CrossMark

Multi-objective Parameter Auto-tuning for Tissue Image Segmentation Workflows

Luis F. R. Taveira [1] · Tahsin Kurc [2,3] · Alba C. M. A. Melo [1] · Jun Kong [4,5,6] · Erich Bremer [2] · Joel H. Saltz [2] · George Teodoro [1,4]

Abstract

We propose a software platform that integrates methods and tools for multi-objective parameter auto-tuning in tissue image segmentation workflows. The goal of our work is to provide an approach for improving the accuracy of nucleus/cell segmentation pipelines by tuning their input parameters. The shape, size, and texture features of nuclei in tissue are important biomarkers for disease prognosis, and accurate computation of these features depends on accurate delineation of boundaries of nuclei. Input parameters in many nucleus segmentation workflows affect segmentation accuracy and have to be tuned for optimal performance. This is a time-consuming and computationally expensive process; automating this step facilitates more robust image segmentation workflows and enables more efficient application of image analysis in large image datasets. Our software platform adjusts the parameters of a nuclear segmentation algorithm to maximize the quality of image segmentation results while minimizing the execution time. It implements several optimization methods to search the parameter space efficiently. In addition, the methodology is developed to execute on high-performance computing systems to reduce the execution time of the parameter tuning phase. These capabilities are packaged in a Docker container for easy deployment and can be used through a friendly interface extension in 3D Slicer. Our results using three real-world image segmentation workflows demonstrate that the proposed solution is able to (1) search a small fraction (about 100 points) of the parameter space, which contains billions to trillions of points, and improve the quality of segmentation output by $\times 1.20$, $\times 1.29$, and $\times 1.29$, on average; (2) decrease the execution time of a segmentation workflow by up to $11.79\times$ while improving output quality; and (3) effectively use parallel systems to accelerate parameter tuning and segmentation phases.

Keywords Digital pathology · Computer-assisted image analysis · Cell morphology · Cancer · Parameter auto-tuning

Introduction

We propose and experimentally evaluate a software platform that integrates a suite of methods and tools to enable automatic parameter tuning in analysis algorithms that segment nuclei in digitized images of tissue specimens fixed on glass slides, also called Whole Slide Tissue Images (WSIs). Microscopic examination of whole slide tissue specimens by pathologists has long been considered a de facto standard for disease diagnosis and prognosis. Diseased tissue shows changes in tissue morphology, which are indicators of disease onset and progress and provide rich information with which to study disease

✉ George Teodoro
teodoro@unb.br

1 Department of Computer Science, University of Brasília, Brasília, Brazil

2 Department of Biomedical Informatics, Stony Brook University, Stony Brook, NY, USA

3 Scientific Data Group, Oak Ridge National Laboratory, Oak Ridge, TN, USA

4 Department of Biomedical Informatics, Emory University School of Medicine, Atlanta, GA, USA

5 Department of Biomedical Engineering, Emory - Georgia Institute of Technology, Atlanta, GA, USA

6 Department of Mathematics and Statistics, Georgia State University, Atlanta, GA, USA

biology at the subcellular level. Manual examination of tissue specimens, however, has had limited use in biomedical research because it is a labor-intensive and time-consuming process. Advances in digital microscopy scanners have made it possible to capture tissue images at very high resolutions; state-of-the-art scanners can capture images at $100,000 \times 100,000$ square pixel resolutions and can automatically scan hundreds of tissue slides rapidly, thanks to sophisticated auto-focusing mechanisms. Whole slide tissue images enable quantitative and reproducible analysis of tissue morphology—the importance of improving precision and reducing inter-observer variability in pathology studies is well recognized [1–14]. In addition, the Food and Drug Administration (FDA) has recently approved the use of digitized tissue images for diagnostic purposes, both recognizing the value of whole slide tissue imaging in clinical settings and paving the way for routine use of tissue imaging, which we expect will lead to significant increases in the number and volume of WSI datasets for imaging studies. A number of projects have developed tissue image analysis methods [15–21] and shown that quantitative image characterizations from pathology images can be used to predict outcome and treatment responses [16, 22–26]. Nevertheless, development of robust and efficient computerized image analysis workflows to reliably extract imaging features from WSIs remains an open challenge.

Our work targets nucleus segmentation workflows as key part of this open challenge. Segmentation of nuclei is one of the most common steps in WSI analysis, because a disease often manifests itself as changes to the properties, such as shape and texture, and organization of nuclei in tissue. A nuclear segmentation workflow detects nuclei and delineates their boundaries. Shape, size, intensity, and texture features are computed for each segmented nucleus based on the characteristics of the image (i.e., tissue) within the boundary of the said nucleus. These features can then be used to classify images and patients in downstream analyses [27–29]. Thus, the segmentation quality in an image may significantly impact the accuracy and robustness of results obtained from image analysis studies. The inherent complexity of tissue makes it a challenging task to develop accurate and reliable segmentation algorithms. Moreover, many segmentation workflows are configured and controlled by multiple input parameters, which have to be tuned in order to optimize segmentation quality for a given dataset. The parameters oftentimes have to be re-tuned when the segmentation workflow is to be used for a new set of images. This problem is referred to in this work as *the problem of parameter tuning*, i.e., the problem of finding a set of parameter values that generate accurate segmentation results for a set of images.

Our work is motivated by the fact that manual parameter tuning is very time-consuming and error-prone, particularly in the context of WSI analysis [30, 31]. An alternative approach is to manually segment several image tiles accurately, generating a ground truth segmentation set for a given image dataset, and then apply a computerized method to search for a set of parameter values that produce the best segmentation output with respect to the ground truth. This also is a challenging task, because parameter search space for a segmentation workflow can be very large, containing billions or trillions of points as is shown in Table 1 for the example segmentation workflows studied in this work. Moreover, the computational cost of evaluating a single point in the parameter space, which involves segmenting an image tile and computing a quality metric for the segmentation results, can be very high. The parameter tuning process becomes even more challenging when it involves multiple conflicting objectives such as the quality of segmentation output and the execution time of the segmentation process [32–38].

The problem of parameter tuning and optimization has been investigated in several projects [32, 39–50]. Majority of the previous works use solutions for particular segmentation models. A pseudo-likelihood is used in [47] to estimate parameters for a conditional random field–based algorithm. Graph cuts are employed to compute maximum margin efficient learning in segmentation parameters [48]. Open-Box [50] is another interesting solution specific to segmentation algorithms based on spectral clustering. It deals with the optimization by exposing key components of the segmentation to the user. The Tuner system [30] treats a segmentation pipeline as a black-box process, but it uses statistical models to explore the parameter space. Our work has several novel improvements over the prior art. Our approach tunes a segmentation workflow as a black-box with efficient optimization algorithms that quickly converge to desired results. It also allows for the use of multiple auto-tuning algorithms and multiple objectives as well as several domain-specific metrics for evaluating algorithm output. It is implemented to take advantage of high-performance computing (HPC) systems to accelerate runs in the parameter-tuning phase. Another related work proposed the use of parameter auto-tuning and efficient parameter sensitivity analysis [45] in microscopy image with single-objective optimization. We extend the previous work to develop a multi-objective parameter tuning software platform. Our contributions can be summarized as follows:

1. We have adapted multi-objective optimization algorithms, some of which were developed to optimize performance of other classes of applications, in pathology image analysis, and implemented them in an integrated platform. As such, our goal is to demonstrate that the target class of applications can substantially benefit from these methods and to evaluate the effectiveness of efficient algorithms in the domain and, for instance, provide a reference and understand their performance.

