

Overview

Cancer diagnosis is based upon taking tissue from a patient and identifying irregular cells. The practice of using the tissue for a diagnostic purpose is called pathology. Cells are translucent, so stain is utilized as a means to identify structures. Different cell organelles and structures uptake the stain differently. A popular stain used for pathology is Hematoxylin and Eosin (H&E).

The biggest revolution to occur in pathology is the advent of digital imaging, or the ability to use digital versions of these slides for diagnostic purposes. However, the intrinsic variability of biological tissue and the vast differences in tissue preparation protocols lead to significant image variability that can hamper the effectiveness of computational tools.

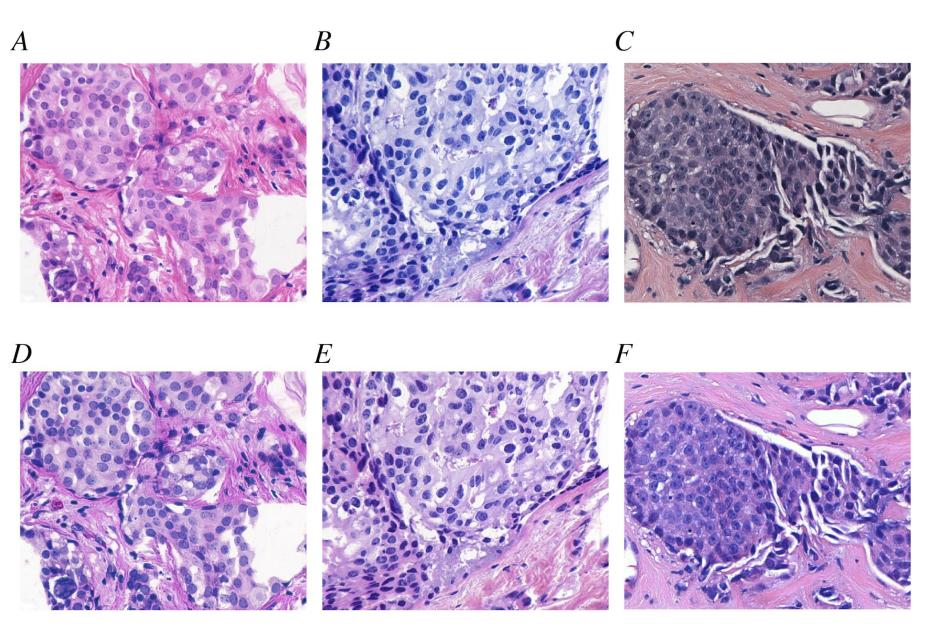


Fig 6. Color normalization for visualization from the 2017 Paper. (A,B) Two images from the UCSB data set exhibited different staining properties. A strong red hue was observed in (A) while the image in (B) had a greater proportion of blue. (C) Likewise, a representative image from the Drexel breast cancer databank exhibited yet a different color property. (D-F) Color normalization produced visually similar color

Our laboratory is interested in transformations to normalize pathology images for improved standardization and calibration in the anatomic pathology laboratory. At present, many algorithms have been developed, but there are limitations to their widespread usage, mainly due to documentation.

Key Concepts

Histology: The scientific study of the structure of tissue from plants, animals, and other living things.

Hematoxylin & Eosin: A stain that produces color variation in different tissue structures, which would otherwise be transparent.

Project Description

Dr. Zarella and Dr. Breen have developed algorithms to identify tissue structures in H&E stained histology images and to normalize the colors in these images. These algorithms accompany several published papers. We are working on several tasks in order to promote the usage of the work and aid pathologists and other researchers in the incorporation of these tools into current workflows and future research

- reproduce the original results
- updated the software to work with current MATLAB versions
- provide documentation and usage instructions for the algorithms
- integration the workflow with other pathology tools
- produce Python versions of the software

