Nile Red Kidney Segmentation/Quantification Manual

Adrienne Kline, MD, PhD

July 21, 2021

Contents

1 Introduction **4**
- 1.1 Motivation for work . . . 4
- 1.2 Kidney Biopsy with Nile Red staining . . . 4
- 1.3 Segmentation . . . 5

2 Methodology **6**
- 2.1 Image sets used in development . . . 6
- 2.2 Image Pre-processing . . . 6
- 2.3 Glomeruli Segmentation . . . 9
- 2.4 Tubules Segmentation . . . 10
- 2.5 Interstitium Segmentation . . . 11
- 2.6 Lipid Quantification . . . 11

3 Deployment w.r.t the demo code **14**
- 3.1 About and when to use . . . 14
- 3.2 Running/Loading in a pair of images . . . 14
- 3.3 Specifying glomeruli center . . . 15
- 3.4 Specifying glomeruli size . . . 17
- 3.5 Tubule sub-sample . . . 18
- 3.6 Active contours . . . 19
- 3.7 Tubules . . . 22
- 3.8 Interstitium Segmentation . . . 24
- 3.9 Lipid Quantification . . . 26
- 3.10 Output . . . 29

4 Deployment w.r.t. batch processing code **32**
- 4.1 Pipeline code - about and when to use . . . 32
- 4.2 Loading in files . . . 33
- 4.3 Naming files . . . 33
- 4.4 Maindata . . . 35
- 4.5 Outputs . . . 35

5 Final Comments **37**

6 Repo and Additional information **38**
- 6.1 Github: . . . 38
- 6.2 Contact info . . . 38

List of Figures

1 Nile Red (691) image with labeled structures 7
2 Glomeruli segmentation 9
3 Tubule segmentation 10
4 Interstitium Segmentation 11
5 Loading in image pair 15
6 Select Glomeruli Centers 16
7 Glomeruli diameter relative to image 17
8 User defined tubule sub-sample 18
9 Active Contours input image, intitial contour and final output 19
10 Active contours glomeruli mask 20
11 Final glomeruli mask 20
12 Glomeruli boundary - segmentation viewing 21
13 Initial tubule mask (threshold based) 22
14 Final tubule mask - closing and area filtering 23
15 Tubule bounding segmentation shown on RGb structural image 23
16 Interstitium mask final 24
17 Interstitium bounds on RGB structural image 25
18 Binary Lipid droplet image 26
19 Lipid droplets within the glomeruli mask 27
20 Lipid droplets within the tubule mask 27
21 Lipid droplets within the interstitium mask 28
22 Workspace variables - Matlab 30
23 Code that defined Excel output 30
24 Excel Output 31
25 File type to be loaded in 33
26 Naming for processing 34
27 Workspace of Pipeline 35
28 Workspace of Pipeline 36
29 Excel Output 36

List of Algorithms

1 Segmentation Code Overview 13

1 Introduction

1.1 Motivation for work

Most segmentation of histopathogical images have been driven by deep learning supervised segmentation algorithms. This feat requires manual or semi-manual segmentation of structures and subsequent learning of the structures, often requiring thousands of training images and immense computational resources to develop. Here I will outline a methodology for the application of segmenting Nile Red stained kidney images segmentation for purposes of biological quantification. As an alternative to manual or deep learning particularly for small datasets I have implemented a semi-automated algorithm making use of an active contours segmentation algorithm and a series of other classical image processing approach to isolate the glomerulus, tubules, interstitium and background.

1.2 Kidney Biopsy with Nile Red staining

Renal biopsy interpretation remains the gold standard for the diagnosis and staging severity of many kidney diseases [1]. Visual morphological assessment of the renal parenchyma provides useful information for disease categorization by pathologists. Current kidney biopsy tissue preparation for light microscopy, immunohistochemistry and electron microscopy are complex, time consuming and technically challenging. Nile Red (9- diethylamino-5Hbenzoαphenoxazine-5-one) is a fluorescent lipophilic dye that can be used as a fast staining alternative for structural investigation [2],[3]. Comparing morphology, composition and distribution of structures in various kidney compartments (glomerulus, tubule, and interstitium) across different disease states using Nile Red may provide mechanistic insight into their prognostic value of evaluating patients with kidney disease. However, such an analysis requires reproducible identification and segmentation of the different kidney compartments, a time-consuming process limited by poor intra- and inter-reader manual assessment of renal biopsy specimens [4],[5].