Table 1 Input parameter sets of three example segmentation workflows

Parameter	Description	Range value
B/G/R	Background detection thresholds	[210, 220,…, 240]
T1/T2	Red blood cell thresholds	[2.5, 3.0,…, 7.5]
G1/G2	Thresholds to identify candidate nuclei	[5, 10,…, 80]
		[2, 4,…, 40]
MinSize	Area threshold of candidate nuclei	[2, 4,…, 40]
MaxSize	Area threshold of candidate nuclei	[900,…, 1500]
MinSizePl	Area threshold before Watershed	[5, 10,…, 80]
MinSizeSeg	Area threshold from final segmentation	[2, 4,…, 40]
MaxSizeSeg	Area threshold from final segmentation	[900,…, 1500]
FillHoles	Propagation neighborhood	[4-conn, 8-conn]
MorphRecon	Propagation neighborhood	[4-conn, 8-conn]
Watershed	Propagation neighborhood	[4-conn, 8-conn]

(a) Parameters of the Morphological Operation- and Watershed-based segmentation pipelines. The search space contains about 21 trillion parameter points.

Parameter	Description	Range value
OTSU	OTSU threshold value	[0.3, 0.4,…, 1.3]
Curvature weight	Curvature weight (CW) for the Level Set	[0.0, 0.05,…, 1.0]
MinSize	Area threshold for nuclei	[1, 2,…, 20]
MaxSize	Area threshold for nuclei	[50, 55,…, 400]
MsKernel	Radius in Mean-Shift calculation	[5, 6,…, 30]
LevelSetIt	Number of iterations of the Level Set computation	[5, 6,…, 150]

(b) Parameters of the Level Set- and Mean-Shift-based segmentation pipeline. The search space contains about 1.4 billion parameter points.

Parameter	Description	Range value
OTSU	OTSU threshold value	[0.3, 0.4,…, 1.3]
Curvature weight	Curvature weight (CW) for the Level Set	[0.0, 0.05,…, 1.0]
MinSize	Area threshold for nuclei	[1, 2,…, 20]
MaxSize	Area threshold for nuclei	[50, 55,…, 400]
Watershed	Propagation neighborhood	[4-conn, 8-conn]
LevelSetIt	Number of iterations of the Level Set computation	[5, 6,…, 150]

(c) Parameters of the Level Set- and Watershed-based segmentation pipelines. The search space contains about 96 million parameter points.

2. We show via experimental evaluation that pathology image analysis can substantially benefit from multi-objective parameter optimization that targets the quality of analysis output and the execution time of analysis. The proposed platform supports two objectives in parameter optimization: reducing the execution time of an image segmentation run and increasing the quality of the segmentation results. These objectives conflict with each other in that reducing execution time often leads to reduction in segmentation quality, and vice versa. Our approach identifies parameters that produce segmentation results with good-quality and reduced segmentation execution time. Users can adjust the weights of the objectives (quality vs. execution time) to tune the segmentation pipeline according to their priorities. Our experimental evaluation with three nucleus segmentation workflows shows very good results. First, the quality of segmentation can be improved up to $\times 7.8$ when only one objective is set. Second, the multi-objective tuning can speed up the segmentation process by $\times 11.79$ while improving the segmentation quality by $\times 1.28$ (please see Table 2).

3. We package our implementation as a Docker container with a RESTFul interface for easy deployment as a service. A user can run the parameter auto-tuning infrastructure as a service on a local machine or as a remote server in a Cloud environment. A Cloud deployment can benefit scientists without access to high-performance systems. Our implementation supports a variety of spatial queries that include spatial cross-matching, overlap, and spatial proximity computations, used to derive segmentation quality metrics as Dice and Jaccard coefficients [51].

4. We integrate the Docker container with the 3D Slicer [52] as part of the SlicerPathology extension [53, 54]. This integration provides a graphical user interface for a researcher to interact with the infrastructure.

5. We propose a high-performance computing approach that integrates parameter auto-tuning processes and spatial

query capabilities for comparing analysis results in order to reduce the execution time of auto-tuning.

Materials and Methods

An overview of the auto-tuning platform is presented in Fig. 1. Using the 3D SlicerPathology extension from the 3D Slicer, a user specifies input images and corresponding segmentation masks, the set of parameters to be tuned and their value ranges, and the optimization algorithm to be used. In step 2 in Fig. 1, the user can employ the SlicerPathology module [37] in the 3D Slicer [36] to create a segmentation mask for an input image. The user then invokes the auto-tuning infrastructure to execute the auto-tuning task—we have implemented a prototype extension to SlicerPathology so a user can submit the auto-tuning task through the SlicerPathology graphical user interface. The task is received in the infrastructure through a web-service interface and executed as the tuning system becomes available. During the tuning, the optimization algorithm selects parameter sets and the segmentation workflow is executed for each of the sets. The output of an execution are the segmentation results (image masks), which are compared to ground truth results using spatial metrics (Dice and Jaccard coefficients) [51]. The value of the computed metric is used as input to the auto-tuning or optimization algorithm to guide the search. This search and comparison loop is repeated until either the desired objective is met or the maximum number of iterations is reached. In the rest of this section, we describe the multi-objective strategy used to tune the application execution times and result quality, the optimization algorithms employed and evaluated in this work, the application workflows used in the evaluation, and the implementation details of the auto-tuning framework.

Multi-objective Auto-tuning Methodology

Multi-objective tuning deals with optimization problems with conflicting goals, e.g., result quality and execution time or performance and resource usage. There are multiple solutions to a multi-objective optimization problem. Existing works in the literature typically employ one of three fundamental approaches [33, 35]: (i) A posteriori in which the largest number of possible solutions is first sought, and then, the one that best fits the problem is selected; (ii) A priori *insertion*, in which there exists a preference about the type of solution most appropriate to the problem at hand. The search is directed to find that type of solutions; and (iii) *progressive insertion of preferences* that is done by targeting the choices of a decision-maker (a person skilled in the problem domain) during optimization to regions that are more likely to contain appropriate solutions.

Our work employs an a priori *insertion* approach, because (a) in our problem, it is not possible to test a large number of parameter's combinations as in the a posteriori. Calculating a large number of combinations (i.e., the set of Pareto optimal solutions or those in which none of the objectives can be improved without affecting another objective [32, 37]) would be very expensive due to the high cost of the test or fitness function (segmentation) [35], and (b) the use of *progressive insertion of preferences* during the search would require the intervention of a domain expert, but we want to carry out the optimization process automatically and minimize the users' burden.

We have chosen the scalarization approach [35, 36, 55] for inserting a priori preferences and combining objectives in an optimization function. This approach can efficiently solve multi-objective optimization problems through adaptation of single-objective optimization methods. There exists a large class of optimization methods that efficiently solve single-objective problems and can be used with scalarization.
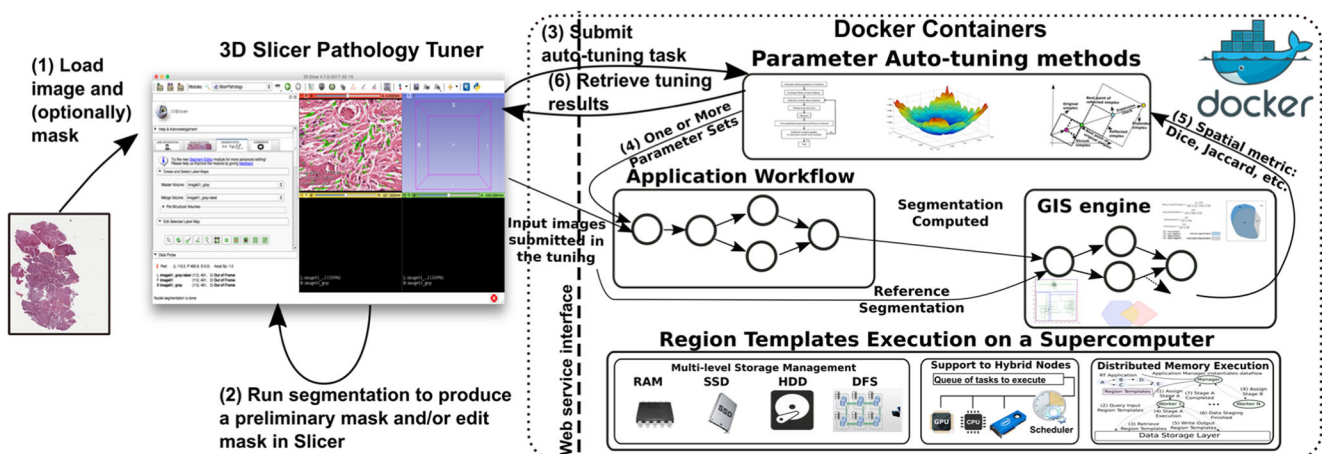

Fig. 1 Auto-tuning framework

Moreover, in our case, we are typically interested in a given set of weights or preference selected by a user according to the objective of the optimization. We use a linear scalarization that assigns weights (w_i) to each objective ($f_i(x)$), so that the sum of the weights is equals to 1. For N objectives, the function to be optimized is as follows: $f(x) = \sum_{i=1}^{N} w_i \times f_i(x)$. In our evaluation, we mainly target the simultaneous optimization of a segmentation quality/accuracy that is maximized along with the minimization of the segmentation workflow execution time.

