Tomáš Vojnar
Lijun Zhang (Eds.)

ARCoSS
LNCS 11427

Tools and Algorithms for the Construction and Analysis of Systems

25th International Conference, TACAS 2019
Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019
Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I

Part I

ETAPS
EUROPEAN JOINT CONFERENCES ON THEORY & PRACTICE OF SOFTWARE

Springer Open

Tomáš Vojnar · Lijun Zhang (Eds.)

Tools and Algorithms for the Construction and Analysis of Systems

25th International Conference, TACAS 2019
Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019
Prague, Czech Republic, April 6–11, 2019
Proceedings, Part I

Springer Open

Editors
Tomáš Vojnar
Brno University of Technology
Brno, Czech Republic

Lijun Zhang
Chinese Academy of Sciences
Beijing, China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-17461-3 ISBN 978-3-030-17462-0 (eBook)
https://doi.org/10.1007/978-3-030-17462-0

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

ETAPS Foreword

Welcome to the 22nd ETAPS! This is the first time that ETAPS took place in the Czech Republic in its beautiful capital Prague.

ETAPS 2019 was the 22nd instance of the European Joint Conferences on Theory and Practice of Software. ETAPS is an annual federated conference established in 1998, and consists of five conferences: ESOP, FASE, FoSSaCS, TACAS, and POST. Each conference has its own Program Committee (PC) and its own Steering Committee (SC). The conferences cover various aspects of software systems, ranging from theoretical computer science to foundations to programming language developments, analysis tools, formal approaches to software engineering, and security.

Organizing these conferences in a coherent, highly synchronized conference program enables participation in an exciting event, offering the possibility to meet many researchers working in different directions in the field and to easily attend talks of different conferences. ETAPS 2019 featured a new program item: the Mentoring Workshop. This workshop is intended to help students early in the program with advice on research, career, and life in the fields of computing that are covered by the ETAPS conference. On the weekend before the main conference, numerous satellite workshops took place and attracted many researchers from all over the globe.

ETAPS 2019 received 436 submissions in total, 137 of which were accepted, yielding an overall acceptance rate of 31.4%. I thank all the authors for their interest in ETAPS, all the reviewers for their reviewing efforts, the PC members for their contributions, and in particular the PC (co-)chairs for their hard work in running this entire intensive process. Last but not least, my congratulations to all authors of the accepted papers!

ETAPS 2019 featured the unifying invited speakers Marsha Chechik (University of Toronto) and Kathleen Fisher (Tufts University) and the conference-specific invited speakers (FoSSaCS) Thomas Colcombet (IRIF, France) and (TACAS) Cormac Flanagan (University of California at Santa Cruz). Invited tutorials were provided by Dirk Beyer (Ludwig Maximilian University) on software verification and Cesare Tinelli (University of Iowa) on SMT and its applications. On behalf of the ETAPS 2019 attendants, I thank all the speakers for their inspiring and interesting talks!

ETAPS 2019 took place in Prague, Czech Republic, and was organized by Charles University. Charles University was founded in 1348 and was the first university in Central Europe. It currently hosts more than 50,000 students. ETAPS 2019 was further supported by the following associations and societies: ETAPS e.V., EATCS (European Association for Theoretical Computer Science), EAPLS (European Association for Programming Languages and Systems), and EASST (European Association of Software Science and Technology). The local organization team consisted of Jan Vitek and Jan Kofron (general chairs), Barbora Buhnova, Milan Ceska, Ryan Culpepper, Vojtech Horky, Paley Li, Petr Maj, Artem Pelenitsyn, and David Safranek.

The ETAPS SC consists of an Executive Board, and representatives of the individual ETAPS conferences, as well as representatives of EATCS, EAPLS, and EASST. The Executive Board consists of Gilles Barthe (Madrid), Holger Hermanns (Saarbrücken), Joost-Pieter Katoen (chair, Aachen and Twente), Gerald Lüttgen (Bamberg), Vladimiro Sassone (Southampton), Tarmo Uustalu (Reykjavik and Tallinn), and Lenore Zuck (Chicago). Other members of the SC are: Wil van der Aalst (Aachen), Dirk Beyer (Munich), Mikolaj Bojanczyk (Warsaw), Armin Biere (Linz), Luis Caires (Lisbon), Jordi Cabot (Barcelona), Jean Goubault-Larrecq (Cachan), Jurriaan Hage (Utrecht), Rainer Hähnle (Darmstadt), Reiko Heckel (Leicester), Panagiotis Katsaros (Thessaloniki), Barbara König (Duisburg), Kim G. Larsen (Aalborg), Matteo Maffei (Vienna), Tiziana Margaria (Limerick), Peter Müller (Zurich), Flemming Nielson (Copenhagen), Catuscia Palamidessi (Palaiseau), Dave Parker (Birmingham), Andrew M. Pitts (Cambridge), Dave Sands (Gothenburg), Don Sannella (Edinburgh), Alex Simpson (Ljubljana), Gabriele Taentzer (Marburg), Peter Thiemann (Freiburg), Jan Vitek (Prague), Tomas Vojnar (Brno), Heike Wehrheim (Paderborn), Anton Wijs (Eindhoven), and Lijun Zhang (Beijing).

I would like to take this opportunity to thank all speakers, attendants, organizers of the satellite workshops, and Springer for their support. I hope you all enjoy the proceedings of ETAPS 2019. Finally, a big thanks to Jan and Jan and their local organization team for all their enormous efforts enabling a fantastic ETAPS in Prague!

February 2019

Joost-Pieter Katoen
ETAPS SC Chair
ETAPS e.V. President

Preface

TACAS 2019 was the 25th edition of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems conference series. TACAS 2019 was part of the 22nd European Joint Conferences on Theory and Practice of Software (ETAPS 2019). The conference was held at the Orea Hotel Pyramida in Prague, Czech Republic, during April 8–11, 2019.

Conference Description. TACAS is a forum for researchers, developers, and users interested in rigorously based tools and algorithms for the construction and analysis of systems. The conference aims to bridge the gaps between different communities with this common interest and to support them in their quest to improve the utility, reliability, flexibility, and efficiency of tools and algorithms for building systems. TACAS 2019 solicited four types of submissions:

– *Research papers*, identifying and justifying a principled advance to the theoretical foundations for the construction and analysis of systems, where applicable supported by experimental validation.
– *Case-study papers*, reporting on case studies and providing information about the system being studied, the goals of the study, the challenges the system poses to automated analysis, research methodologies and approaches used, the degree to which goals were attained, and how the results can be generalized to other problems and domains.
– *Regular tool papers*, presenting a new tool, a new tool component, or novel extensions to an existing tool, with an emphasis on design and implementation concerns, including software architecture and core data structures, practical applicability, and experimental evaluations.
– *Tool-demonstration papers* (short), focusing on the usage aspects of tools.

Paper Selection. This year, 164 papers were submitted to TACAS, among which 119 were research papers, 10 case-study papers, 24 regular tool papers, and 11 were tool-demonstration papers. After a rigorous review process, with each paper reviewed by at least three Program Committee members, followed by an online discussion, the Program Committee accepted 29 research papers, 2 case-study papers, 11 regular tool papers, and 8 tool-demonstration papers (50 papers in total).

Artifact-Evaluation Process. The main novelty of TACAS 2019 was that, for the first time, artifact evaluation was compulsory for all regular tool papers and tool demonstration papers. For research papers and case-study papers, artifact evaluation was optional. The artifact evaluation process was organized as follows:

– *Regular tool papers and tool demonstration papers.* The authors of the 35 submitted papers of these categories of papers were required to submit an artifact alongside their paper submission. Each artifact was evaluated independently by three reviewers. Out of the 35 artifact submissions, 28 were successfully evaluated, which corresponds to an acceptance rate of 80%. The AEC used a two-phase

reviewing process: Reviewers first performed an initial check to see whether the artifact was technically usable and whether the accompanying instructions were consistent, followed by a full evaluation of the artifact. The main criterion for artifact acceptance was consistency with the paper, with completeness and documentation being handled in a more lenient manner as long as the artifact was useful overall. The reviewers were instructed to check whether results are consistent with what is described in the paper. Inconsistencies were to be clearly pointed out and explained by the authors. In addition to the textual reviews, reviewers also proposed a numeric value about (potentially weak) acceptance/rejection of the artifact. After the evaluation process, the results of the artifact evaluation were summarized and forwarded to the discussion of the papers, so as to enable the reviewers of the papers to take the evaluation into account. In all but three cases, tool papers whose artifacts did not pass the evaluation were rejected.
- *Research papers and case-study papers*. For this category of papers, artifact evaluation was voluntary. The authors of each of the 25 accepted papers were invited to submit an artifact immediately after the acceptance notification. Owing to the short time available for the process and acceptance of the artifact not being critical for paper acceptance, there was only one round of evaluation for this category, and every artifact was assigned to two reviewers. The artifacts were evaluated using the same criteria as for tool papers. Out of the 18 submitted artifacts of this phase, 15 were successfully evaluated (83% acceptance rate) and were awarded the TACAS 2019 AEC badge, which is added to the title page of the respective paper if desired by the authors.