1.3 Segmentation

Partitioning a digital image has been applied in very diverse fields of study such as area detection in satellite images, traffic control, autonomous cars, medical imaging,face/iris recognition and object detection/classification. To our knowledge, it has not been used, in segmenting Nile Red stained renal biopsy specimens. There are currently several options for segmentation including region-based, edge detection, clustering, mask R-CNN (regional convolution neural network), and fast marching [6]. Within these options several techniques are subsumed. Region-based utilizes objects based on threshold value(s). This method works well when high contrast exists between the foreground and background, and has the advantage of efficient calculations. However, region based is limited when the foreground and background are homogeneous. Edge detection (eg. water shedding) uses discontinuous local change within an image to define boundaries. However it is not suitable when there is an overabundance of edges in the image even after applying filters. Clustering (eg. K-mean) divides the image into a user defined 'n' number of homogeneous areas based on distances and works well with small data sets. Draw backs to this approach are computation time and that distance algorithms are not suitable for non-convex clusters. While R-CNN models entail high training time require many samples,they are most generalizable thereafter and have been and havebeen used in kidney pathologic segmentation [7],[8]. A fast-marching method (eg. Chan-Vese) finds a segmentation that optimizes an energy functional. A limitation is that it is highly susceptible to starting position. However, it can be used for regional or global segmentation. Our images contain an abundance of edges, poor contrast between distinct morphology, and too few samples on which to train a neural network adequately. Hence, we selected a combination of Chan-Vese and a regional based semi-supervised approach thus benefiting from Chan-Vese generalizability while retaining the benefit of user input. The latter alleviates the issue of starting position and thus allows for multiple glomeruli location(s) and fluctations in image scale (size of glomeruli).

2 Methodology

2.1 Image sets used in development

Images were acquired from human healthy normal nephrectomy samples and patients with diabetic renal pathology (society classification class 3) acquired from the Biobank for the Molecular Classification of Kidney Disease (PMID: 28747168). Usage of human samples were conducted in accordance with guidelines set forth by the Research Ethics Board at the University of Calgary and Alberta Health Services. Samples were fixed with 4% paraformaldehyde for 10 minutes followed by staining with Nile Red for 10 minutes, using a Zeiss LSM 880 confocal microscope with a 20X objective, NA 0.8 equipped with a 24-channel detector (485nm to 691nm) for spectroscopy. Specifically channel 691 was used to capture structural information for purposes and 548 for the lipid droplet channel. The image set included diabetic disease states, healthy normal controls, and glomerularnephritis states. Figure 1 below shows an overexposed normal human kidney tissue specimen processed with Nile Red stain. This example shows two glomeruli - denoted with 'G', multiple tubules are also visible, some labeled 'T', and interstitium 'I' which is non-glomeruli or tubule nor background.

2.2 Image Pre-processing

Images used in development were in '.tif' format however this method is generalizable to other image file types. They were processed in Matlab 2020b. Kidney biopsy images were loaded into the Matlab environment as Red Green Blue (RGB). Prior to the algorithm running, the user completes three tasks: 1) selecting the center of all glomeruli in the image by double clicking, 2) drawing a ROI around a portion of a tubule (inclusive), and 3) defining the size of the glomeruli relative to the image in decimal format to 3 decimal places. RGB images were subsequently coverted to greyscale using a luminosity conversion (Equation 1).
A 2-Dimensional Guassian filter (Equation 2) was applied to the images with with an adaptive kernel based on the size of the glomeruli relative to the image size.
Lastly, background was isolated using a threshold based on the Guassian processed images.

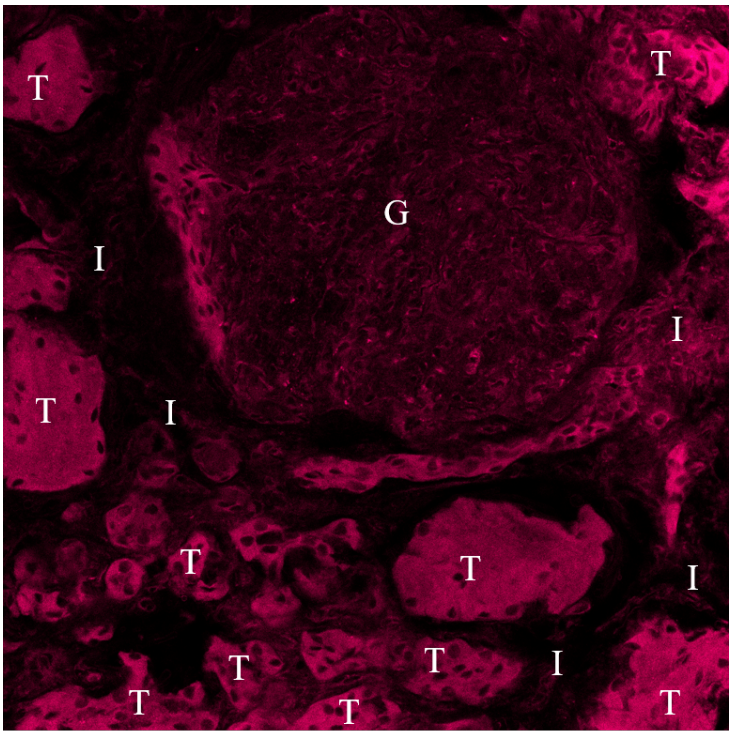

Figure 1: Nile Red (691) image with labeled structures

Background is established as any pixel from the gray scale and Guassian filtered image as having a value less than or equal to 2.

$$rgb2gray = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{1}$$

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+Y^2}{2\sigma^2}} \tag{2}$$

2.3 Glomeruli Segmentation

Both the location and size of glomeruli established in the preprocessing phase provide a preliminary starting mask in the shape of a circle of the glomeruli from which a Chan-Vese (active contours) optimization algorithm (Equation 3) finds a local minimum of entropy over a set number of iterations, where $H(\phi)$ is the Heaviside equation, $u_0(x, y)$ is the input image, and c_1, c_2, ϕ are updated recursively. The number of times the user selects center(s) of glomeruli also serves as a counter for the number of times the algorithm loop over the same image and run a glomerulus segmentation centered at that point. Following this, a series of image morphological processing occurs including closing, erosion and dilation as area filtering steps, these are outlined in Equations 4-6 respectively. This is then followed by background subtraction.