Optimization Algorithms

The optimization methods implemented in our work include Nelder-Mead (NM) simplex [56], Parallel Rank Order (PRO) [39], Bayesian Optimization Algorithm (BOA) [42], and Genetic Algorithm (GA) [35, 40]. Briefly, the NM is a commonly used optimization algorithm in multi-dimensional space problems in which the derivatives are may not be known. It is a heuristic search method that explores the search space using a simplex or a special polytope with $k + 1$ vertices, where k is the dimensionality of the search space. The search is carried out by modifying and moving the simplex through a set of complementary operations, such as reflection expansion, contraction, and shrink, that are intended to either quickly find a minimum in the region being explored or to leave a local minima region. The PRO is similar to NM, meaning that it uses the same searching mechanisms and operations, but it enables the evaluation of multiple points of the simplex concurrently. The GA optimization algorithm models the auto-tuning problem with individuals whose genes represent the application parameters. In our GA algorithm, the first set of individuals is randomly initialized, whereas they are modified or evolved among iterations of the algorithm using crossover and mutation. The crossover uses one-point crossover in which individuals' parents are combined by swapping parts of their genes starting into a single point. The crossover between pairs of individuals occurs with a probability of C. After that transformation, the mutation in each gene can take place with a probability of M. Once the new population is created, it is evaluated via segmentation workflow runs (fitness function), and the results are used again to create another generation of individuals. The tuning with GA executes for a number of iterations selected by the end-user. The probabilities C and M were selected experimentally as 0.5 and 0.3, respectively, as those are the values that maximize the GA performance in our case.

The BOA [42] is an iterative process that develops a global statistical model of the objective function. This probabilistic model is exploited to make decisions about the next point in the search space for which the objective function should be evaluated. It uses information from the model or previous runs in this decision and minimizes the number of function evaluations. As such, this method is expected to be competitive [57] for objective functions whose evaluations are costly, which is the case of medical image analysis.

Segmentation Workflows

We have evaluated our approaches using three segmentation workflows presented in Fig. 2. We include these workflows as part of our software distribution. The analysis workflows of tissue images used in our studies compute information from the images that include segmented objects (e.g., nuclei or cells) and about 30–50 features per object (shape, intensity, and texture features). The overall image computation workflow includes normalization, segmentation, feature computation, and other data analysis stages, the first three being the most expensive. In this work, we focus on the study of the segmentation stage. Figure 2 presents three analysis workflows used in this work. The workflows have the same high-level structure, but they are different with respect to the approaches used to implement the segmentation stage. The first workflow (Fig. 2a) uses Morphological Operations and Watershed in the segmentation [58], whereas the second one (Fig. 2b) performs the segmentation based on Level Set and Mean-Shift clustering [53]. The third workflow uses Level Set and Watershed for declumping [53] (Fig. 2c). The operations within these workflows are shown in the figures. Please see Table 1 for the list of parameters for the segmentation phase of each workflow.

Software Implementation

This section presents the implementation aspects of the main modules of our auto-tuning platform presented in Fig. 1. First, in the "Execution on High-Performance Machines with Region Templates" section, we describe the region template framework that is used to implement the applications for efficient execution on distributed high-performance computing systems and is the baseline solution in which the tuning methods and spatial comparison engine were integrated. The spatial comparison engine that computes the differences between different segmentation workflows is detailed in the "Spatial Query Module for Computing Error Metrics" section. The containerization and integration of our solutions with the 3D Slicer for simplifying the system deployment and interaction with the proposed tuning tools are then discussed in the "Containerization and Integration with the 3D Slicer" section.

Execution on High-Performance Machines with Region Templates

The auto-tuning methods are deployed in the region templates (RTs) [59] for efficient execution of image analysis pipelines
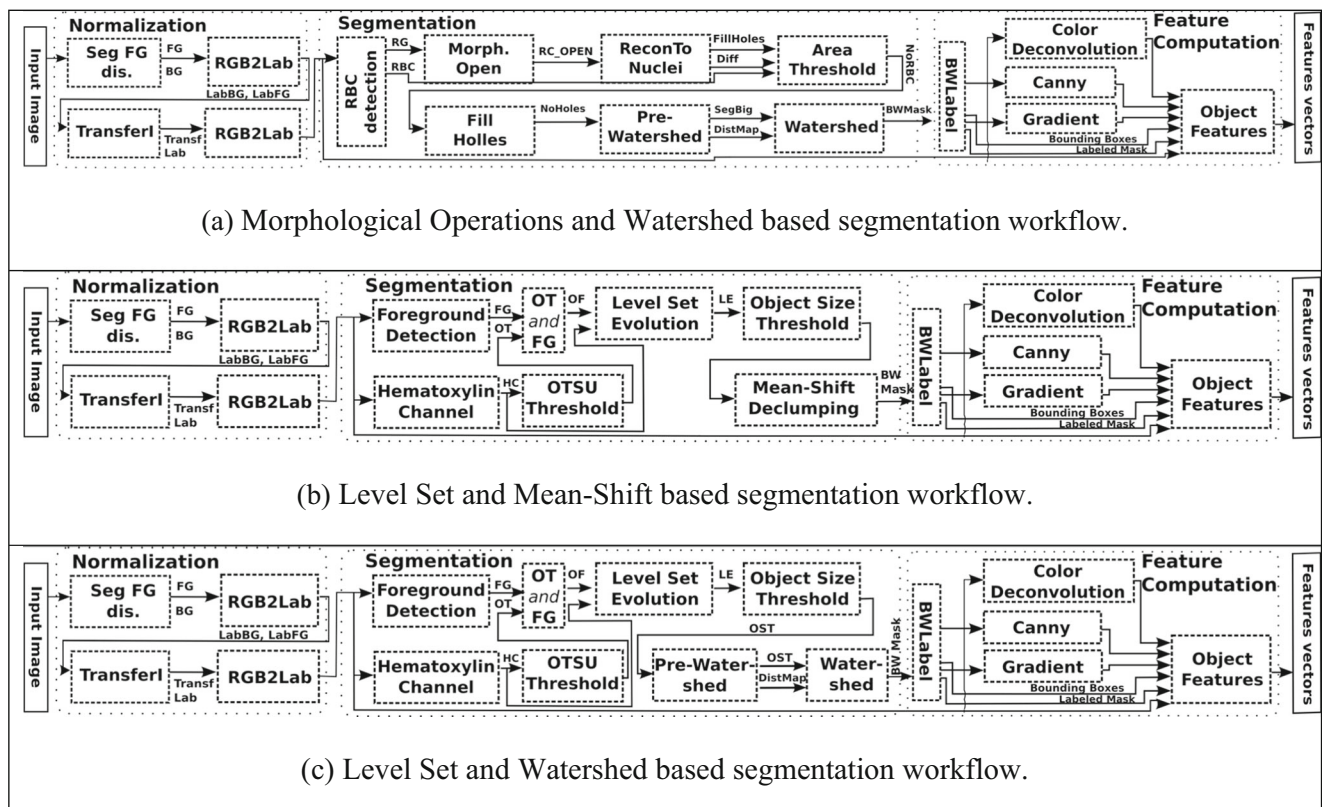
(a) Morphological Operations and Watershed based segmentation workflow.

(b) Level Set and Mean-Shift based segmentation workflow.