TOOLympics. TOOLympics 2019 was part of the celebration of the 25th anniversary of the TACAS conference. The goal of TOOLympics is to acknowledge the achievements of the various competitions in the field of formal methods, and to understand their commonalities and differences. A total of 2^4 competitions joined TOOLympics and were presented at the event. An overview and competition reports of 11 competitions are included in the third volume of the TACAS 2019 proceedings, which are dedicated to the 25th anniversary of TACAS. The extra volume contains a review of the history of TACAS, the TOOLympics papers, and the papers of the annual Competition on Software Verification.

Competition on Software Verification. TACAS 2019 also hosted the 8th International Competition on Software Verification (SV-COMP), chaired and organized by Dirk Beyer. The competition again had high participation: 31 verification systems with developers from 14 countries were submitted for the systematic comparative evaluation, including three submissions from industry. The TACAS proceedings includes the competition report and short papers describing 11 of the participating verification systems. These papers were reviewed by a separate program committee (PC); each of the papers was assessed by four reviewers. Two sessions in the TACAS program (this year as part of the TOOLympics event) were reserved for the presentation of the results: the summary by the SV-COMP chair and the participating tools by the developer teams in the first session, and the open jury meeting in the second session.

Acknowledgments. We would like to thank everyone who helped to make TACAS 2019 successful. In particular, we would like to thank the authors for submitting their

papers to TACAS 2019. We would also like to thank all PC members, additional reviewers, as well as all members of the artifact evaluation committee (AEC) for their detailed and informed reviews and, in the case of the PC and AEC members, also for their discussions during the virtual PC and AEC meetings. We also thank the Steering Committee for their advice. Special thanks go to the Organizing Committee of ETAPS 2019 and its general chairs, Jan Kofroň and Jan Vitek, to the chair of the ETAPS 2019 executive board, Joost-Pieter Katoen, and to the publication team at Springer.

March 2019

Tomáš Vojnar (PC Chair)
Lijun Zhang (PC Chair)
Marius Mikucionis (Tools Chair)
Radu Grosu (Use-Case Chair)
Dirk Beyer (SV-COMP Chair)
Ondřej Lengál (AEC Chair)
Ernst Moritz Hahn (AEC Chair)

Organization

Program Committee

Michael Tautschnig	Queen Mary University of London/Amazon Web Services, UK
Tomáš Vojnar (Co-chair)	Brno University of Technology, Czech Republic
Thomas Wies	New York University, USA
Lijun Zhang (Co-chair)	Institute of Software, Chinese Academy of Sciences, China
Florian Zuleger	Vienna University of Technology, Austria

Program Committee and Jury—SV-COMP

Dirk Beyer (Chair)	LMU Munich, Germany
Peter Schrammel (Representing 2LS)	University of Sussex, UK
Jera Hensel (Representing AProVE)	RWTH Aachen, Germany
Michael Tautschnig (Representing CBMC)	Amazon Web Services, UK
Kareem Khazem (Representing CBMC-Path)	University College London, UK
Vadim Mutilin (Representing CPA-BAM-BnB)	ISP RAS, Russia
Pavel Andrianov (Representing CPA-Lockator)	ISP RAS, Russia
Marie-Christine Jakobs (Representing CPA-Seq)	LMU Munich, Germany
Omar Alhawi (Representing DepthK)	University of Manchester, UK
Vladimír Štill (Representing DIVINE-Explicit)	Masaryk University, Czechia
Henrich Lauko (Representing DIVINE-SMT)	Masaryk University, Czechia
Mikhail R. Gadelha (Representing ESBMC-Kind)	University of Southampton, UK
Philipp Ruemmer (Representing JayHorn)	Uppsala University, Sweden
Lucas Cordeiro (Representing JBMC)	University of Manchester, UK
Cyrille Artho (Representing JPF)	KTH, Denmark
Omar Inverso (Representing Lazy-CSeq)	Gran Sasso Science Inst., Italy
Herbert Rocha (Representing Map2Check)	Federal University of Roraima, Brazil
Cedric Richter (Representing PeSCo)	University of Paderborn, Germany

Artifact Evaluation Committee (AEC)

Le Quang Loc	Teesside University, UK
Rasool Maghareh	National University of Singapore, Singapore
Tobias Meggendorfer	TU Munich, Germany
Malte Mues	TU Dortmund, Germany
Tuan Phong Ngo	Uppsala, Sweden
Chris Novakovic	University of Birmingham, UK
Thai M. Trinh	Advanced Digital Sciences Center, Illinois at Singapore, Singapore
Wytse Oortwijn	University of Twente, The Netherlands
Aleš Smrčka	Brno University of Technology, Czech Republic
Daniel Stan	Saarland University, Germany
Ilina Stoilkovska	TU Wien, Austria
Ming-Hsien Tsai	Academia Sinica, Taiwan
Jan Tušil	Masaryk University, Czech Republic
Pedro Valero	IMDEA, Spain
Maximilian Weininger	TU Munich, Germany

Additional Reviewers

Aiswarya, C.
Albarghouthi, Aws
Aminof, Benjamin
Américo, Arthur
Ashok, Pranav
Atig, Mohamed Faouzi
Bacci, Giovanni
Bainczyk, Alexander
Barringer, Howard
Basset, Nicolas
Bensalem, Saddek
Berard, Beatrice
Besson, Frédéric
Biewer, Sebastian
Bogomolov, Sergiy
Bollig, Benedikt
Bozga, Marius
Bozzano, Marco
Brazdil, Tomas
Caulfield, Benjamin
Chaudhuri, Swarat
Cheang, Kevin
Chechik, Marsha
Chen, Yu-Fang
Chin, Wei-Ngan
Chini, Peter
Ciardo, Gianfranco
Cohen, Liron
Cordeiro, Lucas
Cyranka, Jacek
Čadek, Pavel
Darulova, Eva
Degorre, Aldric
Delbianco, Germán Andrés
Delzanno, Giorgio
Devir, Nurit
Dierl, Simon
Dragoi, Cezara
Dreossi, Tommaso
Dutra, Rafael
Eilers, Marco
El-Hokayem, Antoine
Faella, Marco
Fahrenberg, Uli
Falcone, Ylies
Fox, Gereon
Freiberger, Felix
Fremont, Daniel
Frenkel, Hadar
Friedberger, Karlheinz
Frohme, Markus
Fu, Hongfei

Furbach, Florian
Garavel, Hubert
Ghosh, Bineet
Ghosh, Shromona
Gondron, Sebastien
Gopinath, Divya
Gossen, Frederik
Goyal, Manish
Graf-Brill, Alexander
Griggio, Alberto
Gu, Tianxiao
Guatto, Adrien
Gutiérrez, Elena
Hahn, Ernst Moritz
Hansen, Mikkel
Hartmanns, Arnd
Hasani, Ramin
Havlena, Vojtěch
He, Kangli
He, Pinjia
Hess, Andreas Viktor
Heule, Marijn
Ho, Mark
Ho, Nhut Minh
Holik, Lukas
Hsu, Hung-Wei
Inverso, Omar
Irfan, Ahmed
Islam, Md. Ariful
Itzhaky, Shachar
Jakobs, Marie-Christine
Jaksic, Stefan
Jasper, Marc
Jensen, Peter Gjøl
Jonas, Martin
Kaminski, Benjamin Lucien
Karimi, Abel
Katelaan, Jens
Kauffman, Sean
Kaufmann, Isabella
Khoo, Siau-Cheng
Kiesl, Benjamin
Kim, Eric
Klauck, Michaela
Kong, Hui
Kong, Zhaodan
Kopetzki, Dawid
Krishna, Siddharth
Krämer, Julia
Kukovec, Jure
Kumar, Rahul
Köpf, Boris
Lange, Martin
Le Coent, Adrien
Lemberger, Thomas
Lengal, Ondrej
Li, Yi
Lin, Hsin-Hung
Lluch Lafuente, Alberto
Lorber, Florian
Lu, Jianchao
Lukina, Anna
Lång, Magnus
Maghareh, Rasool
Mahyar, Hamidreza
Markey, Nicolas
Mathieson, Luke
Mauritz, Malte
Mayr, Richard
Mechtaev, Sergey
Meggendorfer, Tobias
Micheli, Andrea
Michelmore, Rhiannon
Monteiro, Pedro T.
Mover, Sergio
Mu, Chunyan
Mues, Malte
Muniz, Marco
Murano, Aniello
Murtovi, Alnis
Muskalla, Sebastian
Mutluergil, Suha Orhun
Neumann, Elisabeth
Ngo, Tuan Phong
Nickovic, Dejan
Nies, Gilles
Noller, Yannic
Norman, Gethin
Nowack, Martin
Olmedo, Federico
Pani, Thomas
Petri, Gustavo

Piazza, Carla
Poli, Federico
Poulsen, Danny Bøgsted
Prabhakar, Pavithra
Quang Trung, Ta
Ranzato, Francesco
Rasmussen, Cameron
Ratasich, Denise
Ravanbakhsh, Hadi
Ray, Rajarshi
Reger, Giles
Reynolds, Andrew
Rigger, Manuel
Rodriguez, Cesar
Rothenberg, Bat-Chen
Roveri, Marco
Rydhof Hansen, René
Rüthing, Oliver
Sadeh, Gal
Saivasan, Prakash
Sanchez, Cesar
Sangnier, Arnaud
Schlichtkrull, Anders
Schwoon, Stefan
Seidl, Martina
Shi, Xiaomu
Shirmohammadi, Mahsa
Shoukry, Yasser
Sighireanu, Mihaela
Soudjani, Sadegh
Spießl, Martin
Srba, Jiri
Srivas, Mandayam
Stan, Daniel
Stoilkovska, Ilina
Stojic, Ivan
Su, Ting
Summers, Alexander J.
Tabuada, Paulo
Tacchella, Armando
Tang, Enyi
Tian, Chun
Tonetta, Stefano
Trinh, Minh-Thai
Trtík, Marek
Tsai, Ming-Hsien
Valero, Pedro
van der Berg, Freark
Vandin, Andrea
Vazquez-Chanlatte, Marcell
Viganò, Luca
Villadsen, Jørgen
Wang, Shuai
Wang, Shuling
Weininger, Maximilian
Wendler, Philipp
Wolff, Sebastian
Wüstholz, Valentin
Xu, Xiao
Zeljić, Aleksandar
Zhang, Fuyuan
Zhang, Qirun
Zhang, Xiyue