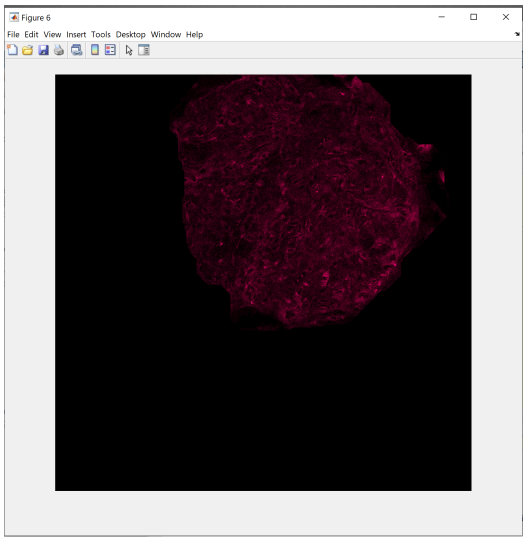

Figure 2: Glomeruli segmentation

$$
\begin{aligned}
F(c_1, c_2, \phi) = &\int_{\Omega} (u_0(x, y) - c_1)^2 H(\phi)\, dxdy \\
&+ \int_{\Omega} (u_0(x, y) - c_1)^2 (1 - H(\phi))\, dxdy \\
&+ v \int_{\Omega} |\nabla H(\phi)|
\end{aligned}
\tag{3}
$$

$$
A \bullet B = (A \oplus B) \ominus B \tag{4}
$$

$$
A \ominus B = \bigcup_{b \in B} A_b \tag{5}
$$

$$A \oplus B = \bigcap_{b \in B} A_{-b} \tag{6}$$

2.4 Tubules Segmentation

The tubule sub-sample drawn using the freehand ROI tool serves to create a mask that is applied to the gray scale structural image to create a distribution of pixel intensities within that sub-region. This was performed as intensities of the tubules w.r.t. insterstitium intensity varied throughout the images captures and by implementing this small user driven task without which would have necessitated the collection of all specimens at the same image parameters. This allows the user to bypass this. From the distribution defined by the ROI we calculate the mean and standard deviation (remembering to by this calculation by 255 to bring the range between 0 and 1. The calculated standard deviation is then halved and a threshold for tubules is defined as having a value greater than or equal to half a standard deviation below the mean of the tubules.

Using the inverse mask of the addition of the glomeruli mask and the background mask, we are able to isolate from the gray scale Guassian structural image which now is compromised of an image of both tubules and insterstitium. Apply the threshold as defined previously, we are able to isolate the tubules and form a tubule only mask. Following this, again a series of morphological closing and area exclusion are performed to include some of the darker spots within the tubule.

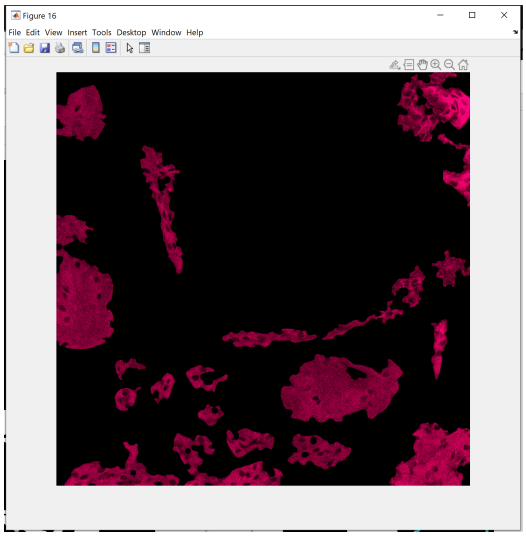

Figure 3: Tubule segmentation

2.5 Interstitium Segmentation

Interstitium was defined last (Fig. 4). By the addition of the glomeruli, tubules and background mask we can isolate all the structures which are now ¬interstitium. What remains after applying the inverse masks of the other 3 structures (glom, tubule and background) will be allocated to the interstitium class. This process is carried out through equation 7 and 8 below.