(c) Level Set and Watershed based segmentation workflow.

Fig. 2 Three example image analysis workflows. The workflows are built using the same coarse-grain stages (normalization, segmentation, and feature computation), but they differ in terms of the techniques used to implement the segmentation. The first (figure **a**) uses Morphological Operations to identify candidate cells along with Watershed to delineate and separate clumped cells. The second (figure **b**) applies the Level Set strategy to create the initial or candidate cell set and a Mean-Shift-based clustering to delineate and separate clumped cells. The third (figure **c**) uses the Level Set to create candidate cells and the Watershed to separate clumped ones

on parallel machines. The integration of applications or workflows with RT for tuning is performed using an interface in which the user exports the parameters to be tuned and their value ranges. In the same interface, the user can also choose the optimization algorithm to be used and modify the weights given to each parameter in multi-objective tuning tasks.

We implemented the three example segmentation pipelines in RT to tune and accelerate their execution. A RT application stage itself can be composed of lower-level operations organized into another dataflow, and different scheduling strategies may be used at each level. The runtime system implements a Manager-Worker execution model for assignment of work among nodes of a distributed memory machine. The application Manager creates instances of (coarse-grain) stages and exports the dependencies between them. The assignment of work from the Manager to Worker nodes is performed at the granularity of a stage instance using a demand-driven mechanism.

Each Worker uses multiple computing devices in a node by dispatching fine-grain tasks (operations that implement a stage instance) for execution in a CPU core or a co-processor (e.g., Intel Phi or GPU). Different scheduling strategies and runtime optimizations were developed targeting heterogeneous computing devices [60, 61]. Further, RT also implements optimizations to reduce the cost of exchanging data among stages and to improve data access locality implemented and available to the applications transparently [62].

Spatial Query Module for Computing Error Metrics

The computation of quality (or error) metrics to guide the parameter tuning process involves spatial queries and comparison operations. We implemented a spatial query module, called RT GIS Engine, to speed up the quantitative comparison of segmentation results via a query-based interface with which queries are expressed using a SQL-like language. The implementation uses a query engine [63] that supports several spatial operations, including spatial cross-matching, overlay of objects, spatial proximity computations between objects, and global spatial pattern discoveries. These operations are used to compute high-level metrics for comparison of results from different analysis runs. The quality metrics include Dice coefficient, Jaccard coefficient, intersection overlapping area, and non-overlapping area [51].

The workflow for computation of the comparison and error metrics is depicted in Fig. 3. The user application computes a mask and passes it along with the reference mask as input to the
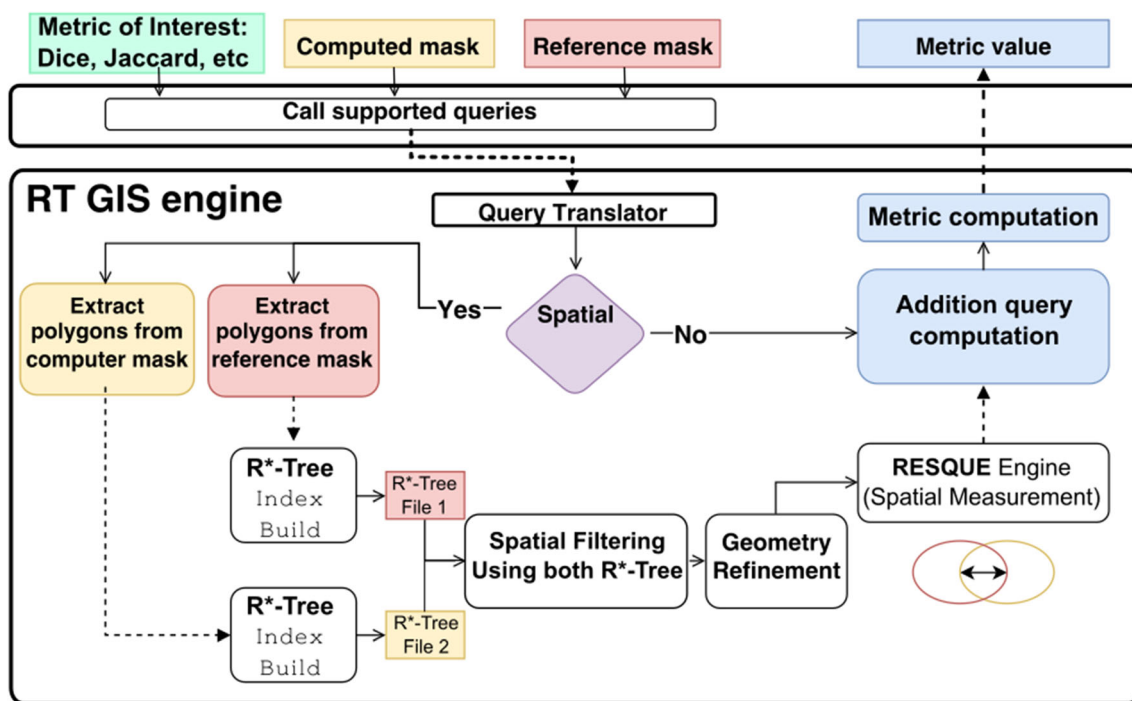
Fig. 3 Metrics' computation workflow. Mask computed by the segmentation workflow and the reference mask (ground-truth) are received along with the metric of interest. Segmented objects (nuclei, etc.) in the masks are converted into polygons and indexed using Hilbert R*-tree index [60] to quickly identify intersecting objects. The overlapping objects are passed to the RESQUE Engine to compute the actual spatial metrics. The query engine is deployed as a generic workflow stage in RT. As such, the query engine can execute on parallel machines and can be reused among application workflows

RT GIS engine. In order to execute spatial queries, the objects (i.e., cell nuclei) identified in the masks are converted into polygons and processed into the query engine. Each of the metrics implemented uses a set of queries (expressed using a SQL-like language) that are combined to compute the user-selected metric. The spatial queries are executed using a Hilbert R*-tree index [64] to quickly identify intersecting objects and minimize the computation costs. First, the R*-trees are built from objects minimum bounding boxes in each mask (computed and reference). The spatial filtering operation is performed to identify possibly overlapping objects (those with intersecting bounding boxes), which are refined to those that overlap. This set is passed to a final phase that computes the spatial measurements. The entire query engine is deployed as a generic workflow stage in RT. As such, the query engine can execute on parallel machines and may have several copies running in the computing environment as a regular RT application stage.

Containerization and Integration with the 3D Slicer

In order to simplify the use of the auto-tuning methods and the execution platform proposed in this work, we have (i) packaged our implementation as a service in a Docker container [65] and (ii) exported the main functionalities through a user-friendly interface with the pathology module [37] of the 3D Slicer [36]. With the Docker container of the auto-tuning

platform, the user can easily build the entire system and deploy it into local or remote computing systems (i.e., Cloud providers), whereas the pathology module contains a graphical interface for interacting with the entire system.

The auto-tuning process is invoked by calling the service hosted in the Docker container. The user specifies the input image, the reference mask, a set of parameters to be tuned, an optimization algorithm, and a quality/error metric. This request is sent to the service via a RESTFul interface call. The service will parse the input request and insert it into a queue of tuning requests. A tasks hander is returned by the services and can be used by the client application (SlicerPathology in our case) for querying the status or results of the given tuning request. As such, the client side of the application or Slicer is not blocked during the execution of the auto-tuning. The queue of requests is handled internally by the web-service running into the container. In our current implementation, the web-service processes, those requests by executing multiple instances of the RT implementation of the application being tuned in a demand-driven basis as the computing resources (CPU cores or GPUs), become available. Once the request is processed, it is placed into a queue of completed requests and remains available for the user to retrieve the results. The auto-tuning execution itself goes through a set of steps that consists of executing the segmentation workflow, computing the quality metric, and computing a new set of parameters to evaluate, as described in the "Materials and Methods" section.

This process is executed until it converges. Another important aspect that we want to highlight is that the web-service interface is built independently of the 3D Slicer. As such, the same interface could be used to integrate with other tools.