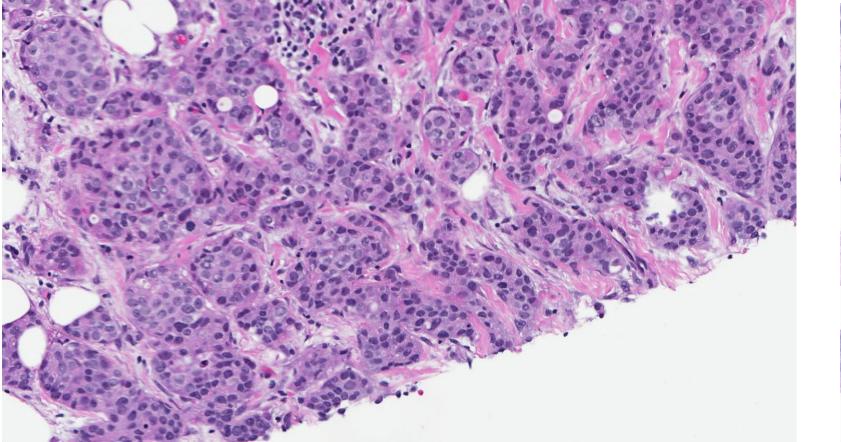
Increasing the usage of Color Normalization Algorithms in Histology

Jason DeFuria, Laurence Liss Faculty Advisors: Dr. Mark Zarella and Dr. David Breen

Shuyi Gu

Activities

The first step of the project was reproducing the algorithms to work using the latest version of MATLAB. One of the machine learning algorithms was completely removed, so it became necessary to verify results amongst multiple versions of MATLAB. Below are the results of the rewritten and verified color normalization algorithms:

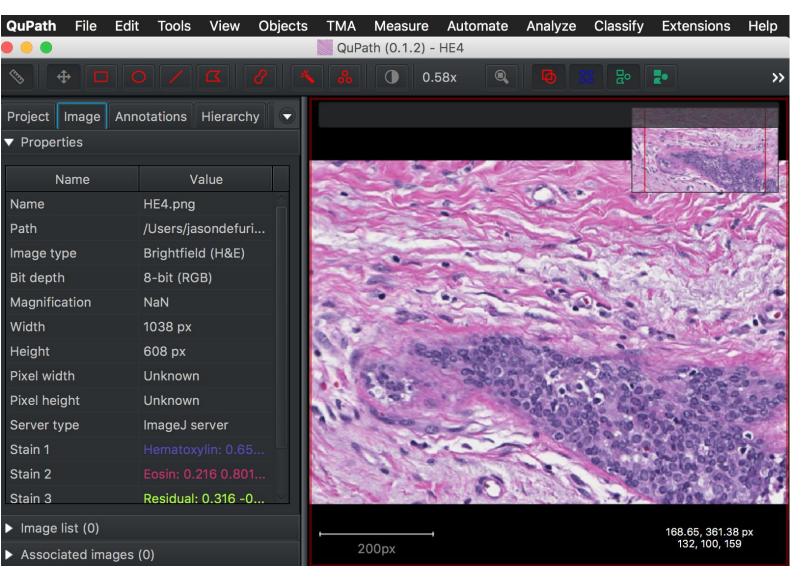




H&E Slide before Color Normalization

The project then veered off in two directions, but with the same end goal of increasing usage by making the algorithms more accessible.

Jason focused on integrating the algorithms with pathology software known as QuPath. QuPath is an open-source pathology analysis tool that is gaining momentum in the digital pathology world. To achieve this, he wrote QuPath scripts into a workflow so that users of this popular digital pathology tool would be able to easily use the color normalization scripts and output a color normalized image.



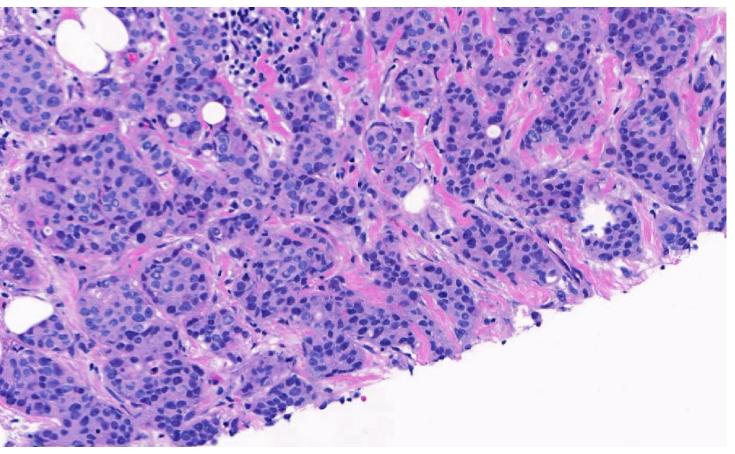
QuPath Interface

Python is one of the most widely used languages for scientific computing, data analysis, and machine learning. Laurence focused on reproducing the software in Python to allow experimentation and investigation by those without access to MATLAB. The goal was to allow new students and programers to quickly begin using and adapting the programs.