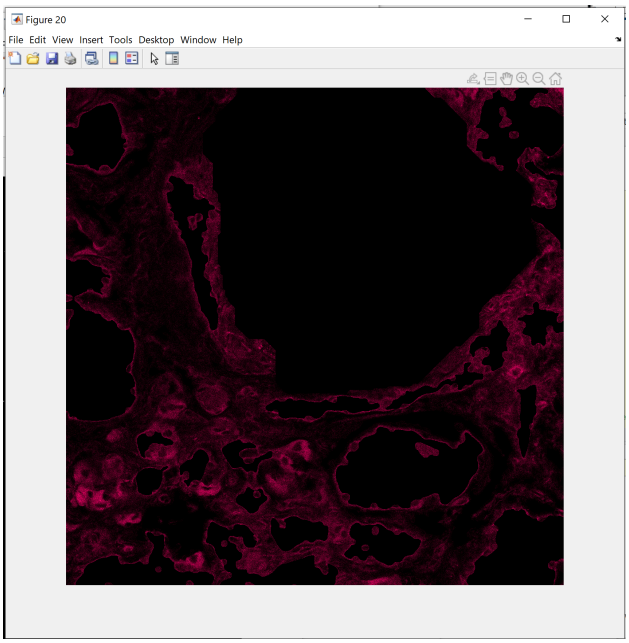

Figure 4: Interstitium Segmentation

$$\neg interstitium = G_{mask} \wedge T_{mask} \wedge B_{mask} \tag{7}$$

$$interstitium = f^{-1}(\neg interstitium) \tag{8}$$

2.6 Lipid Quantification

The RGB lipid channel image is converted to gray scale as in equation 1, and then binarized with a threshold of 0 so we include all lipid droplets

contained within the image. Lipid quantification is performed by applying the 3 masks created in the segmentation algorithm to the lipid (548 spectra) image. Quantification of lipid droplets within each of these are calculated based on area. The lipid droplets within each of the 3 bounding regions (tubule, glom and interstitium) is how we will calculate structure specific lipid droplet information. The antecedent in the ratio is area of lipid droplet (where the lipid binary image = 1), and the consequent is the total area of each bounding (structural) region - glom, tubule, interstitium. The lipid ratio is reported a decimal, not percent. Based on the script a name is created for each file and written to excel where it should appear in the same folder as where the main script, functions and image samples are stored and being run from.

$$Lipid_{ratio} = \frac{A_{lipid}}{A_{structure}} \tag{9}$$

A pseudo-code outline of the process is described in Algorithm 1.

Algorithm 1: Segmentation Code Overview

Result: Glomeruli, Tubulese, Interstitium
gnum, gsize, tubule$_{thresh}$← User defined
for $j = 1 : numberofimages$ **do**
  $j \leftarrow$ `rgb2gray(`j`)`, $0.2989R + 0.587G + 0.114B$
  $j \leftarrow$ `guassfilt(`j`)`, $G(x, y) \otimes j(x, y)$
  for $i = 1 : len(gnum)$ **do**
    glom-mask←(glom-mask=[])+`Chan-Vese(`$j(i)$`)`
  end for
  j ← j - glom-mask
  if $j \geq tubule_{thresh} - 0.5 * tubulesample_{\sigma}$ **then**
    tubule-mask ← j
  else
    interstitium-mask ← j
  end if
  for $struct = glom, tubule, intersittium$ **do**
    Lipid$_{ratio} = A_{lipid}/A_{struct}$
  end for
`writecell`(Lipid$_{ratio}$)
end for

3 Deployment w.r.t the demo code

3.1 About and when to use

This code is for processing a single pair of structural (691) and lipid (548) spectra channels. The purpose of the demo code is to showcase all stages of the processing for user need/investigation where you are interested in saving the images from different stages of the output, or trialing various starting glomerulus size/starting locations.
This lipid quantification will still be done per structure type and output to an excel spreadsheet with the name of the file being a concatenation of sample number and file type. The files you wish to process for lipid quantification should be pairs of files and have the name structure: patientnumber_withinpatientsample_lipid.tif equivalent to the lipid channel being used and patientnumber_withinpatientsample_struct.tif to represent the structural (691) channel being used. This is important for the name of the files as they are processed so the 691 ad 548 channels of the same sample can remain affiliated as well as save to an excel spreadsheet with the associated file name based on the input data. Should you wish to vary this you will need to alter the code accordingly.
If the user is unsatisfied with the segmentation of the glomerulus and other structures, advise starting the changing size of the glomerulus. The new 'run' of the code will overwrite the old file with the same name. It is possible to run all files through the demo code one at a time by changing the input files at the beginning part of the script. Output files will continue to save based on the file name input.

3.2 Running/Loading in a pair of images

You will need to load in the 2 files by name in the first part of the script. As seen here, the purple colored text corresponds to the file name and the quotes indicate that this is a string. The computer will match this to the file located in the same directory you are working in. `lipid` denotes the lipid image to be quantified and `structureFile` indicate the 691 or structural channel used for the segmentation aspect of the code.

```
%% INPUT LIPID FILE
lipid = imread('sample13011_a_lipid.tif'); % load in corresponding 548 (lipid) cha
figure(1)
imshow(lipid)
%% INPUT STRUCTURAL IMAGE
%struct = imread('19436_glom2_all_struct.tif');
structureFile = dir('sample13011_a_struct.tif'); % load in corresponding 691 (stru
```

Figure 5: Loading in image pair

To run the code click the green play button in the editor part of the menu.