Our Slicer module interface provides a set of additional features that can speed up and simplify the tuning. For instance, it enables the integration with remote repositories[1] for loading the data employed in the auto-tuning process, which alleviates the burden on the user with respect to the data management. Additionally, tools available in the 3D Slicer module allow for the user to delineate objects and create reference (ground truth) masks manually. This process may also be sped up by starting the ground-truth generation with a mask produced by the segmentation workflow and parameter values chosen by the scientist (step 2 in Fig. 1). The results of this segmentation are presented in the Slicer and can be corrected/modified manually using the editing tools to change the polygons that describe objects found in the segmentation, instead of starting the mask generation from scratch.

Results and Discussion

Experiment Setup

The experiments were conducted on the Stampede-distributed memory machine. Each node on Stampede has dual Intel Xeon E5-2680 processors, an Intel Xeon Phi SE10P co-processor, and 32 GB RAM. The nodes are inter-connected via Mellanox FDR Infiniband switches. The auto-tuning methods and the example segmentation pipelines are implemented in the region template (RT) framework [59] for efficient execution on this machine. In this implementation, input images are partitioned into image tiles—each image tile can be processed independent of other tiles for nucleus segmentation.

In the experiments, the genetic algorithm (GA) method was configured to evolve 10 individuals for 10 generations and set up with a mutation rate of $M = 0.3$ and a crossover rate of $C = 0.5$, because this setup experimentally led to best results. The NM, PRO, and BOA algorithms were configured to stop after testing 100 points in the search space, which asserts that all optimization algorithms perform 100 application runs. We have repeated all experiments 10 times. Average standard deviation is smaller than 1% for the Watershed-based workflows and 3% for the Mean-Shift workflow. The time to select the next set of parameters varies among the optimization methods. It was about 77 s for BOA and around 10 ms for the other algorithms. This cost is amortized by the high execution times of the segmentation. The quality of segmentation results was quantified with the *average Dice coefficient*, which ranges

from 0.0 to 1.0, in which higher values mean a better agreement with ground-truth segmentation. The experiments used 15 image tiles extracted from Glioblastoma multiforme (GBM) WSIs and manually segmented by a pathologist.

Multi-objective Parameter Auto-tuning: Segmentation Quality and Execution Time

This section evaluates our methodology to maximize the quality of segmentation results and to minimize the execution time of the 15 images used in our analysis. The objectives are combined into a single optimization function using scalarization, and the user defines the weight for each objective. To simplify the weighting, we have normalized execution times between 0 and 1 (higher is better), which is the same range of the average Dice coefficient used for quality.

Experimental results with multi-objective tuning are presented in Table 2 for the three segmentation workflows. The weights of the objectives are varied to evaluate the ability of the parameter auto-tuning framework to find parameters for different user preferences. The results for single-objective tuning (execution time weight set to 0) are also presented for reference. The experimental results for the workflow using Morphological Operations and Watershed show that the optimization algorithms were able to improve the quality of segmentation results by up to × 1.14 and speed up execution by × 1.07 compared with the segmentation quality and execution times using the default parameters. Moreover, the results show a consistency in segmentation quality improvement when the weight of this component is increased, i.e., higher quality metric weights resulted in better Dice values. Also, GA and NM achieved a slightly better performance than the other methods.

Results for the Level Set–based workflows are also presented in Table 2. As is shown in the table, NM and GA attained the best aggregated multi-objective optimization value (marked in bold font) when segmentation quality and execution times are considered in all configurations. When the workflow used the Watershed declumping method, GA was able to find parameter sets with which the quality metric was increased by × 1.28 and the execution time reduced by 11.79× compared with the default parameters. The parameter sets found for the workflow with Level Set and Mean-Shift are also very good. The segmentation results for this workflow can be significantly improved, while at the same, time the execution time is reduced. When the segmentation operations are applied in datasets with thousands of WSIs, these improvements can translate to significant reduction in resource usage, much faster analysis of data, and ability for large-scale studies.

The PRO optimization method also was able to find parameter configurations with which the execution times of the workflows significantly improved. However, the

Table 2 Results for the multi-objective auto-tuning as compared to the application default parameters. The experiments were carried out using all supported optimization algorithms as the weights for the results quality (average Dice) and execution times were varied. The results are presented as an average for the 15 input example images, and the best results for the multi-objective function are highlighted in each case

Segmentation algorithm	Weights		Average Dice						Speed up vs. default			
	Metric	Time	Default	GA	NM	PRO	BOA	Improv.	GA	NM	PRO	BOA
Morphological Operations + Watershed	1	0	0.65	0.77	0.76	0.76	*0.78*	1.20	–	–	–	–
	1/2	1/2	0.65	0.70	0.72	**0.73**	0.70	1.10	1.09	1.08	**1.07**	1.07
	2/3	1/3	0.65	**0.74**	0.74	0.74	0.72	1.13	**1.08**	1.07	1.06	1.06
	4/5	1/5	0.65	**0.75**	*0.75*	0.74	0.75	1.14	**1.07**	1.07	1.05	1.04
Level Set + Mean-Shift declumping	1	0	0.61	**0.79**	0.75	0.73	0.78	1.29	–	–	–	–
	1/2	1/2	0.61	0.70	**0.66**	0.63	0.71	1.14	3.99	**4.91**	5.09	0.67
	2/3	1/3	0.61	0.73	**0.69**	0.68	0.72	1.19	2.41	**3.74**	2.85	0.61
	4/5	1/5	0.61	0.74	**0.71**	0.71	0.75	1.21	1.68	**2.61**	2.38	0.48
Level Set + Watershed declumping	1	0	0.61	**0.79**	0.78	0.75	0.78	1.29	–	–	–	–
	1/2	1/2	0.61	0.74	**0.69**	0.69	0.73	1.13	14.26	**16.05**	13.01	1.38
	2/3	1/3	0.61	**0.77**	0.73	0.72	0.72	1.26	**12.68**	13.99	11.85	1.42
	4/5	1/5	0.61	*0.78*	0.75	0.71	0.76	1.28	**11.79**	11.61	10.96	1.27

reductions in execution times were achieved with a higher penalty in the quality metric than what GA achieved. The BOA method, on the other hand, was not able to find parameter sets that resulted in similar levels of reduction in the execution times. Indeed, for Level Set with Mean-Shift, parameters selected by BOA resulted in a higher execution time vs. those using the default parameters. These results indicate that BOA is efficient for single-objective tuning, while GA and NM are the best methods for multi-objective tuning.

We also examined why the gains in execution time with the Morphological Operation- and Watershed-based workflows were smaller than those with the Level Set–based segmentations. We found that in practice, variations in input parameters have a small impact in the execution time of the first workflow. As such, it was not a failure of the optimization methods in finding good parameter sets, but a characteristic of the segmentation strategies used.

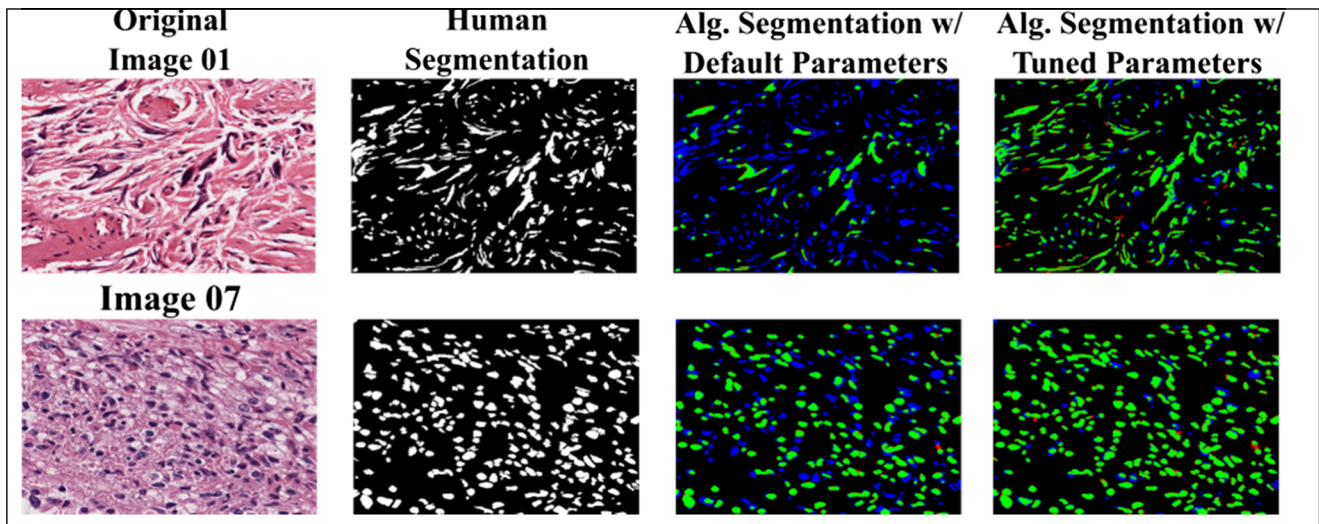Figure 4 illustrates improvements in segmentation results via parameter auto-tuning for two images. The