Contents – Part I

Contents – Part II

VoxLogicA: A Spatial Model Checker for Declarative Image Analysis

Gina Belmonte[1], Vincenzo Ciancia[2(✉)], Diego Latella[2], and Mieke Massink[2]

[1] Azienda Ospedaliera Universitaria Senese, Siena, Italy
[2] Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy
vincenzo.ciancia@isti.cnr.it

Abstract. Spatial and spatio-temporal model checking techniques have a wide range of application domains, among which large scale distributed systems and signal and image analysis. We explore a new domain, namely (semi-)automatic contouring in Medical Imaging, introducing the tool `VoxLogicA` which merges the state-of-the-art library of computational imaging algorithms `ITK` with the unique combination of declarative specification and optimised execution provided by spatial logic model checking. The result is a *rapid*, logic based analysis development methodology. The analysis of an existing benchmark of medical images for segmentation of brain tumours shows that simple `VoxLogicA` analysis can reach state-of-the-art accuracy, competing with best-in-class algorithms, with the advantage of *explainability* and easy *replicability*. Furthermore, due to a two-orders-of-magnitude speedup compared to the existing *general-purpose* spatio-temporal model checker `topochecker`, `VoxLogicA` enables *interactive* development of analysis of 3D medical images, which can greatly facilitate the work of professionals in this domain.

Keywords: Spatial logics · Closure spaces · Model checking · Medical Imaging

1 Introduction and Related Work

Spatial and Spatio-temporal model checking have gained an increasing interest in recent years in various domains of application ranging from Collective Adaptive Systems [11,15,18] and networked systems [27], to *signals* [32] and *digital images* [14,26]. Research in this field has its origin in the *topological* approach to spatial logics, dating back to the work of Alfred Tarski (see [9] for a thorough introduction). More recently these early theoretical foundations have been extended to encompass reasoning about *discrete* spatial structures, such as graphs and images, extending the theoretical framework of topology to *(quasi discrete) closure spaces* (see for instance [1,23,24]). That framework has subsequently been taken further in recent work by Ciancia et al. [13,14,17] resulting in the definition of the *Spatial Logic for Closure Spaces* (SLCS), temporal extensions (see [12,32,36]), and related model checking algorithms and tools.

The main idea of spatial (and spatio-temporal) model checking is to use specifications written in a suitable logical language to describe spatial properties and to automatically identify patterns and structures of interest in a variety of domains (see e.g., [5,16,18]). In this paper we focus on one such domain, namely medical imaging for radiotherapy, and brain tumour segmentation in particular, which is an important and currently very active research domain of its own. One of the technical challenges of the development of automated (brain) tumour segmentation is that lesion areas are only defined through differences in the intensity (luminosity) in the (black & white) images that are *relative* to the intensity of the surrounding normal tissue. A further complication is that even (laborious and time consuming) manual segmentation by experts shows significant variations when intensity gradients between adjacent tissue structures are smooth or partially obscured [31]. Moreover, there is a considerable variation across images from different patients and images obtained with different Magnetic Resonance Images (MRI) scanners. Several automatic and semi-automatic methods have been proposed in this very active research area (see e.g., [20–22,29,34,37]).

This paper continues the research line of [3,7,8], introducing the free and open source tool `VoxLogicA` (*Voxel-based Logical Analyser*)[1], catering for a novel approach to image segmentation, namely a *rapid-development*, declarative, logic-based method, supported by *spatial model checking*. This approach is particularly suitable to reason at the "macro-level", by exploiting the *relative* spatial relations between tissues or organs at risk. `VoxLogicA` is similar, in the accepted logical language, and functionality, to the spatio-temporal model checker `topochecker`[2], but specifically designed for the analysis of (possibly multi-dimensional, e.g. 3D) *digital images* as a specialised image analysis tool. It is tailored to usability and efficiency by employing state-of-the-art algorithms and open source libraries, borrowed from computational image processing, in combination with efficient spatial model checking algorithms.

We show the application of `VoxLogicA` on BraTS 2017[3] [2,31,35], a publicly available set of benchmark MRI images for brain tumour segmentation, linked to a yearly challenge. For each image, a manual segmentation of the tumour by domain experts is available, enabling rigorous and objective qualitative comparisons via established similarity indexes. We propose a simple, yet effective, high-level specification for glioblastoma segmentation. The procedure, partly derived from the one presented in [3], directly competes in accuracy with the state-of-the-art techniques submitted to the BraTS 2017 challenge, most of which based on machine learning. Our approach to segmentation has the unique advantage of *explainability*, and is easy to replicate; in fact, the structure of a logically specified procedure can be explained to domain experts, and improved to encompass new observations. A mathematically formalised, unambiguous semantics permits results to be replicated not only by executing them in the multi-platform, open source tool that has been provided, but also by computing them via different implementations.

[1] `VoxLogicA`: https://github.com/vincenzoml/VoxLogicA.

[2] Topochecker: *a topological model checker*, see http://topochecker.isti.cnr.it, https://github.com/vincenzoml/topochecker.

[3] See https://www.med.upenn.edu/sbia/brats2017/data.html.

2 The Spatial Logic Framework

In this section, we briefly recall the logical language ImgQL (*Image Query Language*) proposed in [3], which is based on the *Spatial Logic for Closure Spaces* SLCS [13,14] and which forms the *kernel* of the framework we propose in the present paper. In Sect. 4 we will see how the resulting logic can be used for actual analysis via spatial model checking.

2.1 Foundations: Spatial Logics for Closure Spaces

The logic for closure spaces we use in the present paper is closely related to SLCS [13,14] and, in particular, to the SLCS extension with distance-based operators presented in [3]. As in [3], the resulting logic constitutes the *kernel* of a solid logical framework for reasoning about texture features of digital images, when interpreted as closure spaces. In the context of our work, a *digital image* is not only a 2-dimensional grid of *pixels*, but, more generally, a multi-dimensional (very often, 3-dimensional) grid of hyper-rectangular elements that are called *voxels* ("volumetric picture elements"). When voxels are not *hypercubes*, images are said to be *anisotropic*; this is usually the case in medical imaging. Furthermore, a digital image may contain information about its "real world" spatial dimensions, position (origin) and rotation, permitting one to compute the real-world coordinates of the centre and edges of each voxel. In medical imaging, such information is typically encapsulated into data by machines such as MRI scanners. In the remainder of the paper, we make no dimensionality assumptions. From now on, we refer to picture elements either as voxels or simply as points.

Definition 1. *A* closure space *is a pair* $(X, \mathcal{C})$ *where* X *is a non-empty set (of* points*) and* $\mathcal{C} : 2^X \to 2^X$ *is a function satisfying the following axioms:* $\mathcal{C}(\emptyset) = \emptyset$*;* $Y \subseteq \mathcal{C}(Y)$ *for all* $Y \subseteq X$*;* $\mathcal{C}(Y_1 \cup Y_2) = \mathcal{C}(Y_1) \cup \mathcal{C}(Y_2)$ *for all* $Y_1, Y_2 \subseteq X$*.* •

Given any relation $R \subseteq X \times X$, function $\mathcal{C}_R : 2^X \to 2^X$ with $\mathcal{C}_R(Y) \triangleq Y \cup \{x \mid \exists y \in Y. y\, R\, x\}$ satisfies the axioms of Definition 1 thus making $(X, \mathcal{C}_R)$ a closure space. Whenever a closure space is generated by a relation as above, it is called a *quasi-discrete* closure space. A quasi-discrete closure space $(X, \mathcal{C}_R)$, can be used as the basis for a mathematical model of a digital image. X represents the finite set of *voxels* and R is the reflexive and symmetric *adjacency* relation between voxels [25]. A closure space $(X, \mathcal{C})$ can be enriched with a notion of *distance*, i.e. a function $d : X \times X \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ such that $d(x, y) = 0$ iff $x = y$, leading to the *distance closure space* $((X, \mathcal{C}), d)$.[4]

[4] We recall that for $\emptyset \neq Y \subseteq X$, $d(x, Y) \triangleq \inf\{d(x, y) \mid y \in Y\}$, with $d(x, \emptyset) = \infty$. In addition, as the definition of d might require the elements of R to be weighted, quasi-discrete distance closure spaces may be enriched with a R-weighting function $\mathcal{W} : R \to \mathbb{R}$ assigning the weight $\mathcal{W}(x, y)$ to each $(x, y) \in R$. In the sequel we will keep $\mathcal{W}$ implicit, whenever possible and for the sake of simplicity.

