Trainable Weka Segmentation: A Machine Learning Tool for Microscopy Image Segmentation

Ignacio Arganda-Carreras, PhD,[1] Verena Kaynig, PhD,[2] Johannes Schindelin, PhD,[3] Albert Cardona, PhD,[4] and H. Sebastian Seung, PhD[5]

[1]The French National Institute for Agricultural Research (INRA)
UMR1318, Institut Jean-Pierre Bourgin
Versailles, France

[2]Harvard School of Engineering and Applied Sciences
Cambridge, Massachusetts

[3]University of Wisconsin–Madison
Madison, Wisconsin

[4]Howard Hughes Medical Institute
Janelia Farm Research Campus
Ashburn, Virginia

[5]Princeton Neuroscience Institute, Princeton University
Princeton, New Jersey

Introduction

State-of-the-art light and electron microscopes are capable of acquiring large image datasets, but quantitatively evaluating the data often involves manually annotating structures of interest. For example, to measure the average size of mitochondria in an electron microscopy image stack, each mitochondrion has to be outlined by a human annotator. This process is time-consuming and is becoming the main bottleneck in the evaluation pipeline. To overcome this problem, we have introduced the Trainable Weka Segmentation (TWS), a machine learning tool that leverages a limited number of manual annotations in order to train a classifier and segment the remaining data automatically. The tool works interactively, allowing the user to guide the training by providing corrections to the classifier output. In addition, TWS can provide unsupervised segmentation learning schemes (clustering) for image data and can be customized to employ user-designed feature maps or classifiers.

The usefulness of the TWS tool has already been demonstrated by its utilization in a wide range of imaging pipelines that involve disparate segmentation tasks: analyzing wing photomicrographs (Dobens and Dobens, 2013), visualizing myocardial blood flow (Krueger et al., 2013), monitoring nests of bees (Hart and Huang, 2012), and other applications. TWS has proven useful for performing segmentation using many different image modalities. These include magnetic resonance imaging (Kulinowski et al., 2011), two-photon microscopy (Villa et al., 2013), serial-section transmission electron microscopy (Laptev et al., 2012), confocal fluorescence microscopy (Felcht et al., 2012; Frank et al., 2012; Crepaldi et al., 2013), micro- and computerized tomography (Maiora and Graña, 2012; Macdonald and Shefelbine, 2013), transmission scanning (Mathew et al., 2012), and angiography (Favazza et al., 2013).

Background

Traditional image processing methods

Image segmentation is generally defined as the process of partitioning a digital image into nonintersecting regions. These regions or segments comprise sets of pixels that share certain visual characteristics and are assigned a specific label. For instance, in the microscopic image of a cell, one could segment the different organelles and label pixels belonging to the nucleus, the mitochondria, and other structures. Similarly, in an image from a security camera, one might want to identify suspicious objects and separate them from the rest of the pixels. In the same example, a face recognition system may attempt to label the person or persons appearing in the image. Therefore, image segmentation can be regarded as an ill-defined problem since, depending on the final application, different ways of partitioning the same image can be considered correct. Hundreds of automatic and semiautomatic segmentation algorithms have been presented since the appearance of the digital image. However, no single method can be considered appropriate for all types of images. Moreover, methods that have been designed for a particular type of image might not be applicable to other types.

Most traditional methods are based only on the intensity information of pixels. Nonetheless, humans use much more knowledge when performing manual segmentation. For that reason, in recent years, trainable methods have emerged as powerful tools to include part of that knowledge in the segmentation process and improve the accuracy of the labeled regions. Algorithms to perform this task have been developed principally for natural and medical images but can be adapted for other types of image data and transferred to platforms that are accessible to both experienced and inexperienced users. Such software should provide a user-friendly and intuitive framework for prototyping and applying machine learning algorithms to image data and visualizing their results.

Platforms that build in machine learning tools

Just a few software platforms partially provide both machine learning and image processing tools. These include commercial platforms (e.g., MATLAB, MathWorks, Natick, MA) and open-source plaforms, e.g., the Konstanz Information Miner (KNIME) by Berthold et al. (2007), RapidMiner (http://rapid-i.com), Vision with Generic Algorithms (VIGRA) by Köthe (1999), and CellProfiler by Kamentsky et al. (2011). Commercial platforms usually target inexperienced users and a wide range of image types. However, the details of the algorithms are hidden, which is undesirable for use in scientific research. Conversely, those details are available in open-source platforms such as KNIME and RapidMiner, which is becoming the world-leading open-source system for data and text mining. Nevertheless, RapidMiner is developed primarily by the machine learning community for the machine learning community, and its image processing extension by Burget et al. (2010) provides only a minimal set of image tools. This makes the platform less attractive for computer scientists to use it to develop image segmentation solutions. Other projects like VIGRA offer powerful computer vision libraries with a focus on algorithms

NOTES

and data structures but no visualization tools or user-friendly interfaces. And only a development version of CellProfiler integrates VIGRA learning methods into custom segmentation pipelines.