3.3 Specifying glomeruli center

Next, a structural image will automatically appear based on the image input (691 spectra) channel and your cursor will look like a cross hair. The title at the top of the image will indicate what now must be done. Select the center of each glomeruli, as I have coded it to accept and loop through multiple if applicable. Keep in mind you need to double click in order to close this pop up window so if there is only one glomeruli in the image, then you will need to double click the center of this to both select the glom and simultaneously close the window. The title at the top of the figure will indicate what task you need to perform. As seen in Fig. 6 the title indicates: *Select all glomeruli centers.*

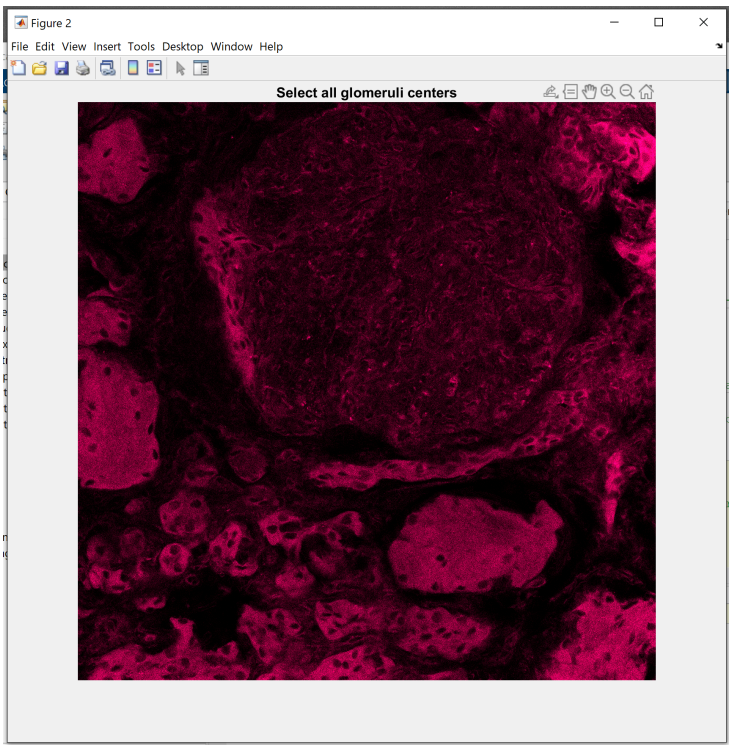

Figure 6: Select Glomeruli Centers

3.4 Specifying glomeruli size

Next, the same structural image will automatically appear based on the image input (691 spectra) channel and a dialogue box pop-up provides you with a prompt in order to indicate what must be done. You must enter the size of the glomeruli as a decimal format, erring on the most conservative side (if ellipsoid) relative to the height of the image. My advice here is to play around with sample images to see how starting size and location affect your segmentation quality. Active contours is a powerful segmentation algorithm and is robust against variations in image type but is limited by staring position. This starting position will form an initial circular mask which through an iterative entropy minimization algorithm will adjust the bounding edges accordingly.

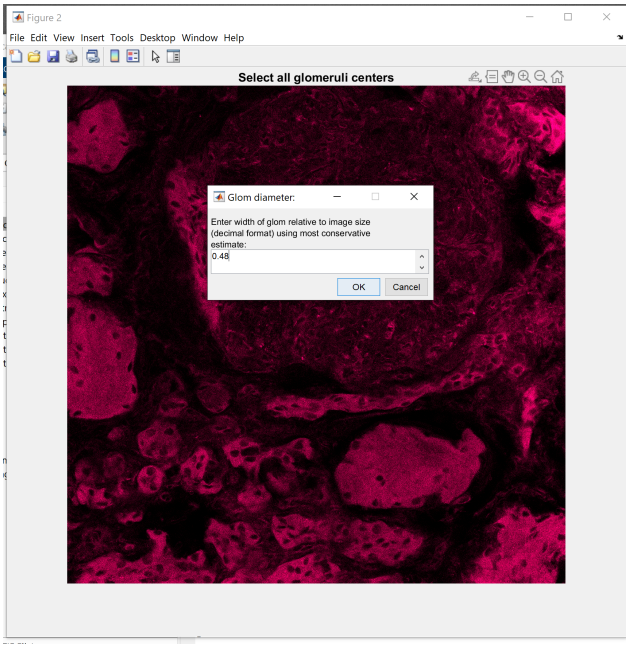

Figure 7: Glomeruli diameter relative to image

3.5 Tubule sub-sample

The last task that makes this code a semi-automated segmentation and lipid quantification code is defining a sub-sample of tubule which will serve to define a threshold for the average tubule intensity.