It is sometimes convenient to equip the points of a closure space with *attributes*; for instance, in the case of images, such attributes could be the color or intensity of voxels. We assume sets A and V of attribute *names* and *values*, and an *attribute valuation* function $\mathcal{A}$ such that $\mathcal{A}(x,a) \in V$ is the value of attribute a of point x. Attributes can be used in *assertions* α, i.e. boolean expressions, with standard syntax and semantics. Consequently, we abstract from related details here and assume function $\mathcal{A}$ extended in the obvious way; for instance, $\mathcal{A}(x, a \leq c) = \mathcal{A}(x,a) \leq c$, for appropriate constant c.

A (quasi-discrete) *path* π in $(X, \mathcal{C}_R)$ is a function $\pi : \mathbb{N} \to X$, such that for all $Y \subseteq \mathbb{N}$, $\pi(\mathcal{C}_{Succ}(Y)) \subseteq \mathcal{C}_R(\pi(Y))$, where $(\mathbb{N}, \mathcal{C}_{Succ})$ is the closure space of natural numbers with the *successor* relation: $(n,m) \in Succ \Leftrightarrow m = n+1$. Intuitively: the ordering in the path imposed by $\mathbb{N}$ is compatible with relation R, i.e. $\pi(i)\, R\, \pi(i+1)$. For given set P of *atomic predicates* p, and interval of $\mathbb{R}$ I, the syntax of the logic we use in this paper is given below:

$$\Phi ::= p \mid \neg \Phi \mid \Phi_1 \wedge \Phi_2 \mid \mathcal{N}\Phi \mid \rho\, \Phi_1[\Phi_2] \mid \mathcal{D}^I \Phi \tag{1}$$

We assume that space is modelled by the set of points of a distance closure space; each atomic predicate $p \in P$ models a specific *feature* of *points* and is thus associated with the points that have this feature[5]. A point x satisfies $\mathcal{N}\,\Phi$ if a point satisfying Φ can be reached from x in at most one (closure) step, i.e. if x is *near* (or *close*) to a point satisfying Φ; x satisfies $\rho\ \Phi_1[\Phi_2]$ if x *may reach* a point satisfying Φ_1 via a path passing only by points satisfying Φ_2; it satisfies $\mathcal{D}^I\,\Phi$ if its distance from the set of points satisfying Φ falls in interval I. The logic includes logical negation ($\neg$) and conjunction ($\wedge$). In the following we formalise the semantics of the logic. A *distance closure model* $\mathcal{M}$ is a tuple $\mathcal{M} = (((X, \mathcal{C}), d), \mathcal{A}, \mathcal{V})$, where $((X, \mathcal{C}), d)$ is a distance closure space, $\mathcal{A} : X \times A \to V$ an attribute valuation, and $\mathcal{V} : P \to 2^X$ is a valuation of atomic propositions.

Definition 2. Satisfaction $\mathcal{M}, x \models \Phi$ *of a formula* Φ *at point* $x \in X$ *in model* $\mathcal{M} = (((X, \mathcal{C}), d), \mathcal{A}, \mathcal{V})$ *is defined by induction on the structure of formulas:*

$$\begin{array}{lll}
\mathcal{M}, x \models p \in P & \Leftrightarrow & x \in \mathcal{V}(p) \\
\mathcal{M}, x \models \neg\,\Phi & \Leftrightarrow & \mathcal{M}, x \models \Phi \textit{ does not hold} \\
\mathcal{M}, x \models \Phi_1 \wedge \Phi_2 & \Leftrightarrow & \mathcal{M}, x \models \Phi_1 \textit{ and } \mathcal{M}, x \models \Phi_2 \\
\mathcal{M}, x \models \mathcal{N}\,\Phi & \Leftrightarrow & x \in \mathcal{C}(\{y \mid \mathcal{M}, y \models \Phi\}) \\
\mathcal{M}, x \models \rho\ \Phi_1[\Phi_2] & \Leftrightarrow & \textit{there is path } \pi \textit{ and index } \ell \textit{ s.t. } \pi(0) = x \textit{ and } \mathcal{M}, \pi(\ell) \models \Phi_1 \\
 & & \textit{and for all indexes } j : 0 < j < \ell \textit{ implies } \mathcal{M}, \pi(j) \models \Phi_2 \\
\mathcal{M}, x \models \mathcal{D}^I\,\Phi & \Leftrightarrow & d(x, \{y \mid \mathcal{M}, y \models \Phi\}) \in I
\end{array}$$

where, when $p := \alpha$ *is a definition for* p, *we let* $x \in \mathcal{V}(p)$ *iff* $\mathcal{A}(x, \alpha)$ *is true.* •

[5] In particular, a predicate p can be a *defined* one, by means of a definition as $p := \alpha$, meaning that the feature of interest is characterized by the (boolean) value of α.

In the logic proposed in [13,14], the "may reach" operator is not present, and the *surrounded* operator $\mathcal{S}$ has been defined as basic operator as follows: x satisfies $\Phi_1 \mathcal{S} \Phi_2$ if and only if x belongs to an area of points satisfying Φ_1 and one cannot "escape" from such an area without hitting a point satisfying Φ_2. Several types of *reachability* predicates can be derived from $\mathcal{S}$. However, reachability is in turn a widespread, more basic primitive, implemented in various forms (e.g., *flooding*, *connected components*) in programming libraries. Thus, in this work, we prefer to use reachability as a basic predicate of the logic, as in [4], which is dedicated to extending the *Spatial Signal Temporal Logic* of [32]. In the sequel we show that $\mathcal{S}$ can be derived from the operators defined above, employing a definition patterned after the model-checking algorithm of [13]. This change simplifies the definition of several derived connectives, including that of *touch* (see below), and resulted in notably faster execution times for analyses using such derived connectives. We recall the definition of $\mathcal{S}$ from [14]: $\mathcal{M}, x \models \Phi_1 \mathcal{S} \Phi_2$ if and only if $\mathcal{M}, x \models \Phi_1$ and for all paths π and indexes ℓ we have: if $\pi(0) = x$ and $\mathcal{M}, \pi(\ell) \models \neg\Phi_1$, then there is j such that $0 < j \leq \ell$ and $\mathcal{M}, \pi(j) \models \Phi_2$.

Proposition 1. *For all closure models* $\mathcal{M} = ((X, \mathcal{C}), \mathcal{A}, \mathcal{V})$ *and all formulas* Φ_1, Φ_2 *the following holds:* $\Phi_1 \mathcal{S} \Phi_2 \equiv \Phi_1 \wedge \neg(\rho\, \neg(\Phi_1 \vee \Phi_2)[\neg\Phi_2])$ ◇

Definition 3. *We define some derived operators that are of particular use in medical image analysis:* $touch(\Phi_1, \Phi_2) \triangleq \Phi_1 \wedge \rho\ \Phi_2[\Phi_1]$; $grow(\Phi_1, \Phi_2) \triangleq \Phi_1 \vee touch(\Phi_2, \Phi_1)$; $flt(r, \Phi_1) \triangleq \mathcal{D}^{<r}(\mathcal{D}^{\geq r}\neg\Phi_1)$ •

The formula $touch(\Phi_1, \Phi_2)$ is satisfied by points that satisfy Φ_1 and that are on a path of points satisfying Φ_1 that reaches a point satisfying Φ_2. The formula $grow(\Phi_1, \Phi_2)$ is satisfied by points that satisfy Φ_1 and by points that satisfy Φ_2 which are on a path of points satisfying Φ_2 that reaches a point satisfying Φ_1. The formula $flt(r, \Phi_1)$ is satisfied by points that are at a distance of less than r from a point that is at least at distance r from points that do not satisfy Φ_1. This operator works as a filter; only contiguous areas satisfying Φ_1 that have a minimal diameter of at least $2r$ are preserved; these are also smoothened if they have an irregular shape (e.g. protrusions of less than the indicated distance).

Example 1. In Fig. 1, the top row shows four pictures using colours *blue* and *red*, interpreted as atomic propositions. Each picture in the bottom row shows in white the points that satisfy a given formula. In particular: Fig. 1e is $blue\, \mathcal{S}\, red$ of (a); Fig. 1f is touch($red, blue$) of (b); Fig. 1g is grow($red, blue$) of (c); Fig. 1h is $red\, \mathcal{S}\ (\mathcal{D}^{\leq 11} blue)$ of (d). For more details the reader is referred to [6].