Fig. 4 Two sample image patches are presented with human segmentation and the Level Set with Mean-Shift declumping workflow segmentation using default and tuned parameter values. The green is used to represent areas of agreement between algorithm and human, blue represents objects present in the human segmentation and not algorithm results, and red are those (few) areas found by the algorithm and not by human. The first image (image 01) has 0.44 and 0.80 Dice values, respectively, with default and tuned parameters. For the second image (image 07), the Dice with default parameter is 0.71 and it is 0.83 after tuning. Even for image 07 in which the Dice improvement is smaller, we can observe that a significant number of nuclei missed with default parameters are identified after tuning

Table 3 One group cross-validation evaluation. Average Dice improvements and execution time reductions for all segmentation workflows in a Monte Carlos cross-validation with 10 repetitions. The best results were highlighted in bold

Segmentation algorithm	Weights		Average Dice (0–1)		
	Metric	Time	Default	Tuned (GA)	Speed up
Morphological Operations + Watershed	1	0	0.65	**0.72**	–
	1	1	0.66	**0.68**	1.09
	2	1	0.65	**0.71**	1.07
	4	1	0.66	**0.73**	1.06
Level Set + Mean-Shift declumping	1	0	**0.63**	0.60	–
	1	1	**0.59**	0.57	3.10
	2	1	**0.62**	0.58	2.32
	4	1	**0.62**	0.60	1.64
Level Set + Watershed declumping	1	0	**0.60**	**0.60**	–
	1	1	**0.60**	0.58	8.97
	2	1	**0.59**	**0.59**	10.50
	4	1	**0.62**	0.60	7.37

The best results were highlighted in bold

images were segmented with the Level Set– and Mean-Shift-based workflows. The second column in the figure shows the segmentation results from manual segmentation; the third column shows the results with the default parameters; and the fourth column with the auto-tuned parameter values. The green areas in the images refer to agreement between the segmentation generated by the workflow and the manual segmentation, whereas the blue ones are the areas missed by the segmentation pipeline. The red areas refer to those segmented by the pipeline but not by the pathologist. Please notice that there are very few red points, which means that the algorithms do not significantly detect objects also not found by human (please zoom in for better visualization).

Cross-Validation

This section computes Monte Carlo cross-validation analysis repeated 10 times that separates the 15 input images into two sets for training and testing. The application parameters are tuned using the training set containing 20% of the images and further evaluated into the remaining 80% images of the testing set. The experiments used the GA optimization and evaluated 100 points. The tuning experiments were repeated 10 times, and the standard deviation is smaller than 2% for the Morphological Operation workflow and 6% for the Level Set workflows.
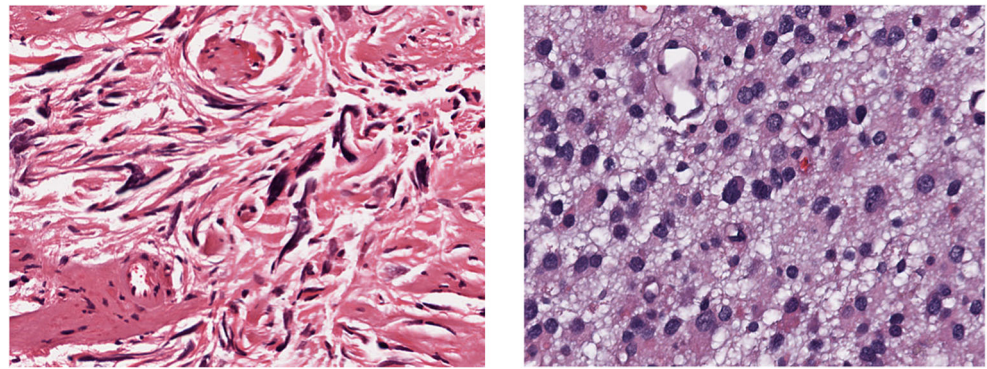
The results are presented in Table 3. Since the results are computed using a random selection of the training and testing sets, results using different weights employ different sets, and as such, default metric values are not the same.

However, the same sets are used within each weight combination (table row) for a fair comparison between default and tuned parameters. For the Morphological Operation–based workflow, the tuning platform found a set of parameters that improved the quality results in over × 1.10 and the segmentation speed up by 1.09×, and similarly to the previous experiments, it finds different trade-offs among segmentation result quality and speed of the execution as weights are varied.

For the Level Set workflows, regardless of the declumping method used, the optimization algorithms were not able to find parameter sets that improve the segmentation quality and execution time together. However, the execution times were significantly improved (up to 10.5×) for both declumping cases for all weight combinations with a small decrease in the segmentation quality. The Level Set workflows are very sensitive to the nuclei shape, and the parameters used for elongated nuclei, for instance, will not perform well in round ones. This indicates that a single set of parameters optimized for images containing different cell shapes will not be optimal for neither of them. Instead, the algorithm should use different parameter sets according the expected nuclei structure.

In order to validate this observation, we performed another cross-validation in which images were separated into two groups: those with more elongated and those with round nuclei. We executed a cross-validation in each of the groups separately. We classified five images in the first group and 10 in the second group. Examples of images in these groups are presented in Fig. 5. For the first group, one image is selected for training and the other four images to test,

Fig. 5 Examples of images in each of the two groups for cross-validation. The first group of images contain those with elongated nuclei (group 1), whereas the second group of images contain more round-shaped ones (group 2). The Monte Carlos cross-validation with 10 repetitions is executed in each of the groups isolated

whereas in the second group, two images are included in the training and the other eight images in the testing set.

The results are presented in Table 4. The optimization algorithm found parameters that improved the quality and execution time of the workflows for most of the weights. In the workflow with Watershed declumping, the same quality result observed in the single-objective optimization (0.70) was attained, but the workflow was accelerated in 10.25× in the multi-objective case.

Conclusions

Pathology image segmentation workflows are sensitive to changes in input parameters, and an input parameter configuration that performs well with a set of images may not compute good segmentation results, for instance, for another dataset. Tuning the application parameters is important to maximize the quality of the results and/or reduce the application's execution time. The main challenges with tuning include (i) the large number of parameter combinations; (ii) the high cost of evaluating a point in the search space due compute expensive nature of the segmentation workflow; and (iii) the difficulty of manually evaluating the search space and the quality of a segmentation result.

In order to address these challenges, we have developed a novel multi-objective optimization framework, implemented as an integrated suite of optimization methods and tools, for automatic parameter tuning of segmentation workflows in pathology image analysis and evaluated it with three real-world segmentation applications. In most experiments, we have observed significant improvements over default parameter values. Our framework was able to improve the average quality of the 15 images in × 1.28 and, at the same time, decrease the segmentation execution time by 11.79×. The impacts of these improvements are very significant to provide segmentation results with a better quality. This should in turn allow for attaining better overall analysis results in integrated studies using cell-level characterization, which are steps that typically follow the segmentation and feature extraction phases. This is essential for enabling the use of these technologies in clinical settings, as accurately segmented objects/extracted features, will lead to more reliable results. Further, the gains in speed up will provide the ability of quickly analyzing large-scale datasets, which are becoming available but are not yet fully exploited. Thus, we expect that pathology image analysis workflows should be submitted for a systematic tuning, such as proposed in this paper, before they are used in practice in order to maximize their benefits.