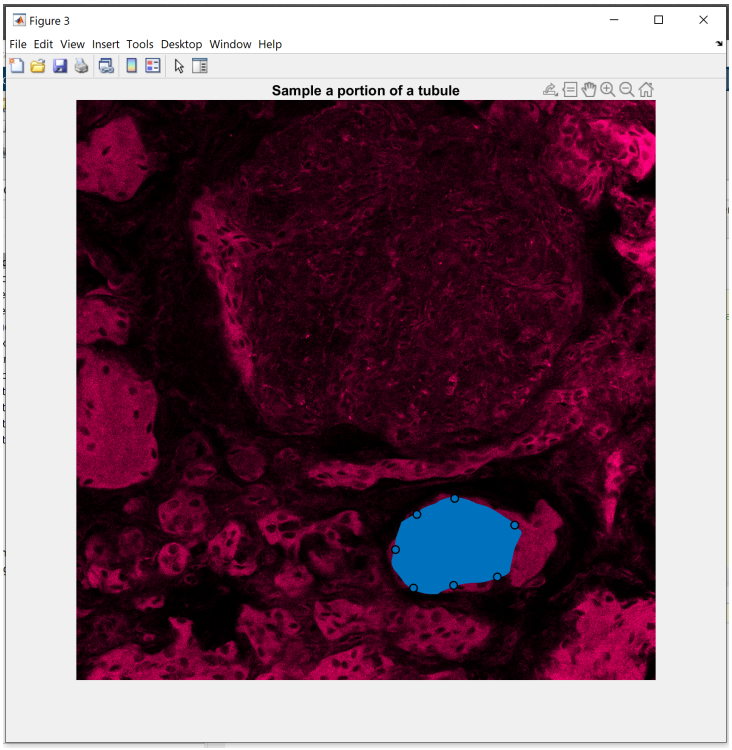

Figure 8: User defined tubule sub-sample

3.6 Active contours

Next you will see a pop up window that will show you the iterating process of the active contour based segmentation of each glomeruli. For images with multiple glomeruli selected you will see this process occur as many times as there are glomeruli in the image (Fig. 9). a larger scale version of the glomeruli mask as output by the active contours segmentation can be viewed in Fig. 10. Additionally, thorough a series of mask opening, dilation and erosion occur to create boundaries for each glomeruli, the output of which is seen in Fig. 11. Figure 12 places the boundary as predefined by the binary mask in Figure 11 and superimposes it onto the RGB structural image. This allows you to assess the quality of segmentation performed.