2.2 Region Similarity via Statistical Cross-correlation

In the sequel, we provide some details on a logical operator, first defined in [3], that we use in the context of Texture Analysis (see for example [10,19,28,30]) for defining a notion of *statistical similarity* between image regions. The statistical distribution of an area Y of a black and white image is approximated

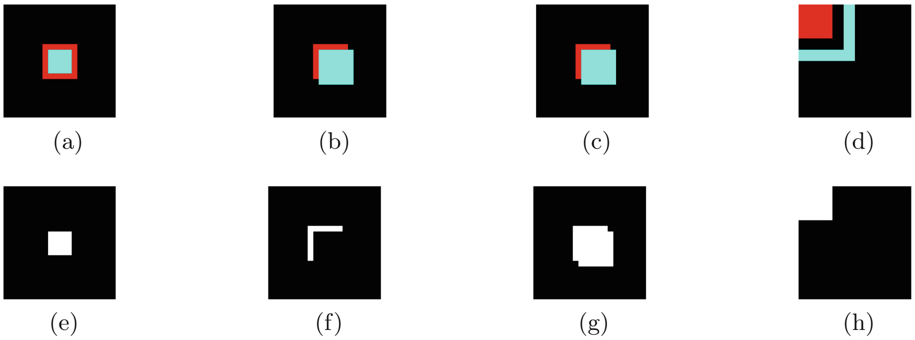

Fig. 1. Some examples of `ImgQL` operators (see Example 1). (Color figure online)

by the *histogram* of the grey levels of points (voxels) belonging to Y, limiting the representation to those levels laying in a certain interval $[m, M]$, the latter being split into k *bins*. In the case of images modelled as closure models, where each point may have several attributes, the histogram can be defined for different attributes. Given a closure model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{A}, \mathcal{V})$, define function $\mathcal{H} : A \times 2^X \times \mathbb{R} \times \mathbb{R} \times \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ such that for all $m < M$, $k > 0$ and $i \in \{1, \ldots, k\}$, $\mathcal{H}(a, Y, m, M, k)(i) = \Big|\{y \in Y \mid (i-1) \cdot \varDelta \leq \mathcal{A}(y, a) - m < i \cdot \varDelta\}\Big|$ where $\varDelta = \frac{M-m}{k}$. We call $\mathcal{H}(a, Y, m, M, k)$ the *histogram* of Y (for attribute a), with k bins and m, M min and max values respectively. The *mean* $\overline{h}$ of a histogram h with k *bins* is the quantity $\frac{1}{k}\sum_{i=1}^{k} h(i)$. The *cross correlation* between two histograms h_1, h_2 with the same number k of *bins* is defined as follows: $\mathbf{r}(h_1, h_2) = \frac{\sum_{i=1}^{k}\left(h_1(i)-\overline{h_1}\right)\left(h_2(i)-\overline{h_2}\right)}{\sqrt{\sum_{i=1}^{k}\left(h_1(i)-\overline{h_1}\right)^2}\sqrt{\sum_{i=1}^{k}\left(h_2(i)-\overline{h_2}\right)^2}}$. The value of $\mathbf{r}$ is *normalised* so that $-1 \leq \mathbf{r} \leq 1$; $\mathbf{r}(h_1, h_2) = 1$ indicates that h_1 and h_2 are *perfectly correlated* (that is, $h_1 = ah_2 + b$, with $a > 0$); $\mathbf{r}(h_1, h_2) = -1$ indicates *perfect anti-correlation* (that is, $h_1 = ah_2 + b$, with $a < 0$). On the other hand, $\mathbf{r}(h_1, h_2) = 0$ indicates no correlation.

We embed *statistical similarity* $\triangle\!\!\!\triangle_{\bowtie c}\left[\begin{smallmatrix} m & M & k \\ r & a & b \end{smallmatrix}\right]$ in the logic by adding it to the grammar defined by (1) and extending the definition of the satisfaction relation (Definition 2) with the following equation, for m, M, k as above:

$$\mathcal{M}, x \models \triangle\!\!\!\triangle_{\bowtie c}\left[\begin{smallmatrix} m & M & k \\ r & a & b \end{smallmatrix}\right]\varPhi \Leftrightarrow \mathbf{r}(h_a, h_b) \bowtie c$$

where $h_a = \mathcal{H}(a, S(x, r), m, M, k)$, $h_b = \mathcal{H}(b, \{y \mid \mathcal{M}, y \models \varPhi\}, m, M, k)$, c is a constant in $[-1, 1]$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $S(x, r) = \{y \in X \mid d(x, y) \leq r\}$ is the *sphere* of radius r centred in x. Note that, differently from `topochecker` that was used in [3], in `VoxLogicA`, for efficiency reasons, $S(x, r)$ is actually the *hypercube* with edge size $2r$, which, for anisotropic images, becomes a hyperrectangle. So $\triangle\!\!\!\triangle_{\bowtie c}\left[\begin{smallmatrix} m & M & k \\ r & a & b \end{smallmatrix}\right]\varPhi$ compares the region of the image constituted by the sphere (hypercube) of radius r centred in x against the region characterised by $\varPhi$. The comparison is based on the cross correlation of the histograms of the chosen attributes of (the points of) the two regions, namely a and b and both histograms share the same range ($[m, M]$) and the same bins ($[1, k]$). In summary, the operator allows to check *to which extent* the *sphere (hypercube) around the point of interest* is *statistically similar* to a given region (specified by) Φ.

3 The Tool `VoxLogicA`

`VoxLogicA` is a framework for image analysis, that embeds the logic ImgQL into a user-oriented expression language to manipulate images. More precisely, the `VoxLogicA` type system distinguishes between *boolean-valued* images, that can be arguments or results of the application of ImgQL operators, and *number-valued* images, resulting from imaging primitives. Underlying such expression language is a *global model checker*, that is, the set of points satisfying a logic formula is computed at once; this is done implicitly when an expression corresponding to a logic formula is saved to an image. Functionality-wise, `VoxLogicA` specialises `topochecker` to the case of spatial analysis of *multi-dimensional images*. It interprets a specification written in the ImgQL language, using a set of multi-dimensional images[6] as models of the spatial logic, and produces as output a set of multi-dimensional images representing the valuation of user-specified expressions. For logical operators, such images are Boolean-valued, that is, *regions of interest* in medical imaging terminology, which may be loaded as *overlays* in medical image viewers. Non-logical operators may generate number-valued images. `VoxLogicA` augments ImgQL with file loading and saving primitives, and a set of additional commodity operators, specifically aimed at image analysis, that is destined to grow along with future developments of the tool. The main execution modality of `VoxLogicA` is *batch execution*. A (currently experimental) *graphical user interface* is under development.

Implementation-wise, the tool achieves a two-orders-of-magnitude speedup with respect to `topochecker`. Such speedup has permitted the rapid development of a novel procedure for automatic segmentation of *glioblastoma* that, besides being competitive with respect to the state-of-the-art in the field (see Sect. 4), is also easily *replicable* and *explainable* to humans, and therefore amenable of improvement by the community of medical imaging practitioners.

3.1 Functionality

We provide an overview of the tool functionality, starting from its syntax. For space reasons, we omit details on parsing rules (delegated to the tool documentation). In the following, `f, x1,..., xN, x` are identifiers, `"s"` is a string, and `e1, ..., eN, e` are expressions (to be detailed later). A `VoxLogicA` specification consists of a text file containing a sequence of **commands** (see Specification 1 in Sect. 4 as an example). Five commands are currently implemented:

[6] Besides common bitmap formats, the model loader of `VoxLogicA` currently supports the NIfTI (Neuro-imaging Informatics Technology Initiative) format (https://nifti.nimh.nih.gov/, version 1 and 2). 3D MR-FLAIR images in this format very often have a slice size of 256 by 256 pixels, multiplied by 20 to 30 slices.

- `let f(x1,...,xN) = e` is used for *function declaration*, also in the form `let f = e` *(constant declaration)*, and with special syntactic provisions to define *infix* operators. After execution of the command, name `f` is bound to a function or constant that evaluates to `e` with the appropriate substitutions of parameters;
- `load x = "s"` loads an image from file `"s"` and binds it to `x` for subsequent usage;
- `save "s" e` stores the image resulting from evaluation of expression `e` to file `"s"`;
- `print "s" e` prints to the log the string `s` followed by the numeric, or boolean, result of computing `e`;
- `import "s"` imports a library of declarations from file `"s"`; subsequent import declarations for the *same* file are not processed; furthermore, such imported files can only contain `let` or `import` commands.

`VoxLogicA` comes equipped with a set of built-in functions, such as arithmetic operators, logic primitives as described in Sect. 2, and imaging operators, for instance for computing the gray-scale intensity of a colour image, or its colour components, or the percentiles of its values (see Sect. 4.1). An exhaustive list of the available built-ins is provided in the user manual[7]. Furthermore, a "standard library" is provided containing short-hands for commonly used functions, and for derived operators. An **expression** may be a numeric literal (no distinction is made between floating point and integer constants), an identifier (e.g. `x`), a function application (e.g. `f(x1,x2)`), an infix operator application (e.g. `x1 + x2`), or a parenthesized (sub-)expression (e.g. `(x1 + x2)`).

The language features **strong dynamic typing**, that is, types of expressions are unambiguously checked and errors are precisely reported, but such checks are only performed at "run time", that is, when evaluating closed-form expressions with no free variables. The type system has currently been kept lightweight (the only typing rules regard constants and function application), in order to leave the design space open to future improvements. For instance, a planned development is function and operator *overloading*, as well as some form of static typing not interfering with the usability of the tool.

However, it is *not* the case that a type error may waste a long-running analysis. Type checking occurs after loading and parsing, but before analysis is run. Actual program execution after parsing is divided into two phases. First (usually, in a negligible amount of time), all the "save" and "print" instructions are examined to determine what expressions actually *need* to be computed; in this phase, name binding is resolved, all constant and function applications are substituted with closed expressions, types are checked and the environment binding expressions to names is discarded. Finally, the set of closed expressions to be evaluated is transformed into a set of tasks to be executed, possibly in parallel, and dependencies among them. After this phase, no further syntax processing or name resolution are needed, and it is guaranteed that the program is free from type errors. The second phase simply runs each task – in an order compliant with dependencies – parallelising execution on multiple CPU cores.