To address these deficiencies in the field, we started the new open-source software project TWS. The project combines the image processing toolkit Fiji (Fiji Is Just ImageJ) by Schindelin et al. (2012), a popular distribution of ImageJ by Rasband (1997–2009), with the state-of-the-art machine learning algorithms provided in the latest version of the data mining and machine learning toolkit Waikato Environment for Knowledge Analysis (WEKA) by Hall et al. (2009). TWS provides a set of library methods for extracting statistical properties of an image from user-provided pixel samples and uses that information to segment the rest of the pixels in that image or a similar image. These methods are then implemented in a modular and transparent way and can be called up from any Fiji plugin, script, or macro. TWS also provides a friendly graphical user interface (GUI) for loading a two-dimensional (2D) image or image stack and performing automatic segmentation by interactive learning.

TWS came about through the perceived need of a general purpose workbench that would allow researchers from the imaging world to access state-of-the-art techniques in machine learning to improve their image segmentation results. This need was observed by Burget et al. (2010), who created an image processing extension for the popular data mining software RapidMiner. Following that innovation, Sommer et al. (2011) presented ilastik, an interactive, user-friendly tool for image classification and segmentation based on training a random forest classifier on precomputed nonlinear features. TWS provides both a set of library functions to design experiments and algorithms, based on the WEKA and Fiji platforms, and a complete GUI for performing interactive and noninteractive learning.

Enhancing Fiji and WEKA

In the past four years, Fiji has become the software of reference for many scientists to meet their image analysis needs, especially in the field of biomedical imaging. Fiji provides its users with powerful tools to generate sophisticated image processing pipelines and algorithms, via scripting languages and library methods that can handle many types and sizes of images. At the same time, WEKA is nowadays recognized as a landmark system in data mining and machine learning. It has achieved widespread acceptance within academia and business circles, and has become a widely used tool for data mining research.

However, little (if any) of the success of both toolboxes would have been possible if they had not been released as open-source software. Giving users free access to the source code has enabled a thriving community to develop and facilitated the creation of many projects that incorporate or extend WEKA's existing functionalities. One of the best examples of these projects is TWS, which combines both toolboxes to enlarge their capabilities and increase their impact and range of application. For WEKA users and developers, TWS offers transparent access to a whole new set of supervised and unsupervised learning problems based on an arbitrarily large number of image features. For Fiji users and developers, respectively, TWS provides a new and user-friendly way of performing image segmentation and facilitates access to learning tools that can be used to either enhance existing image processing algorithms and pipelines or create new ones.

Implementation

Computing environment

TWS has been developed using the developing (but stable) versions of Fiji and WEKA (version 3.7.6). The only requirement to use TWS is to have Fiji installed and up to date. In fact, given the cross-platform nature of Fiji, it can be run with any Unix, Macintosh, or Windows environment. The software is distributed as open-source software with a detailed user manual and multiple tutorials published in the Fiji wiki (http://fiji.sc).

For basic users, TWS requires only basic experience and knowledge of Fiji. The user should be familiar with the simple interface of Fiji and its plugin system. The TWS plugin can be run from the plugin menu under the segmentation submenu. The user can then interact with the GUI without the need for any other Fiji commands (a detailed explanation is given in the user manual). For more experienced users and developers, the TWS library methods are accessible from the scripting interpreters available in Fiji as well as from any third-party script or plugin.

Data input/output

When the plugin is called up from the Fiji menu, TWS runs on the current 2D (gray-scale or color) image or stack of images. If no image data are open, the plugin will ask for a file to load in any of the multiple image formats compatible with Fiji. The image feature information produced in the interactive learning procedure can be saved and loaded in the Attribute-Relation File Format (ARFF) (Witten and Frank, 2005), which can also be loaded and manipulated in the WEKA suite. The trained

models (classifiers and clusterers) can also be saved and loaded in a WEKA-compatible format (.model). This separation between feature data and models enables full compatibility with the WEKA tools and library methods. For instance, a user can save the feature data of an experiment using the plugin's GUI, then load it into the WEKA experimenter to find the most suitable classifier, and finally load the classifier back into the GUI to use it in an arbitrary number of images.

