manolis I. A. and Argyros, Antonis A. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?, 2005.
- [16] Madsen, Kaj, Bruun, Hans, and Imm, Ole Tingleff. Methods for non-linear least squares problems. Technical report, 2004.
- [17] Meijering, E.H.W., Niessen, W.J., and Viergever, M.A. Retrospective motion correction in digital subtraction angiography: A review. *IEEE Transactions on Medical Imaging*, 18:2–21, 1999.
- [18] Meijering, E.H.W., Zuiderveld, K.J., and Viergever, M.A. Image registration for digital subtraction angiography. *International Journal of Computer Vision*, 31:227–246, 1999. 10.1023/A:1008074100927.
- [19] Moré, Jorge J. The Levenberg-Marquardt algorithm: Implementation and theory. In Watson, G. A., editor, *Numerical Analysis*, pages 105–116. Springer, Berlin, 1977.
- [20] Savitzky, Abraham and Golay, M. J. E. Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.*, 36:1627–1639, 6 1964.
- [21] Schafer, Sebastian, Nol, Peter B., Walczak, Alan M., and Hoffmann, Kenneth R. Filtered region of interest cone-beam rotational angiography. *Medical Physics*, 37:694–704, 2010.
- [22] Sensky, Penelope R., Samani, Nilesh J., Reek, Christine, and Cherryman, Graham R. *Cardiac Imaging*, 18:373–383, 2002.
- [23] Smietanski, Marek. Convergence of a generalized newton and an inexact generalized newton algorithms for solving nonlinear equations with nondifferentiable terms. *Numerical Algorithms*, 50:401–415, 2009.
- [24] Stephenson, John. Theory of the measurement of blood flow by the dilution of an indicator. *Bulletin of Mathematical Biology*, 10:117–121, 1948.
- [25] Warner, Homer R. Analysis of the role of indicator technics in quantitation of valvular regurgitation. *Circ Res*, 10(3):519–529, 1962.
- [26] Yoo, T., Ackerman, M., Lorensen, W., Schroeder, W., Chalana, V., Aylward, S., Metaxas, D., and Whitaker, R. Engineering and algorithm design for an image processing api: A technical report on itk - the insight toolkit. 01 2002.

CONTENTS

Conference of PhD Students in Computer Science	1
Preface	3
<i>Bálint Antal and András Hajdu: A Stochastic Approach to Improve Macula Detection in Retinal Images</i>	5
<i>Tamás Bartók and Csanád Imreh: Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints</i>	17
<i>András Bóta, Miklós Krész, and András Pluhár: Dynamic Communities and their Detection</i>	35
<i>Tibor Dobján, Szilveszter Pletl, Tamás Deák, László Doszpod, and Gábor Pór: Identification of the Place and Materials of Knocking Objects in Flow Induced Vibration</i>	53
<i>Erika Griechisch and András Pluhár: Community Detection by using the Extended Modularity</i>	69
<i>Péter Kardos: Sufficient Conditions for Order-Independency in Sequential Thinning</i>	87
<i>Gergely Klár and Gábor Valasek: Employing Pythagorean Hodograph Curves for Artistic Patterns</i>	101
<i>Levente Lócsi: Calculating Non-Equidistant Discretizations Generated by Blaschke Products</i>	111
<i>Gábor Németh, Péter Kardos, and Kálmán Palágyi: 2D Parallel Thinning and Shrinking Based on Sufficient Conditions for Topology Preservation</i> .	125
<i>Rudolf Szendrei, István Elek, and Mátyás Márton: Knowledge-Based Raster- Vector Conversion of Topographic Maps</i>	145
<i>László Varga, Péter Balázs, and Antal Nagy: Projection Selection Depen- dency in Binary Tomography</i>	167
<i>Krisztian Veress and Tibor Csendes: Parameter Estimation of Flow- Measurement in Digital Angiography</i>	189

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János