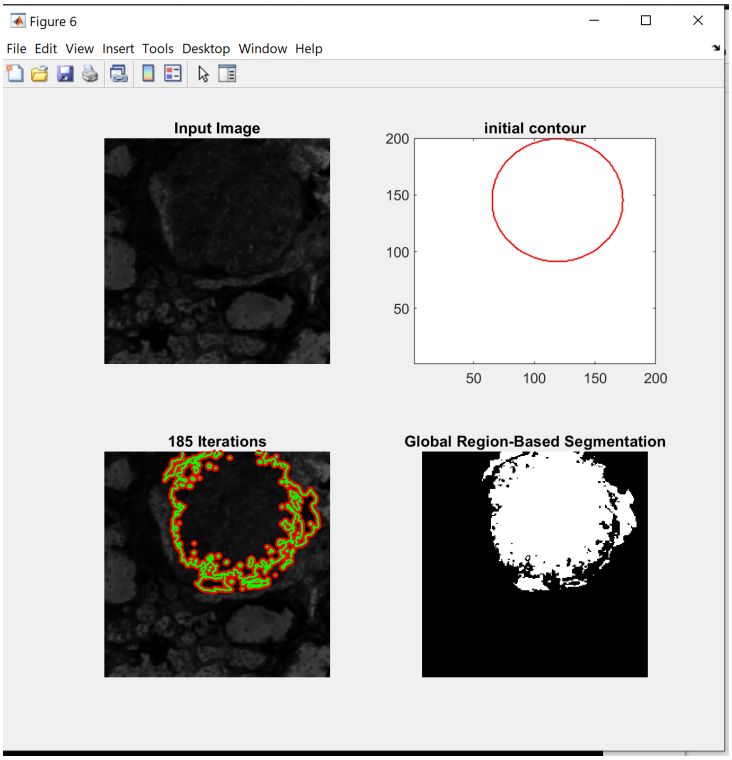

Figure 9: Active Contours input image, intitial contour and final output

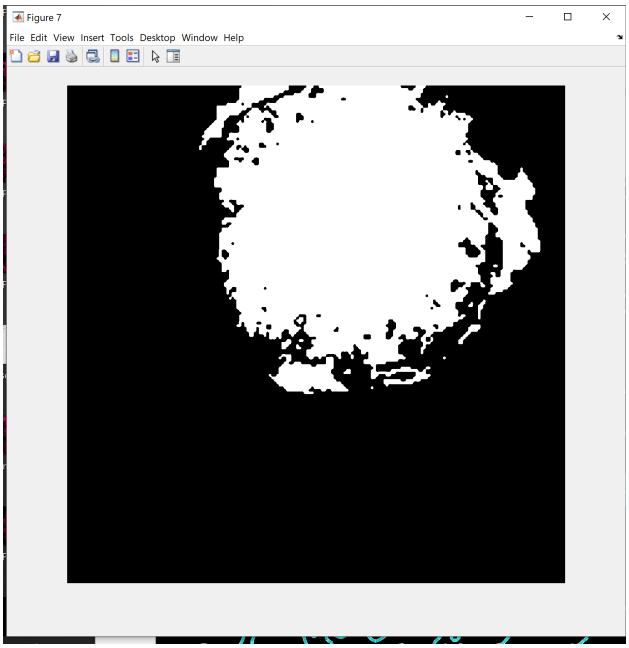

Figure 10: Active contours glomeruli mask

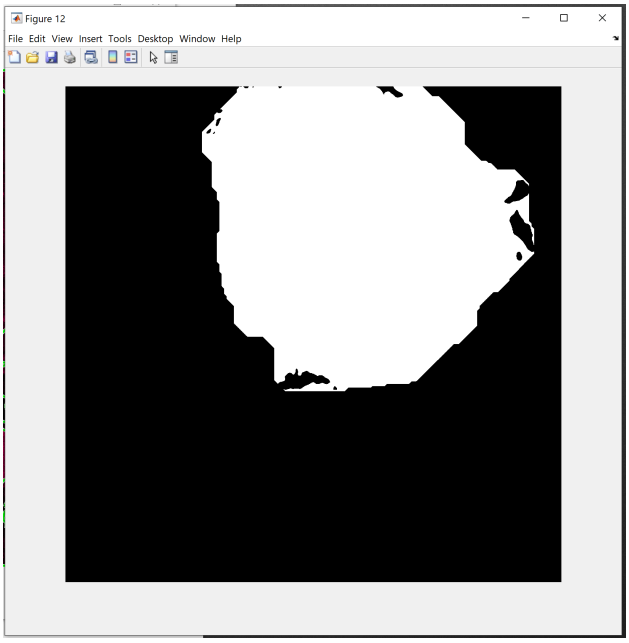

Figure 11: Final glomeruli mask

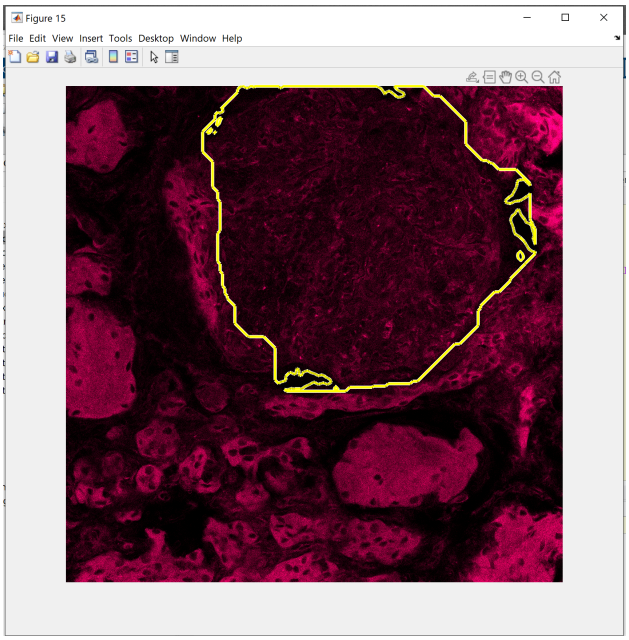

Figure 12: Glomeruli boundary - segmentation viewing

3.7 Tubules

The more complete method for defining a tubule threshold was discussed previously in the methodology section. The user defined tubule sub-sample creates a small mask and applies it to the Guassian filtered image (to perform a calculation on a smoother image) and calculates the mean intensity inside that bounding box. The threshold is defined as a half standard deviation below this mean for inclusivity purposes. Using this threshold a binary mask is generated that will appear such as Fig. 13.
Figure 14 the result of applying closing and area filtering effects to the image. Area filtering is performed based on the expected size of the tubule to glomeruli and stems from the user defined glomerulus size as per previous.
Lastly, Figure 15 is generated by applying the bounding boxes brought over from the binary image to the RGB structural image to assess for tubule segmentation. If there are issues here it is worth noting that perhaps the sub-sample of tubule was too intense (bright) if it doesn't appear as inclusive as it should, and vice versa.

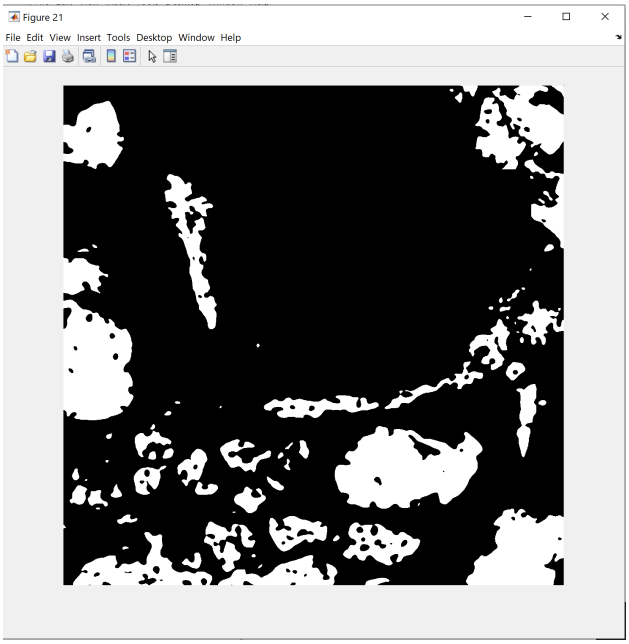

Figure 13: Initial tubule mask (threshold based)