Image features

In computer vision, a feature is usually defined as the part of an image of special interest, and image features are used frequently as the starting point for many algorithms. Therefore, the overall algorithm will often only be as good as its feature detector. For that reason, TWS includes a wide range of image features, most of which are extracted using common filters or plugins available in the Fiji core. By default, more than 70 features are computed using generic parameters and spherical filters with radii varying from 1 to 16 pixels. The user has complete freedom to select features and tune their scales and optional parameters using either the settings dialog in the GUI or the specific library methods.

Based on their purpose, the features available in TWS can be grouped into the following types:

- ***Edge detectors***, which aim at indicating boundaries of objects in an image. TWS includes, among other edge detectors, Laplacian and Sobel filters, difference of Gaussians, Hessian matrix eigenvalues, and Gabor filters.
- ***Texture filters***, to extract texture information. Among others, TWS provides a set of filters including minimum, maximum, median, mean, variance, entropy, structure tensor, etc.
- ***Noise reduction filters***, such as Gaussian blur, bilateral filter, Anisotropic diffusion, Kuwahara, and Lipschitz.
- ***Membrane detectors***, which localize membrane-like structures of a certain size and thickness.

In addition to providing filters, TWS allows the user to customize features. As described in the wiki, a very simple script is needed to include user-defined features in the segmentation process. As long as they contain unique identifiers, an arbitrary number of new features can be used either alone or in combination with the existing filters. This opens the door to all kinds of linear and nonlinear features that users can externally create, including 3D features.

Segmentation by pixel classification

To segment the input image data, TWS transforms the segmentation problem into a pixel classification problem in which each pixel can be classified as belonging to a specific segment or class. A set of input pixels that has been labeled is represented in the feature space and then used as the training set for a selected classifier. Once the classifier is trained, it can be used to classify either the rest of the input pixels or completely new image data. The number and names of the classes, together with the desired learning algorithm, are defined by the user. All methods available in WEKA can be used. These include a large variety of supervised classification and regression algorithms and clusterers. For a complete list, visit http://wiki.pentaho.com/display/DATAMINING/Data+Mining+Algorithms+and+Tools+in+Weka . Figure 1 describes the pixel classification scheme of TWS.

One of the strengths of the TWS toolbox is that it allows different options to perform training and testing. One possibility is to use the GUI following an active learning approach. In a similar way to ilastik (the image classification and segmentation tool developed by Sommer et al., 2011), the user is allowed to interactively provide training samples while navigating the data, obtain on-the-fly test results on the loaded input image, and retrain the classifier as many times as needed. In this way, the user can fine-tune the parameters of the classifier and select labels until achieving satisfactory results. More classical (not interactive) approaches are also available via the library methods, allowing training on arbitrarily large labeled data.

Creating your own algorithms

The segmentation of image data typically requires applying a sequence of algorithms to many images in a so-called pipeline. In Fiji, fast prototyping of segmentation pipelines is facilitated via scripting, which uses simple programming commands (or scripts) to define sequences of operations that can be applied to sets of images.

Scripting became popular among inexperienced users thanks to the simple and friendly ImageJ macro language, which allows users to record GUI commands and rapidly construct basic programs. TWS is not an exception, and all the user actions in its GUI can be recorded and reproduced later in a macro. Additionally, Fiji supports a broad range of scripting languages, which can be used without knowledge of Java (the native language of Fiji and WEKA), and offers more advanced programming capabilities than the macro language while keeping