The entire process and the inner workings of each program were documented to a website hosted on the Drexel School of Medicine domain.

College of Medicine	LUTDE	
ACADEMICS ADMISSIONS Home / About / Departments, Centers, Institute:	S And Offices / Department Of Pathology & Laboratory Medicine / Zarelia Lab / H&E Normalization / Instructions	Home / Abou
ABOUT	H&E HISTOLOGY IMAGE NORMALIZATION	ABOUT
Overview	USAGE INSTRUCTIONS	Overview
Message from the Dean	These instructions intend to serve as a basis for use of the color normalization functions in the hopes that they might be useful in diagnostic settings. For detailed information on the methods used to produce the results, please see the detailed documentation.	Message from
Mission & Vision		Mission & Vis
Diversity, Equity & Inclusion	MATLAB VERSIONS 📣	Diversity, Equ
Departments, Centers, Institutes and Offices	Download MatLab code as zip file	Departments
Department of Pathology & Laboratory	colorassign_manual.m train_classifier.m	Departmer
Medicine	■ color_classify.m	Medicine
Faculty	color_normalize.m	Faculty
Related Programs & Courses	PYTHON VERSIONS 🥐	Related
Research	The Python versions of these programs are a work in progress and are not production ready at this time.	Resear
Zarella Lab		Zarella
H&E Normalization	Download Python code as zip file colorassign_manual.py	H&E
Instructions	train_classifier.py	
Documentation	color_classify.py	In
Advanced Pathology Imaging Laboratory	color_normalize.py	D
Electron Microscopy Laboratory	QUPATH INTEGRATION	Advanc
Molecular Diagnostics Laboratory	We have created a workflow, in which color normalization will be run from QuPath, calling on MatLab to do the calculations and corrections.	Laborat
	Normalize an image based on a selection from QuPath	Electro
Pathology Diagnostics Laboratory	QUICK USAGE EXAMPLE IN MATLAB	Molecu
Regional Autopsy Service	>> sampleImage = imread('/path/to/histology/sample_image.tif')	Patholo
Tissue Procurement Facility	>> fullImage = imread('/path/to/histology/full_image.tif') >> [structureMap. lumen. nuclei. stroma. cvtoplasm] = colorassign manual(loadedImage)	Porior