Each built-in logical operator has an associated type of its input parameters and output result. The available types are inductively defined as `Number`, `Bool`,

[7] See https://github.com/vincenzoml/VoxLogicA.

`String`, `Model`, and `Valuation(t)`, where `t` is in turn a type. The type `Model` is the type assigned to `x` in `load x = "f"`; operations such as the extraction of RGB components take this type as input, and return as output the only parametric type: `Valuation(t)`, which is the type of a multi-dimensional image in which each voxel contains a value of type `t`. For instance, the red component of a loaded model has type `Valuation(Number)`, whereas the result of evaluating a logic formula has type `Valuation(Bool)`[8].

An important aspect of the execution semantics of `VoxLogicA` specifications is *memoization*, constituting the core of its execution engine, and used to achieve maximal sharing of subformulas. In `VoxLogicA`, no expression is ever computed twice, freeing the user from worrying about how many times a given function is called, and making execution of complex macros and logical operators feasible.

3.2 Implementation Details

`VoxLogicA` is implemented in the functional, object-oriented programming language `FSharp`, using the `.NET Core` implementation of the `.NET` specification[9]. This permits a single code base with minimal environment-dependent setup to be cross-compiled and deployed as a standalone executable, for the major desktop operating systems, namely *Linux*, *macOS*, and *Windows*. Despite `.NET` code is compiled for an intermediate machine, this does not mean that efficiency of `VoxLogicA` is somehow "non-native". There are quite a number of measures in place to maximise efficiency. First and foremost, the execution time is heavily dominated by the time spent in native libraries (more details below), and `VoxLogicA` acts as a higher-level, declarative front-end for such libraries, adding a logical language, memoization, parallel execution, and abstraction from a plethora of technical details that a state-of-the-art imaging library necessarily exposes. In our experiments, parsing, memoization, and preparation of the tasks to be run may take a fraction of a second; the rest of the execution time (usually, several seconds, unless the analysis is extremely simple) is spent in *foreign function calls*. The major performance boosters in `VoxLogicA` are: a state-of-the-art computational imaging library (`ITK`); the optimised implementation of the *may reach* operator; a new algorithm for statistical cross-correlation; an efficient memoizing execution engine; parallel evaluation of independent tasks, exploiting modern multi-core CPUs. Moreover, special care has been put in making all performance-critical loops *allocationless*. All used memory along the loops is pre-allocated, avoiding the risk to trigger garbage collection during computation. We will address each of them briefly in the following.

ITK Library. `VoxLogicA` uses the state-of-the-art imaging library `ITK`, via the `SimpleITK` glue library[10]. Most of the operators of `VoxLogicA` are implemented

[8] Although such type system would permit "odd" types such as `Valuation(Model)`, there is no way to construct them; in the future this may change when appropriate.
[9] See https://fsharp.org and https://dotnet.github.io.
[10] See https://itk.org and http://www.simpleitk.org.

directly by a library call. Notably, this includes the *Maurer distance transform*, used to efficently implement the distance operators of ImgQL.

Novel Algorithms. The two most relevant operators that do not have a direct implementation in ITK are `mayReach` and `crossCorrelation`, implementing, respectively, the logical operator ρ, and statistical comparison described in Sect. 2.2. The computation of the voxels satisfying $\rho\ \phi_1[\phi_2]$ can be implemented either using the (classical, in computer graphics) *flood-fill* primitive, or by exploiting the *connected components* of ϕ_2 as a reachability primitive; both solutions are available in `SimpleITK`. In our experiments, connected components perform better using this library from `FSharp`, for large input seeds. Several critical logical connectives (e.g. *surrounded* and *touch*), are defined in terms of `mayReach`. Therefore, an optimised algorithm for `mayReach` is a key performance improvement. The `crossCorrelation` operation is resource-intensive, as it uses the histogram of a multi-dimensional hyperrectangle at each voxel. Pre-computation methods such as the *integral histogram* [33], would not yield the expected benefits, because cross-correlation is called only few times on the same image. In this work, we designed a parallel algorithm exploiting *additivity* of histograms. Given two sets of values P_1, P_2, let h_1, h_2 be their respective histograms, and let h'_1, h'_2 be the histograms of $P_1 \backslash P_2$ and $P_2 \backslash P_1$. For i a bin, we have $h_2(i) = h_1(i) - h'_1(i) + h'_2(i)$. This property leads to a particularly efficient algorithm when P_1 and P_2 are two hyperrectangles centred over adjacent voxels, as $P_1 \backslash P_2$ and $P_2 \backslash P_1$ are *hyperfaces*, having one dimension less than hyperrectangles. Our algorithm divides the image into as many partitions as the number of available processors, and then computes a *Hamiltonian path* for each partition, passing by each of its voxels exactly once. All partitions are visited in parallel, in the order imposed by such Hamiltonian paths; the histogram is computed incrementally as described above; finally cross-correlation is also computed and stored in the resulting image. The *asymptotic algorithmic complexity* of the implementation of ImgQL primitives in `VoxLogicA` is linear in the number of voxels, with the exception of `crossCorrelation`, which, by the above explanation, has complexity $O(k \cdot n)$, where n is the number of voxels, and k is the size of the largest hyperface of the considered hypercube.

Memoizing Execution Semantics. Sub-expressions in `VoxLogicA` are *by construction* identified up-to syntactic equality and assigned a number, representing a unique identifier (UID). UIDs start from 0 and are contiguous, therefore admitting an array of all existing sub-formulas to be used to pre-computed valuations of expressions without further hashing.

3.3 Design and Data Structures

The design of `VoxLogicA` defines three implementation layers. The *core* execution engine implements the concurrent, memoizing semantics of the tool. The *interpreter* is responsible for translating source code into core library invocations. These two layers only include some basic arithmetic and boolean primitives.

Operators can be added by inheriting from the abstract base class `Model`. The third implementation layer is the instantiation of the core layer to define operators from ImgQL, and loading and saving of graphical models, using the `ITK` library. We provide some more detail on the design of the core layer, which is the most critical part of `VoxLogicA`. At the time of writing, the core consists of just 350 lines of `FSharp` code, that has been carefully engineered not only for performance, but also for ease of maintenance and future extensions.

The essential **classes** are `ModelChecker`, `FormulaFactory`, `Formula`, and `Operator`, of which `Constant` is a subclass. Class `Operator` describes the available operators and their evaluation method. Class `Formula` is a symbolic representation of a syntactic sub-expression. Each instance of `Formula` has a unique numeric id (UID), an instance of `Operator`, and (inductively) a list of `Formula` instances, denoting its arguments. The UID of a formula is determined by the operator name (which is unique across the application), and the list of parameter UIDs. Therefore, by construction, it is not possible to build two different instances of `Formula` that are syntactically equal. UIDs are contiguous and start from 0. By this, all created formulas can be inserted into an array. Furthermore, UIDs are allocated in such a way that the natural number order is a topological sort of the dependency graph between subformulas (that is, if f_1 is a parameter of f_2, the UID of f_1 is greater than the UID of f_2). This is exploited in class `ModelChecker`; internally, the class uses an array to store the results of evaluating each `Formula` instance, implementing memoization. The class `ModelChecker` turns each formula into a task to be executed. Whenever a formula with UID i is a parameter of the formula with UID j, a dependency is noted between the associated tasks. The high-level, lightweight concurrent programming library `Hopac`[11] and its abstractions are used to evaluate the resulting task graph, in order to maximise CPU usage on multi-core machines.

4 Experimental Evaluation

The performance of `VoxLogicA` has been evaluated on the Brain Tumor Image Segmentation Benchmark (BraTS) of 2017 [2,31] containing 210 multi contrast MRI scans of high grade glioma patients that have been obtained from multiple institutions and were acquired with different clinical protocols and various scanners. All the imaging data sets provided by BraTS 2017 have been segmented manually and approved by experienced neuro-radiologists. In our evaluation we used the T2 Fluid Attenuated Inversion Recovery (FLAIR) type of scans, which is one of the four provided modalities in the benchmark. Use of other modalities is planned for future work. For training, the numeric parameters of the `VoxLogicA` specification presented in Sect. 4.1 were manually calibrated against a subset of 20 cases. Validation of the method was conducted as follows. A priori, 17 of the 210 cases can be excluded because the current procedure is not suitable for these images. This is because of the presence of multi-focal tumours (different tumours in different areas of the brain), or due to clearly distinguishable artifacts in the

[11] See https://github.com/Hopac/Hopac.

FLAIR acquisition, or because the hyperintense area is too large and clearly not significant (possibly by incorrect acquisition). Such cases require further investigation. For instance, the current procedure may be improved to identify specific types of artefacts, whereas multi-modal analysis can be used to complement the information provided by the FLAIR image in cases where FLAIR hyperintensity is not informative enough. In Sect. 4.2, we present the results both for the full dataset (210 cases), and for the subset without these problematic cases (193 cases). We considered both the *gross tumour volume* (GTV), corresponding to what can actually be seen on an image, and the *clinical target volume* (CTV) which is an extension of the GTV. For glioblastomas this margin is a 2–2.5 cm isotropic expansion of the GTV volume within the brain.