NOTES

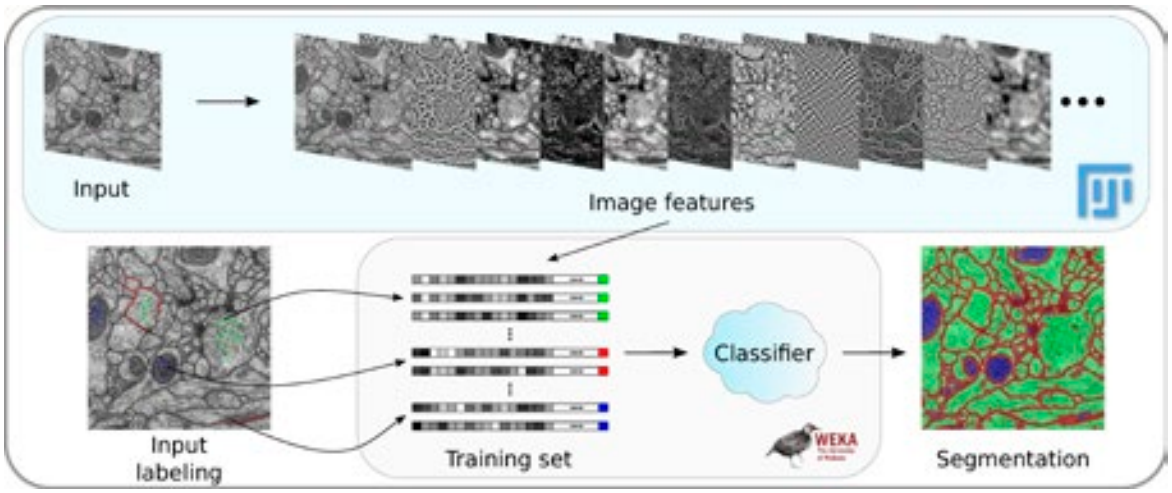

Figure 1. TWS pipeline for pixel classification. Image features are extracted from an input image using Fiji-native methods. Next, a set of pixel samples is defined and represented as feature vectors, and a WEKA learning scheme is trained on those samples and finally applied to classify the remaining image data. The input image in this example pipeline is a serial section from a transmission electron microscopy dataset from the *Drosophila* first instar larva ventral nerve cord; its pixels are divided into three classes: membrane, mitochondria, and cytoplasm.

relative simplicity for the occasional programmer. TWS makes extensive use of this functionality to provide an ideal framework to rapidly prototype and implement new algorithms. Individual commands can be interactively tested on the current images using the corresponding scripting language interpreter. In addition, the script editor enables writing, debugging, testing, and running arbitrarily complex scripts in all the supported languages, including Java itself. For example, it takes only 10 lines of code to load an image and its binary labels to train a classifier on how to identify cells, apply the trained classifier to a second image, and extract each individual cell in the image by running a watershed algorithm implementation (Tsukahara et al., 2008) on the cell probability map (Fig. 2).

The details of the code or the chosen scripting language are not relevant (TWS scripting tutorials are available at http://fiji.sc/Scripting_the_Trainable_Segmentation). All scripting languages in Fiji allow users to access advanced image processing libraries and now, thanks to TWS, to interact with the WEKA machine learning algorithms without mastering Java programming. Moreover, it is also possible to integrate this functionality through the Fiji interfaces in other image platforms such as MATLAB or ITK (Ibanez et al., 2003).

Conclusion

We have presented TWS, a versatile tool for image segmentation based on pixel classification. The software has a library of methods and a GUI that makes it easy to use without any programming experience. This toolbox is an important addition to the growing arsenal of segmentation plugins in Fiji (fiji.sc/Category:Segmentation) for analyzing biological and nonbiological image data. TWS is designed to facilitate the integration of machine learning schemes with image processing modules into a pipeline. Researchers can easily prototype segmentation algorithms using TWS methods with any of the scripting languages available in Fiji. Scripts are indeed vehicles of execution, but also act as mechanisms for disseminating the new algorithm or pipeline that are easily accessible by others.

TWS is intended to work as a bridge between the machine learning and the biomedical imaging communities, facilitating a framework to develop, test, and apply novel solutions to the existing segmentation challenges. The project is completely open source, and we invite users and developers to contribute to its growth.

a)

```
1  import trainableSegmentation.WekaSegmentation;
2  trainInput = IJ.openImage( "https://dl.dropboxusercontent.com/u/1958594/cells/images/A9p5d.tif" );
3  trainLabels = IJ.openImage( "https://dl.dropboxusercontent.com/u/1958594/cells/labels/A9p5d.tif" );
4  testInput = IJ.openImage( "https://dl.dropboxusercontent.com/u/1958594/cells/images/A9p9d.tif" );
5  segmentator = new WekaSegmentation( trainInput ); // create segmentator
6  segmentator.addRandomBalancedBinaryData( trainInput, trainLabels, "class 1", "class 2", 4000 );
7  segmentator.trainClassifier(); // train classifier based on input image and binary labels
8  probs = segmentator.applyClassifier( testInput, 0, true ); // apply classifier to test image
9  probs8bit = new ImagePlus( "Probs 8-bit", probs.getProcessor().convertToByte( true ) );
10 IJ.run( probs8bit, "Watershed ", "blurring=3.0'   watershed='1 0 40 80 0 0'   display='2 4' " );
```