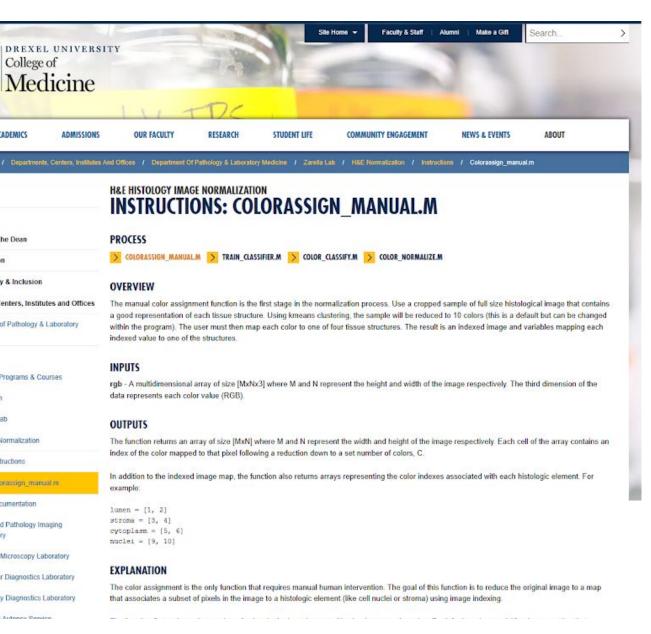




H&E Slide after Color Normalization

Properties Name Value Name HE4.png Path /Users/jasondefuri Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Scripts 061 062	QuPath File B	Edit Tools View Objects TMA Measure Automate Analyze Classify Extensions He QuPath (0.1.2) - HE4
Properties Name Value Name HE4.png Path //Users/jasondefuri Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Script editor Scripts 062 Odd 062 OutHtHed 1 062 Odd 064 OutHtHed 1 065 Off 064 0001 OutHtHed 1 066 067 OutHtHed 1 066 067 0014 OutHthe 1 066 067 0014 0014 OutHthe 1 066 067 0014 0014 0014 OutHthe 1 066 070 def imageData = dpP.getCurrentImageData () 0014	S 🕈 🗖	O/ 🕰 🥜 🔥 👶 0.58x 🔍 🗔 🔀 🛃 🄁
Name Value Name HE4.png Path /Users/jasondefuri Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Script editor Scripts 062 Untitled 1 063 066 // Import the helper class 066 066 067 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQuI 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine (this) try { def imageData = QP.getCurrentImageData() 064 try { 065 if (roi != null) { 071 if (roi != null) { 072 if (roi != null) { 073 if (roi != null) { 074 if (roi != null) { 075 img = imageData.getServer().getBufferedThumbnail (1000, -1, 0)	Project Image	Annotations Hierarchy 🗨
Name HE4.png Path /Users/jasondefuri Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Scripts 061 062 063 // Import the helper class 064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQuI) 065 066 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine (this) 068 try (069 071 064 ouble downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) (075 img = imageData.getServer().readBufferedImage(RegionRequest.createInstructure) 076 else 077 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	 Properties 	
Path /Users/jasondefuri Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Scripts 062 Untitled 1 062 064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu1 065 // Get the MATLAB engine 066 try { 067 QuPathMATLAB.getEngine (this) 068 try { 069 def roi = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInstructure) 076 j else 077 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	Name	Value Value
Image type Brightfield (H&E) Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Script editor Scripts 061 062 063 064 004 065 Untitled 1 063 064 004 065 066 065 066 066 066 066 07 0uPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu1 066 066 066 07 000 066 07 000 066 07 000 066 07 000 066 07 00 066 07 00 07 07 07 07 07 07 07 07 07 07 07	Name	HE4.png
Bit depth 8-bit (RGB) Magnification NaN Width 1038 px Script editor Scripts 061 062 Untitled 1 QuPathColorNormaliza 063 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu) 065 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine(this) 068 try { 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInteredImage) 078	Path	/Users/jasondefuri
Magnification NaN Width 1038 px Script editor Scripts 061 062 063 // Import the helper class 064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu) 065 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine(this) 068 try { 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInt) 076 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	Image type	Brightfield (H&E)
Width 1038 px Script editor Scripts 062 Untitled 1 063 // Import the helper class OuPathMatLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQuI) 066 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine (this) 068 try { 069 061 imageData = QP.getCurrentImageData() 070 def inageData = QP.getSelectedROI() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInst) 076 j else 077 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	Bit depth	8-bit (RGB)
Script editor Scripts Untitled 1 QuPathColorNormaliza 062 063 // Import the helper class 064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQuD 065 066 // Get the MATLAB engine 067 069 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInst) 078 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	Magnification	NaN
Scripts PUntitled 1 QuPathColorNormaliza 063 064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu) 065 066 067 QuPathMATLAB engine 067 068 try { 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { img = imageData.getServer().readBufferedImage(RegionRequest.createInst 078 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)	Width	
Scripts 062 'Untitled 1 063 QuPathMaTLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu1 065 066 067 QuPathMATLAB.getEngine 067 068 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createInst) 076 j else 077 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0)		
	*Untitled 1 QuPathColorNorma	<pre>064 QuPathMATLAB = this.class.classLoader.parseClass(QuPathMATLABExtension.getQu 065 066 // Get the MATLAB engine 067 QuPathMATLAB.getEngine(this) 068 try { 069 070 def imageData = QP.getCurrentImageData() 071 double downsample = 1 072 BufferedImage img; 073 def roi = QP.getSelectedROI() 074 if (roi != null) { 075 img = imageData.getServer().readBufferedImage(RegionRequest.createIm 076 } else 077 img = imageData.getServer().getBufferedThumbnail(1000, -1, 0) 078</pre>

Example QuPath Region of Interest Selected with Script Window



Jason:

In my Biology classroom, I will be sharing this algorithm with my students to demonstrate the cell cycle and the process of detecting cancerous cells. It will hinge upon the importance of being able to standardize imagery and the fact that the scientific method is meant to be replicated.