4.1 ImgQL Segmentation Procedure

Specification 1 shows the tumour segmentation procedure that we used for the evaluation[12]. The syntax is that of `VoxLogicA`, namely: `|`,`&`,`!` are boolean *or*, *and*, *not*; `distlt(c,phi)` is the set $\{y \mid \mathcal{M}, y \models \mathcal{D}^{<c}\texttt{phi}\}$ (similarly, `distgeq`; distances are in millimiters); `crossCorrelation(r,a,b,phi,m,M,k)` yields a cross-correlation coefficient for each voxel, to which a predicate c may be applied to obtain the *statistical similarity* function of Sect. 2.2; the `>` operator performs thresholding of an image; `border` is true on voxels that lay at the border of the image. Operator `percentiles(img,mask)`, where `img` is a number-valued image, and `mask` is boolean-valued, considers the points identified by `mask`, and assigns to each such point x the fraction of points that have an intensity below that of x in `img`. Other operators are explained in Definition 3 (see also Fig. 1). Figure 2 shows the intermediate phases of the procedure, for axial view of one specific 2D slice of an example 3D MRI scan of the BraTS 2017 data set.

We briefly discuss the specification (see [6] for more details). Lines 1–8 merely define utility functions and load the image, calling it `flair`. Lines 9–10 define the `background` as all voxels in the area of intensity less than 0.1 that touches the border of the image, and the `brain` as the complement of the background. The application of `percentiles` in line 11 assigns to each point of the brain the percentile rank of its intensity among those that are part of `brain`. Based on these percentiles, hyper-intense and very-intense points are identified that satisfy hI and vI, respectively (lines 12–13). Hyper-intense points have a very high likelihood to belong to tumour tissue; very-high intensity points are likely to belong to the tumour as well, or to the oedema that is usually surrounding the tumour. However, not all hyper-intense and very-intense points are part of a tumour. The idea is to identify the actual tumour using further spatial information. In lines 14–15 the hyper-intense and very-intense points are filtered, thus removing noise, and considering only areas of a certain relevant size.

[12] Note that, although the procedure is loosely inspired by the one in [3], there are major differences, partly due to a different method for identification of hyperintensities (using percentiles), and partly since the task in this work is simpler, as we only identify the CTV and GTV (avoiding, for instance, to label the oedema).

The points that satisfy `hyperIntense` and `veryIntense` are shown in red in Fig. 2a and in Fig. 2b, respectively. In line 16 the areas of hyper-intense points are extended via the `grow` operator, with those areas that are very intense (possibly belonging to the oedema), and in turn touch the hyper-intense areas. The points that satisfy `growTum` are shown in red in Fig. 2c. In line 17 the previously-defined (line 8) similarity operator is used to assign to all voxels a texture-similarity score with respect to `growTum`. In line 18 this operator is used to find those voxels that have a high cross correlation coefficient and thus are likely part of the tumour. The result is shown in Fig. 2d. Finally (line 19), the voxels that are identified as part of the whole tumour are those that satisfy `growTum` extended with those that are statistically similar to it via the `grow` operator. Points that satisfy `tumFinal` are shown in red in Fig. 2e and points identified by manual segmentation are shown for comparison in blue in the same figure (overlapping areas are purple).

ImgQL **Specification 1:** Full specification of tumour segmentation

```
 1 import "stdlib.imgql"
 2 let grow(a,b) = (a | touch(b,a))
 3 let flt(r,a) = distlt(r,distgeq(r,!a))
 4 load imgFLAIR = "Brats17_2013_2_1_flair.nii.gz"
 5 load imgManualSeg = "Brats17_2013_2_1_seg.nii.gz"
 6 let manualContouring = intensity(imgManualSeg) > 0
 7 let flair = intensity(imgFLAIR)
 8 let similarFLAIRTo(a) =
   crossCorrelation(5,flair,flair,a,min(flair),max(flair),100)
 9 let background = touch(flair < 0.1,border)
10 let brain = !background
11 let pflair = percentiles(flair,brain)
12 let hI = pflair > 0.95
13 let vI = pflair > 0.86
14 let hyperIntense = flt(5.0,hI)
15 let veryIntense = flt(2.0,vI)
16 let growTum = grow(hyperIntense,veryIntense)
17 let tumSim = similarFLAIRTo(growTum)
18 let tumStatCC = flt(2.0,(tumSim > 0.6))
19 let tumFinal= grow(growTum,tumStatCC)
20 save "output_Brats17_2013_2_1/complete-FLAIR_FL-seg.nii" tumFinal
```

Interesting aspects of the ImgQL specification are its relative simplicity and abstraction level, fitting that of neuro-radiologists, its explainability, its time-efficient verification, admitting a rapid development cycle, and its independence of normalisation procedures through the use of percentiles rather than absolute values for the intensity of voxels.

4.2 Validation Results

Results of tumour segmentation are evaluated based on a number of indexes commonly used to compare the quality of different techniques (see [31]). These indexes are based on the true positive (TP) voxels (voxels that are identified as part of a tumour in both manual and `VoxLogicA` segmentation), true negatives (TN) voxels (those that are *not* identified as part of a tumour in both manual and `VoxLogicA` segmentation), false positives (FP) voxels (those identified as part of a tumour by `VoxLogicA` but not by manual segmentation) and false negatives (FN) voxels (those identified as part of a tumour by manual segmentation but not by `VoxLogicA`). Based on these four types the following indexes are defined: *sensitivity*: $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$; *specificity*: $\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$; *Dice*: $2*\mathrm{TP}/(2*\mathrm{TP}+\mathrm{FN}+\mathrm{FP})$. Sensitivity measures the fraction of voxels that are correctly identified as part of a tumour. Specificity measures the fraction of voxels that are correctly identified as *not* being part of a tumour. The Dice similarity coefficient is used to provide a measure of the similarity of two segmentations. Table 1 shows the mean values of the above indexes both for GTV and CTV volumes for Specification 1 applied to the BraTS 2017 training phase collection. The top-scoring methods of the BraTS 2017 Challenge [35] can be considered a good sample of the state-of-the-art in this domain. Among those, in order to collect significant statistics, we selected the 18 techniques that have been applied to at least 100 cases of the dataset. The *median* and *range of values* of the sensitivity, specificity and Dice indexes for the GTV segmentation of the whole tumour are, respectively, 0.88 (ranging from 0.55 to 0.97), 0.99 (0.98 to 0.999) and 0.88 (0.64 to 0.96). The 3D images used in this experiment have size $240 \times 240 \times 155$ (about 9 million voxels). The evaluation of each case study takes about 10 s on a desktop computer equipped with an Intel Core I7 7700 processor (with 8 cores) and 16 GB of RAM.

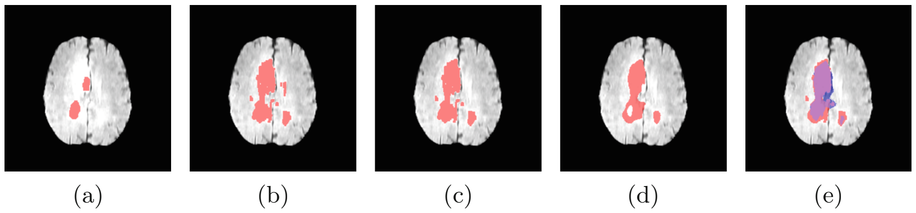

Fig. 2. Tumour segmentation of image `Brats17_2013_2_1`, FLAIR, axial 2D slice at X = 155, Y = 117 and Z = 97. (a) hyperIntense (b) veryIntense (c) growTum (d) tumStatCC (e) tumFinal (red) and manual (blue, overlapping area is purple). (Color figure online)

Table 1. VoxLogicA evaluation on the BraTS 2017 benchmark.

	Sensitivity (193 cases)	Specificity (193 cases)	Dice (193 cases)	Sensitivity (210 cases)	Specificity (210 cases)	Dice (210 cases)
GTV	0.89(0.10)	1.0(0.00)	0.85(0.10)	0.86(0.16)	1.0(0.0)	0.81(0.18)
CTV	0.95(0.07)	0.99(0.01)	0.90(0.09)	0.93(0.14)	0.99(0.2)	0.87(0.15)

4.3 Comparison with `topochecker`

The evaluation of `VoxLogicA` that we presented in this section uses features that are present in `VoxLogicA`, but not in `topochecker`. On the other hand, the example specification in [3], and its variant aimed at 3D images, are quite similar to the one we presented, and can be readily used to compare the performance of `VoxLogicA` and `topochecker`. The specifications consist of two human-authored text files of about 30 lines each. The specifications were run on a desktop computer equipped with an Intel Core I7 7700 processor (with 8 cores) and 16 GB of RAM. In the 2D case (image size: 512×512), `topochecker` took 52 s to complete the analysis, whereas `VoxLogicA` took 750 ms. In the 3D case (image size: $512 \times 512 \times 24$), `topochecker` took about 30 min, whereas `VoxLogicA` took 15 s. As we mentioned before, this huge improvement is due to the combination of a specialised imaging library, new algorithms (e.g., for statistical similarity of regions), parallel execution and other optimisations. More details could be obtained by designing a specialised set of benchmarks, where some of which can also be run using `topochecker`; however, for the purposes of the current paper, the performance difference is so large that we do not deem such detailed comparison necessary.