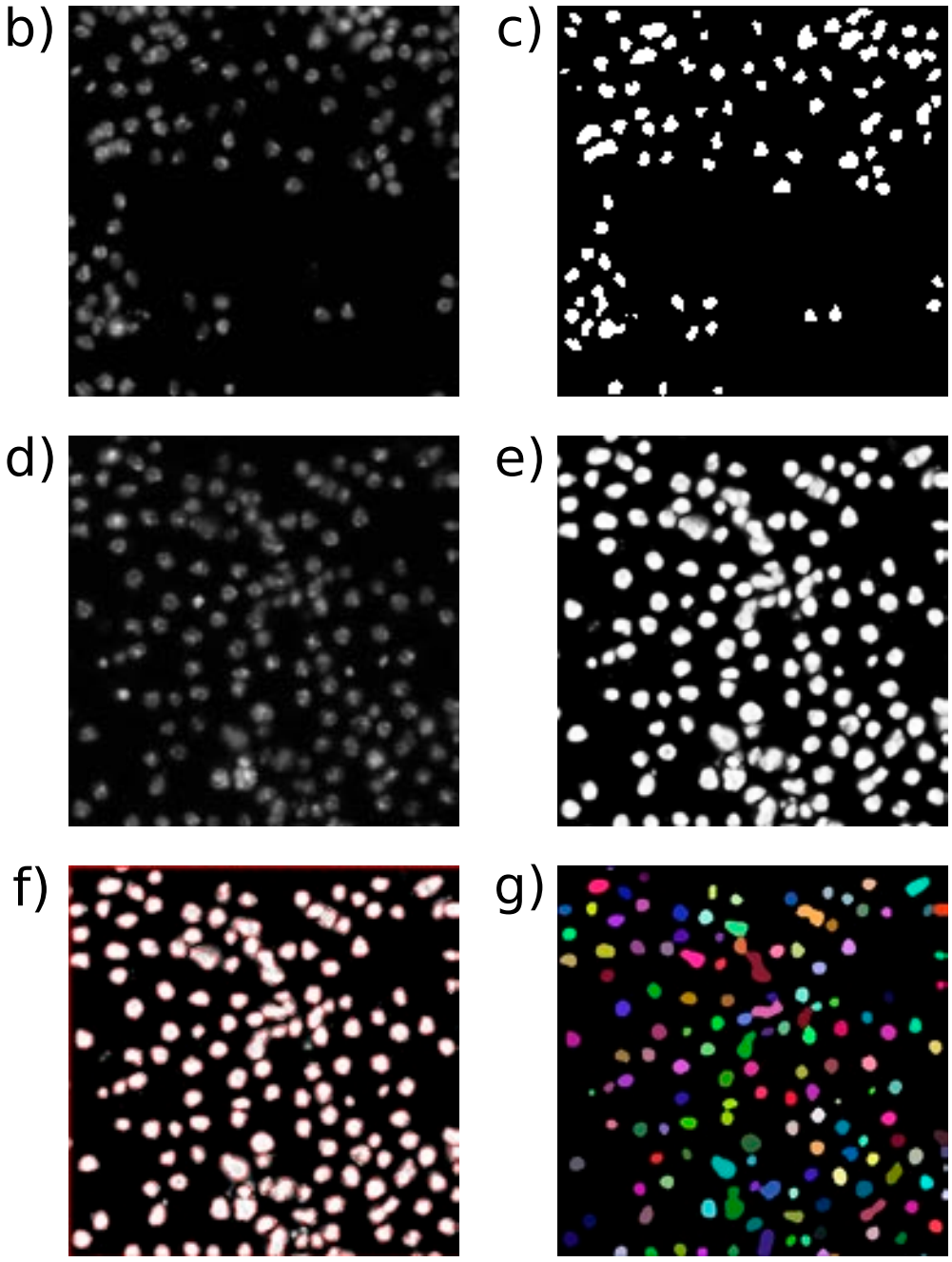

Figure 2. Scripting your own algorithm. *a,* Example of BeanShell script that trains on a cell colony image (*b*) from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012) and its corresponding binary labels (*c*) and then applies the trained classifier to a test image (*d*) to obtain its in/out probability map (*e*). The algorithm finally runs watershed segmentation to extract the boundaries of the cells (*f*) and labels the individual objects (*g*).