In my Forensics classroom, I will be adding a digital analysis portion to my crime scene unit. I will be filling a few virtual hard drives with data, and students will have to develop an algorithm for sorting the files to be able to determine which are important for discovering the murderer. The victim of the murder will be a pathology student, and images will be hidden in irregular nuclei in the victim's computer.

Laurence:

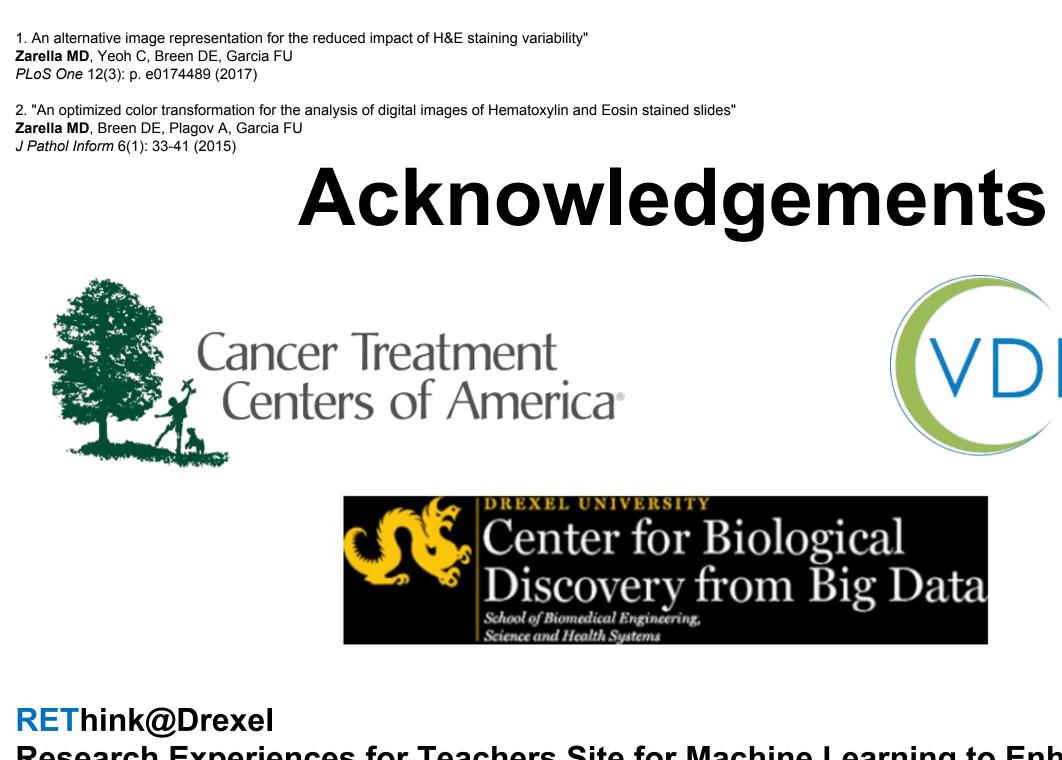
I intend to produce data sets that can be used with machine learning approaches such as kmeans and SVM in order to demonstrate training an algorithm and using the results to classify further data. In this way, students will have a chance to see that the field (like much of the rest of computer science) is in the understanding and correct application of established tools and principles.

Jason:

Currently, QuPath will export the normalized image. I investigated adding the image back to QuPath as an overlay, but unfortunately time is a limiting factor. A further goal would be to remove the dependence on MATLAB. This could be accomplished by either rewriting the scripts in Python or by writing native extensions that can be installed within QuPath. In my classroom, I am also hoping to eventually add an image analysis unit to my Forensics class, specifically of various fibers and hairs with digital image analysis. I need to spend some time refining and researching an appropriate way of incorporating this.

Laurence:

I have successfully reproduced one of four MATLAB functions needed to complete the color normalization of an histology image. Future work would include the completion of the remaining three functions. Additionally, the first stage of the normalization process relies on a manual process to identify tissue structures. Ideally, this identification would be completed algorithmically. Machine learning techniques would be a good fit for this type of problem and the automation of the process would be both a non-trivial and an extremely useful feature.



Research Experiences for Teachers Site for Machine Learning to Enhance Human-Centered Computing

. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1711773 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Course Materials

Future Work

References



Informatics