5 Conclusions and Future Work

We presented `VoxLogicA`, a spatial model checker designed and optimised for the analysis of multi-dimensional digital images. The tool has been successfully evaluated on 193 cases of an international brain tumour 3D MRI segmentation benchmark. The obtained results are well-positioned w.r.t. the performance of state-of-the-art segmentation techniques, both efficiency-wise and accuracy-wise. Future research work based on the tool will focus on further benchmarking (e.g. various other types of tumours and tumour tissue such as necrotic and non-enhancing parts), and clinical application. On the development side, planned future work includes a graphical (web) interface for interactive parameter calibration (for that, execution times will need to be further improved, possibly employing *GPU computing*); improvements in the type-system (e.g. *operator overloading*); turning the core design layer into a reusable library available for other projects. Finally, the (currently small, albeit useful) library of logical and imaging-related primitives available will be enhanced, based on input from case studies. Calibration of the numerical parameters of our Glioblastoma segmentation was done manually. Future work aims at exploring different possibilities for human-computer interaction in designing such procedures (e.g. via ad-hoc graphical interfaces), to improve user friendliness for domain experts. Experimentation in combining *machine-learning* methods with the logic-based approach of `VoxLogicA` are also worth being explored in this respect.

References

1. Aiello, M., Pratt-Hartmann, I., van Benthem, J.: Handbook of Spatial Logics. Springer, Dordrecht (2007). https://doi.org/10.1007/978-1-4020-5587-4
2. Bakas, S., et al.: Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. Sci. Data **4** (2017). https://doi.org/10.1038/sdata.2017.117. Accessed 05 Sept 2017
3. Banci Buonamici, F., Belmonte, G., Ciancia, V., et al.: Int. J. Softw. Tools Technol. Transfer (2019). https://doi.org/10.1007/s10009-019-00511-9
4. Bartocci, E., Bortolussi, L., Loreti, M., Nenzi, L.: Monitoring mobile and spatially distributed cyber-physical systems. In: Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2017, pp. 146–155. ACM, New York (2017). http://doi.acm.org/10.1145/3127041.3127050
5. Bartocci, E., Gol, E.A., Haghighi, I., Belta, C.: A formal methods approach to pattern recognition and synthesis in reaction diffusion networks. IEEE Trans. Control Netw. Syst. 1 (2016). https://doi.org/10.1109/tcns.2016.2609138
6. Belmonte, G., Ciancia, V., Latella, D., Massink, M.: VoxLogicA: a spatial model checker for declarative image analysis (Extended Version). ArXiv e-prints, November 2018. https://arxiv.org/abs/1811.05677
7. Belmonte, G., et al.: A topological method for automatic segmentation of glioblastoma in MRI flair for radiotherapy. Magn. Reson. Mater. Phys. Biol. Med. **30**(S1), 437 (2017). https://doi.org/10.1007/s10334-017-0634-z. In ESMRMB 2017, 34th annual scientific meeting
8. Belmonte, G., Ciancia, V., Latella, D., Massink, M.: From collective adaptive systems to human centric computation and back: spatial model checking for medical imaging. In: ter Beek, M.H., Loreti, M. (eds.) Proceedings of the Workshop on FORmal Methods for the Quantitative Evaluation of Collective Adaptive SysTems, FORECAST@STAF 2016, Vienna, Austria, 8 July 2016. EPTCS, vol. 217, pp. 81–92 (2016). https://doi.org/10.4204/EPTCS.217.10
9. van Benthem, J., Bezhanishvili, G.: Modal logics of space. In: Aiello, M., Pratt-Hartmann, I., Van Benthem, J. (eds.) Handbook of Spatial Logics, pp. 217–298. Springer, Dordrecht (2007). https://doi.org/10.1007/978-1-4020-5587-4_5
10. Castellano, G., Bonilha, L., Li, L., Cendes, F.: Texture analysis of medical images. Clin. Radiol. **59**(12), 1061–1069 (2004)
11. Ciancia, V., Gilmore, S., Latella, D., Loreti, M., Massink, M.: Data verification for collective adaptive systems: spatial model-checking of vehicle location data. In: Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW, pp. 32–37. IEEE Computer Society (2014)
12. Ciancia, V., Grilletti, G., Latella, D., Loreti, M., Massink, M.: An experimental spatio-temporal model checker. In: Bianculli, D., Calinescu, R., Rumpe, B. (eds.) SEFM 2015. LNCS, vol. 9509, pp. 297–311. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-49224-6_24

13. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and verifying properties of space. In: Diaz, J., Lanese, I., Sangiorgi, D. (eds.) TCS 2014. LNCS, vol. 8705, pp. 222–235. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44602-7_18
14. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Model checking spatial logics for closure spaces. Log. Methods Comput. Sci. **12**(4), October 2016. http://lmcs.episciences.org/2067
15. Ciancia, V., Latella, D., Massink, M., Pakauskas, R.: Exploring spatio-temporal properties of bike-sharing systems. In: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops, pp. 74–79. IEEE Computer Society (2015)
16. Ciancia, V., Gilmore, S., Grilletti, G., Latella, D., Loreti, M., Massink, M.: Spatio-temporal model checking of vehicular movement in public transport systems. Int. J. Softw. Tools Technol. Transfer (2018). https://doi.org/10.1007/s10009-018-0483-8
17. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Spatial logic and spatial model checking for closure spaces. In: Bernardo, M., De Nicola, R., Hillston, J. (eds.) SFM 2016. LNCS, vol. 9700, pp. 156–201. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34096-8_6
18. Ciancia, V., Latella, D., Massink, M., Paškauskas, R., Vandin, A.: A tool-chain for statistical spatio-temporal model checking of bike sharing systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016, Part I. LNCS, vol. 9952, pp. 657–673. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47166-2_46
19. Davnall, F., et al.: Assessment of tumor heterogeneity: an emerging imaging tool for clinical practice? Insights Imaging **3**(6), 573–589 (2012)
20. Despotović, I., Goossens, B., Philips, W.: MRI segmentation of the human brain: challenges, methods, and applications. Comput. Math. Methods Med. **2015**, 1–23 (2015). https://doi.org/10.1155/2015/450341
21. Dupont, C., Betrouni, N., Reyns, N., Vermandel, M.: On image segmentation methods applied to glioblastoma: state of art and new trends. IRBM **37**(3), 131–143 (2016). https://doi.org/10.1016/j.irbm.2015.12.004
22. Fyllingen, E.H., Stensjøen, A.L., Berntsen, E.M., Solheim, O., Reinertsen, I.: Glioblastoma segmentation: comparison of three different software packages. PLOS ONE **11**(10), e0164891 (2016). https://doi.org/10.1371/journal.pone.0164891
23. Galton, A.: The mereotopology of discrete space. In: Freksa, C., Mark, D.M. (eds.) COSIT 1999. LNCS, vol. 1661, pp. 251–266. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48384-5_17
24. Galton, A.: A generalized topological view of motion in discrete space. Theor. Comput. Sci. **305**(1–3), 111–134 (2003). https://doi.org/10.1016/S0304-3975(02)00701-6
25. Galton, A.: Discrete mereotopology. In: Calosi, C., Graziani, P. (eds.) Mereology and the Sciences: Parts and Wholes in the Contemporary Scientific Context, pp. 293–321. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05356-1_11
26. Grosu, R., Smolka, S., Corradini, F., Wasilewska, A., Entcheva, E., Bartocci, E.: Learning and detecting emergent behavior in networks of cardiac myocytes. Commun. ACM **52**(3), 97–105 (2009)
27. Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Grosu, R., Belta, C.: Spatel: a novel spatial-temporal logic and its applications to networked systems. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC 2015, pp. 189–198. ACM, New York (2015)

28. Kassner, A., Thornhill, R.E.: Texture analysis: a review of neurologic MR imaging applications. Am. J. Neuroradiol. **31**(5), 809–816 (2010)
29. Lemieux, L., Hagemann, G., Krakow, K., Woermann, F.: Fast, accurate, and reproducible automatic segmentation of the brain in t1-weighted volume mri data. Magn. Reson. Med. **42**(1), 127–135 (1999)
30. Lopes, R., et al.: Prostate cancer characterization on MR images using fractal features. Med. Phys. **38**(1), 83 (2011)
31. Menze, B.H., et al.: The multimodal brain tumor image segmentation benchmark (brats). IEEE Trans. Med. Imaging **34**(10), 1993–2024 (2015)
32. Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties. In: Bartocci, E., Majumdar, R. (eds.) RV 2015. LNCS, vol. 9333, pp. 21–37. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23820-3_2
33. Porikli, F.M.: Integral histogram: a fast way to extract histograms in Cartesian spaces. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 829–836 (2005)
34. Simi, V., Joseph, J.: Segmentation of glioblastoma multiforme from MR images-a comprehensive review. Egypt. J. Radiol. Nucl. Med. **46**(4), 1105–1110 (2015). https://doi.org/10.1016/j.ejrnm.2015.08.001
35. Spyridon (Spyros) Bakas, et al. (Ed.): 2017 international MICCAI BraTS Challenge: Pre-conference Proceedings, September 2017. https://www.cbica.upenn.edu/sbia/Spyridon.Bakas/MICCAI_BraTS/MICCAI_BraTS_2017_proceedings_shortPapers.pdf
36. Tsigkanos, C., Kehrer, T., Ghezzi, C.: Modeling and verification of evolving cyber-physical spaces. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, pp. 38–48. ACM, New York (2017). http://doi.acm.org/10.1145/3106237.3106299
37. Zhu, Y., et al.: Semi-automatic segmentation software for quantitative clinical brain glioblastoma evaluation. Acad. Radiol. **19**(8), 977–985 (2012). https://doi.org/10.1016/j.acra.2012.03.026

Author Index