NOTES

References

Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Sieb C, Thiel K, Wiswedel B (2007) KNIME: The Konstanz Information Miner in studies in classification, data analysis, and knowledge organization (GfKL 2007). Berlin, Heidelberg: Springer.

Burget R, Karásek J, Smékal Z, Uher V, Dostál O (2010) RapidMiner Image Processing Extension: a platform for collaborative research. International Conference on Telecommunications and Signal Processing, Baden, Austria, August 7–10.

Crepaldi L, Policarpi C, Coatti A, Sherlock WT, Jongbloets BC, Down TA, Riccio A (2013) Binding of TFIIIC to SINE elements controls the relocation of activity-dependent neuronal genes to transcription factories. PLoS Genetics 9:e1003699.

Dobens AC, Dobens LL (2013) FijiWings: an open source toolkit for semiautomated morphometric analysis of insect wings. G3 (Bethesda) 3:1443–1449.

Favazza TL, Tanimoto N, Munro RJ, Beck SC, Garrido MG, Seide C, Sothilingam V, Hansen RM, Fulton AB, Seeliger MW, Akula JD (2013) Alterations of the tunica vasculosa lentis in the rat model of retinopathy of prematurity. Doc Ophthalmol 127:3–11.

Felcht M, Luck R, Schering A, Seidel P, Srivastava K, Hu J, Bartol A, Kienast Y, Vettel C, Loos EK, Kutschera S, Bartels S, Appak S, Besemfelder E, Terhardt D, Chavakis E, Wieland T, Klein C, Thomas M, Uemura A, et al. (2012) Angiopoietin-2 differentially regulates angiogenesis through TIE2 and integrin signaling. J Clin Invest 122:1991–2005.

Frank M, Duvezin-Caubet S, Koob S, Occhipinti A, Jagasia R, Petcherski A, Ruonala MO, Priault M, Salin B, Reichert AS (2012) Mitophagy is triggered by mild oxidative stress in a mitochondrial fission dependent manner. Biochim Biophys Acta 1823:2297–2310.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten I (2009) The WEKA data mining software: an update. SIGKDD Explor 11:10–18.

Hart N, Huang L (2012) Monitoring nests of solitary bees using image processing techniques. 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Auckland, New Zealand, November 28–30, 2012. IEEE Explore, pp. 1–4.

Ibanez L, Schroeder W, Ng L, Cates J (2003) The ITK software guide, Ed 2. Kitware Inc.

Kamentsky L, Jones TR, Fraser A, Bray MA, Logan DJ, Madden KL, Ljosa V, Rueden C, Eliceiri KW, Carpenter AE (2011) Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. Bioinformatics 27:1179–1180.

Köthe U (1999) Reusable software in computer vision. In: Handbook of computer vision and applications, Vol 3, Systems and applications. San Diego: Academic Press.

Krueger MA, Huke SS, Glenny RW (2013) Visualizing regional myocardial blood flow in the mouse. Circulation Res112:e88–e97.

Kulinowski P, Dorozynski P, Mlynarczyk A, Weglarz WP (2011) Magnetic resonance imaging and image analysis for assessment of HPMC matrix tablets structural evolution in USP apparatus 4. Pharm Res 28:1065–1073.

Laptev D, Vezhnevets A, Dwivedi S, Buhmann JM (2012) Anisotropic ssTEM image segmentation using dense correspondence across sections. In: 15th Annual Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2012), October 1–5, 2012, Nice, France; pp. 323–330. Berlin, Heidelberg: Springer.

Ljosa V, Sokolnicki KL, Carpenter AE (2012) Annotated high-throughput microscopy image sets for validation. Nat Methods 9:637–637.

Macdonald W, Shefelbine S (2013) Characterising neovascularisation in fracture healing with laser Doppler and micro-CT scanning. Med Biol Eng Comp 51:1157–1165.

Maiora J, Graña M (2012) Abdominal CTA image analisys through active learning and decision random forests: application to AAA segmentation. In: The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10–15, 2012. IEEE Explore, pp. 1–7.

Mathew MD, Mathew ND, Ebert PR (2012). WormScan: a technique for high-throughput phenotypic analysis of *Caenorhabditis elegans*. PloS One 7:e33483.

Rasband W (1997–2009) ImageJ. US National Institutes of Health, Bethesda, Maryland.

Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B, Tinevez JY, White DJ, Hartenstein V, Eliceiri K, Tomancak P, Cardona A (2012) Fiji: an open-source platform for biological-image analysis. Nat Methods 9:676–682.