In addition, the evaluation of multiple optimization algorithms has highlighted that a single algorithm will not always

Table 4 Two-group cross-validation test. Average metric improvements and execution time reductions for the Level Set–based segmentation workflow using the two groups of validation data (4 + 8 images). The best results are highlighted

Segmentation algorithm	Weights		Average Dice (0–1)		
	Metric	Time	Default	Tuned (GA)	Speed up
Level Set + Mean-Shift declumping	1	0	0.61	**0.69**	–
	1	1	0.62	**0.62**	8.56
	2	1	0.61	**0.62**	1.72
	4	1	0.62	**0.69**	1.37
Level Set + Watershed declumping	1	0	0.62	**0.70**	–
	1	1	0.61	**0.62**	14.55
	2	1	0.61	**0.66**	13.26
	4	1	0.61	**0.70**	10.25

be able to attain the best performance. Instead, depending on the optimization task configuration (objective, workflow choice, and input image), different algorithms may attain better performance. For instance, the BOA algorithm attained good results in single-objective runs, but was less efficient than the GA in a configuration in which we want to tune for quality and execution time.

References

1. Muenzel D, Engels H-P, Bruegel M, Kehl V, Rummeny E, Metz S: Intra-and inter-observer variability in measurement of target lesions: implication on response evaluation according to RECIST 1.1. Radiology and oncology 46(1):8–18, 2012

2. Grilley-Olson JE, Hayes DN, Moore DT, Leslie KO, Wilkerson MD, Qaqish BF, Hayward MC, Cabanski CR, Yin X, Socinski MA: Validation of interobserver agreement in lung cancer assessment: hematoxylin-eosin diagnostic reproducibility for non-small cell lung cancer: the 2004 World Health Organization classification and therapeutically relevant subsets. Archives of pathology & laboratory medicine 137(1):32–40, 2012

3. Warth A, Stenzinger A, von Brünneck A-C, Goeppert B, Cortis J, Petersen I, Hoffmann H, Schnabel PA, Weichert W: Interobserver variability in the application of the novel IASLC/ATS/ERS classification for pulmonary adenocarcinomas. European respiratory journal 40(5):1221–1227, 2012

4. Yoon SH, Kim KW, Goo JM, Kim D-W, Hahn S: Observer variability in RECIST-based tumour burden measurements: a meta-analysis. European Journal of Cancer 53:5–15, 2016

5. Nakazato Y, Maeshima AM, Ishikawa Y, Yatabe Y, Fukuoka J, Yokose T, Tomita Y, Minami Y, Asamura H, Tachibana K: Interobserver agreement in the nuclear grading of primary pulmonary adenocarcinoma. Journal of Thoracic Oncology 8(6):736–743, 2013

6. Bueno-de-Mesquita J, Nuyten D, Wesseling J, van Tinteren H, Linn S, van De Vijver M: The impact of inter-observer variation in pathological assessment of node-negative breast cancer on clinical risk assessment and patient selection for adjuvant systemic treatment. Annals of oncology 21(1):40–47, 2010

7. Matasar M, Shi W, Silberstien J, Lin O, Busam K, Teruya-Feldtein J, Filippa D, Zelenetz A, Noy A: Expert second-opinion pathology review of lymphoma in the era of the World Health Organization classification. Annals of oncology:mdr029, 2011

8. Rizzardi AE, Johnson AT, Vogel RI, Pambuccian SE, Henriksen J, Skubitz AP, Metzger GJ, Schmechel SC: Quantitative comparison of immunohistochemical staining measured by digital image analysis versus pathologist visual scoring. Diagnostic pathology 7(1):42, 2012

9. Berney DM, Algaba F, Camparo P, Compérat E, Griffiths D, Kristiansen G, Lopez-Beltran A, Montironi R, Varma M, Egevad L: The reasons behind variation in Gleason grading of prostatic biopsies: areas of agreement and misconception among 266 European pathologists. Histopathology 64(3):405–411, 2014

10. Netto GJ, Eisenberger M, Epstein JI, T. T. Investigators: Interobserver variability in histologic evaluation of radical prostatectomy between central and local pathologists: findings of TAX 3501 multinational clinical trial. Urology 77(5):1155–1160, 2011

11. Allsbrook WC, Mangold KA, Johnson MH, Lane RB, Lane CG, Epstein JI: Interobserver reproducibility of Gleason grading of prostatic carcinoma: general pathologist. Human pathology 32(1):81–88, 2001

12. Sørensen J, Hirsch F, Gazdar A, Olsen J: Interobserver variability in histopathologic subtyping and grading of pulmonary adenocarcinoma. Cancer 71(10):2971–2976, 1993

13. Roggli VL, Vollmer RT, Greenberg SD, McGavran MH, Spjut HJ, Yesner R: Lung cancer heterogeneity: a blinded and randomized study of 100 consecutive cases. Human pathology 16(6):569–579, 1985

14. Wilkins BS, Erber WN, Bareford D, Buck G, Wheatley K, East CL, Paul B, Harrison CN, Green AR, Campbell PJ: Bone marrow pathology in essential thrombocythemia: interobserver reliability and utility for identifying disease subtypes. Blood 111(1):60–70, 2008

15. Kong J, Cooper LA, Wang F, Gao J, Teodoro G, Scarpace L, Mikkelsen T, Schniederjan MJ, Moreno CS, Saltz JH: Machine-based morphologic analysis of glioblastoma using whole-slide pathology images uncovers clinically relevant molecular correlates. PloS one 8(11):e81049, 2013

16. Cooper LAD, Kong J et al.: Integrated morphologic analysis for the identification and characterization of disease subtypes. Journal of the American Medical Informatics Association 19(2):317–323, 2012

17. Sertel O, Kong J, Shimada H, Catalyurek U, Saltz JH, Gurcan MN: Computer-aided prognosis of neuroblastoma on whole-slide images: classification of stromal development. Pattern recognition 42(6):1093–1103, 2009

18. Kothari S, Phan JH, Stokes TH, Wang MD: Pathology imaging informatics for quantitative analysis of whole-slide images. Journal of the American Medical Informatics Association 20(6):1099–1108, 2013

19. Hsu W, Markey MK, Wang MD: Biomedical imaging informatics in the era of precision medicine: progress, challenges, and opportunities. Journal of the American Medical Informatics Association 20(6):1010–1013, 2013

20. Han D, Wang S, Jiang C, Jiang X, Kim H-E, Sun J, Ohno-Machado L: Trends in biomedical informatics: automated topic analysis of JAMIA articles. Journal of the American Medical Informatics Association 22(6):1153–1163, 2015

21. Blom S, Paavolainen L, Bychkov D, Turkki R, Mäki-Teeri P, Hemmes A, Välimäki K, Lundin J, Kallioniemi O, Pellinen T: Systems pathology by multiplexed immunohistochemistry and whole-slide digital image analysis. Scientific Reports 7(1):15580, 2017

22. Yu K-H, Zhang C, Berry GJ, Altman RB, Ré C, Rubin DL, Snyder M: Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. Nature Communications 7, 2016

23. Romo-Bucheli D, Janowczyk A, Romero E, Gilmore H, Madabhushi A: Automated tubule nuclei quantification and correlation with oncotype DX risk categories in ER+ breast cancer whole slide images. SPIE Medical Imaging, 2016

24. Leo P, Lee G, Madabhushi A: Evaluating stability of histomorphometric features across scanner and staining variations: predicting biochemical recurrence from prostate cancer whole slide images, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 2016

25. Beck AH, Sangoi AR, Leung S, Marinelli RJ, Nielsen TO, van de Vijver MJ, West RB, van de Rijn M, Koller D: Systematic analysis of breast cancer morphology uncovers stromal features associated with survival. Sci Transl Med 3(108):108ra113, Nov 9, 2011

26. Chen XS, Wu JY, Huang O, Chen CM, Wu J, Lu JS, Shao ZM, Shen ZZ, Shen KW: Molecular subtype can predict the response and outcome of Chinese locally advanced breast cancer patients treated with preoperative therapy. Oncology reports 23(5):1213–1220, 2010

27. Gurcan MN, Boucheron LE, Can A, Madabhushi A, Rajpoot NM, Yener B: Histopathological image analysis: a review. IEEE Reviews in Biomedical Engineering 2:147–171, 2009

28. Ahrens MB, Orger MB, Robson DN, Li JM, Keller PJ: Whole-brain functional imaging at cellular resolution using light-sheet microscopy. Nature Methods 10:413, 2013

29. Bankhead P, Loughrey MB, Fernández JA, Dombrowski Y, McArt DG, Dunne PD, McQuaid S, Gray RT, Murray LJ, Coleman HG, James JA, Salto-Tellez M, Hamilton PW: QuPath: open source software for digital pathology image analysis. Scientific Reports 7(1):16878, 2017

30. Torsney-Weir T, Saad A, Moller T, Hege H-C, Weber B, Verbavatz J-M: Tuner: principled parameter finding for image segmentation algorithms using visual response surface exploration. IEEE Trans. on Visualization and Computer Graphics 17(12):1892–1901, 2011

31. Held C, Nattkemper T, Palmisano R, Wittenberg T: Approaches to automatic parameter fitting in a microscopy image segmentation pipeline: an exploratory parameter space analysis. Journal of Pathology Informatics 4(2):5–5, 2013

32. Budinich M, Bourdon J, Larhlimi A, Eveillard D: A multi-objective constraint-based approach for modeling genome-scale microbial ecosystems. PLOS ONE 12(2):e0171744, 2017

33. Coello CA, Lamont GB, Veldhuizen DAV: Evolutionary algorithms for solving multi-objective problems. Berlin: Springer, 2007

34. Jordan H, Thoman P, Durillo JJ, Pellegrini S, Gschwandtner P, Fahringer T, Moritsch H: A multi-objective auto-tuning framework for parallel codes, in International Conference on High Performance Computing. Networking, Storage and Analysis (SC 12):10:1–10:12, 2012

35. Trivedi A, Srinivasan D, Sanyal K, Ghosh A: A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Transactions on Evolutionary Computation 21(3):440–462, 2017

36. Miettinen K, Mäkelä M: On scalarizing functions in multiobjective optimization. OR Spectrum 24(2), 2002

37. Miettinen KM: Nonlinear multiobjective optimization. Dordrecht: Kluwer Academic Publishers, 1998

38. Figueira JR, Fonseca CM, Halffmann P, Klamroth K, Paquete L, Ruzika S, Schulze B, Stiglmayr M, Willems D: Easy to say they are hard, but hard to see they are easy— towards a categorization of tractable multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 24(1–2):82–98, 2017

39. Tabatabaee V, Tiwari A, Hollingsworth JK: Parallel parameter tuning for applications with performance variability. Proc. of the ACM/IEEE Conf. on Supercomputing, 2005

40. Sareni B, Krähenbühl L: Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary Computation 2(3):97–106, 1998

41. Jones DR: A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21(4):345–383, 2001

42. Snoek J, Larochelle H, Adams RP: In: Pereira F, Burges CJC, Bottou L, Weinberger KQ Eds. Practical Bayesian optimization of machine learning algorithms, advances in neural information processing systems, Vol. 25. Red Hook: Curran Associates, Inc., 2012, pp. 2951–2959

43. Morris MD: Factorial sampling plans for preliminary computational experiments. Technometrics 33(2):161–174, 1991

44. Campolongo F, Cariboni J, Saltelli A: An effective screening design for sensitivity analysis of large models. Environmental Modelling & Software 22(10):1509–1518, 2007

45. Teodoro G, Kurç TM, Taveira LF, Melo AC, Gao Y, Kong J, Saltz JH: Algorithm sensitivity analysis and parameter tuning for tissue image segmentation pipelines. Bioinformatics:btw749, 2017

46. Rios LM, Sahinidis NV: Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization 56(3):1247–1293, 2013

47. Kumar S, Hebert M: Discriminative random fields: a discriminative framework for contextual interaction in classification. In: Proc. 9th IEEE International Conference on Computer Vision, 2003, pp. 1150–1157

48. Szummer M, Kohli P, Hoiem D: Learning CRFs using graph cuts. In: Proceedings of the 10th European Conference on Computer Vision: Part II, 2008, pp. 582–595

49. McIntosh C, Hamarneh G: Is a single energy functional sufficient? Adaptive energy functionals and automatic initialization. In: Lecture notes in computer science, Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2007, pp. 503–510

50. Schultz T, Kindlmann GL: Open-box spectral clustering: applications to medical image analysis. IEEE Trans. Vis. Comput. Graph. 19(12):2100–2108, 2013

51. Taha AA, Hanbury A: Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC Medical Imaging 15:29–29, 2015

52. Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J, Fillion-Robin J-CC, Pujol S, Bauer C, Jennings D, Fennessy FM, Sonka M, Buatti J, Aylward S, Miller JV, Pieper S, Kikinis R: 3D slicer as an image computing platform for the quantitative imaging network. Magnetic Resonance Imaging 30(9):10, 2012

53. Gao Y, Ratner V, Zhu L, Diprima T, Kurc T, Tannenbaum A, Saltz J: Hierarchical nucleus segmentation in digital pathology images. SPIE Medical Imaging, 2016

54. Gao Y, Aucoin N, Fedorov A, Fillion-Robin J-C: SlicerPathology, 01/15/2018, 2018; http://www.slicer.org/wiki/Documentation/Nightly/Extensions/SlicerPathology

55. Wang R, Purshouse RC, Fleming PJ: Preference-inspired coevolutionary algorithms for many-objective optimization. IEEE Transactions on Evolutionary Computation 17(4):474–494, 2013

56. Nelder JA, Mead R: A simplex method for function minimization. The Computer Journal 7(4):308–313, 1965

57. Jones DR: In: Floudas CA, Pardalos PM Eds. Direct global optimization algorithm, Encyclopedia of Optimization. New York: Springer US, 2001, pp. 431–440

58. Kong J, Cooper L, Wang F, Gao J, Teodoro G, Scarpace L, Mikkelsen T, Schniederjan M, Moreno C, Saltz J: A novel paradigm for determining molecular correlates of tumor cell morphology in human glioma whole slide images. NEURO-ONCOLOGY 15:158–159, 2013

59. Teodoro G, Pan T, Kurc T, Kong J, Cooper L, Klasky S, Saltz J: Region templates: data representation and management for high-throughput image analysis. Parallel Computing 40(10):589–610, 2014

60. Kurç TM, Qi X, Wang D, Wang F, Teodoro G, Cooper LAD, Nalisnik M, Li Z-Y, Saltz JH, Foran DJ: Scalable analysis of big pathology image data cohorts using efficient methods and high-performance computing strategies. BMC Bioinformatics 16:399:1–399:21, 2015

61. Teodoro G, Kurc T, Kong J, Cooper L, Saltz J: Comparative performance analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: a case study from microscopy image analysis. In: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 1063–1072

62. Barreiros W, Teodoro G, Kurc T, Kong J, Melo ACMA, Saltz J: Parallel and efficient sensitivity analysis of microscopy image segmentation workflows in hybrid systems. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER), 2017, pp. 25–35

63. Aji A, Wang F, Vo H, Lee R, Liu Q, Zhang X, Saltz J: Hadoop GIS: a high performance spatial data warehousing system over mapreduce. Proceedings of the VLDB Endowment 6(11):1009–1020, 2013

64. Beckmann N, Kriegel H, Schneider R, Seeger B: The R*-tree: an efficient and robust access method for points and rectangles. SIGMOD:322–331, 1990

65. Merkel D: Docker: lightweight Linux containers for consistent development and deployment. Linux J. 2014(239):2, 2014