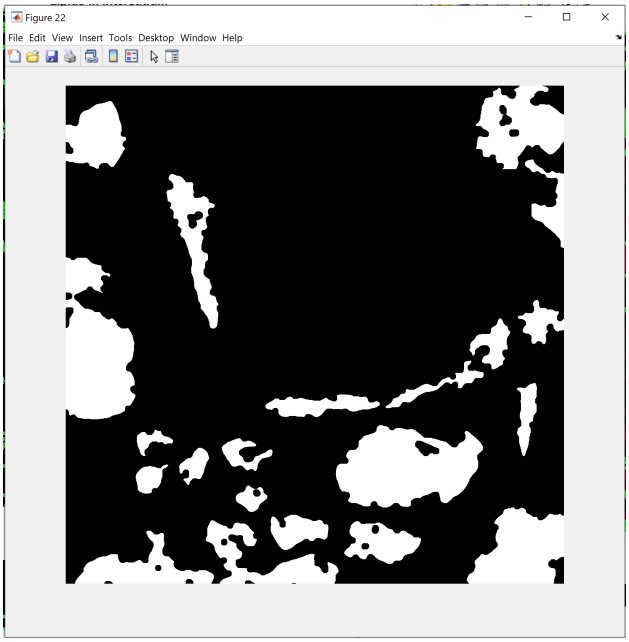

Figure 14: Final tubule mask - closing and area filtering

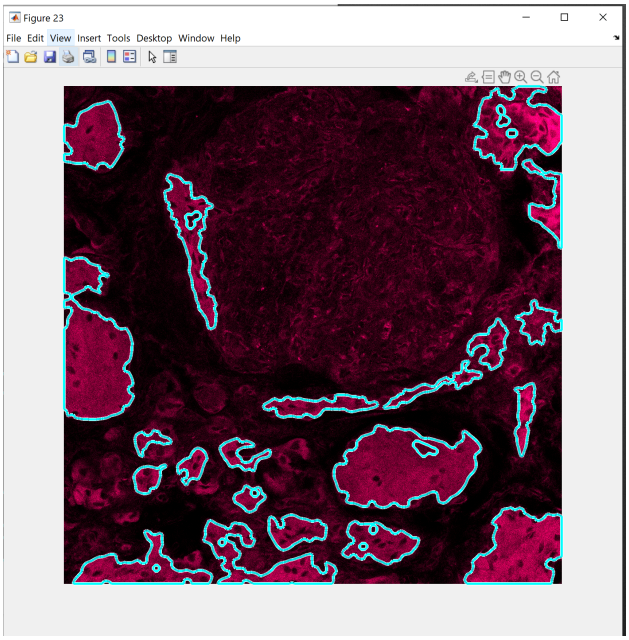

Figure 15: Tubule bounding segmentation shown on RGb structural image

3.8 Interstitium Segmentation

Interstitium segmentation results from segmentation that are not glomeruli, tubule or background. Therefore taking the addition of these three masks and inverting it we are able to arrive at our final interstitium segmentation mask (Fig. 16).
The bounding boxes for this segmentation are shown in Figure 17 and obviously share borders with tubule and glomeruli segmentation masks however they can be though of as the other side of the same boarder. I.e. Canada's border with the U.S. vs. U.S.'s border with Canada.

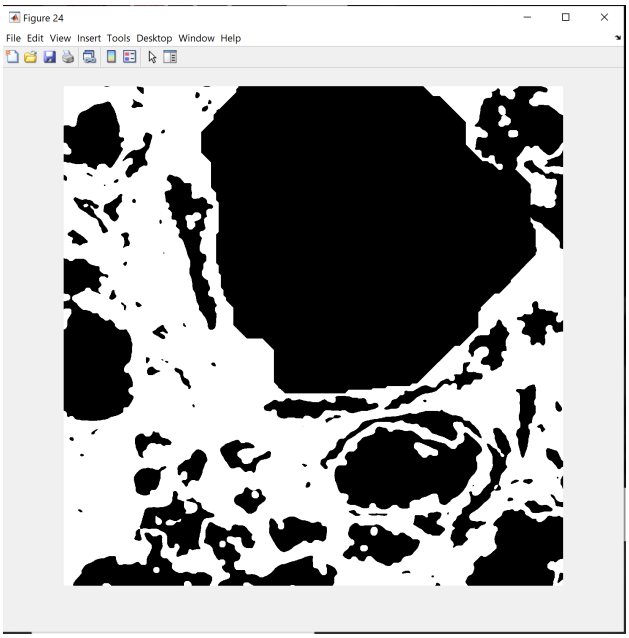

Figure 16: Interstitium mask final

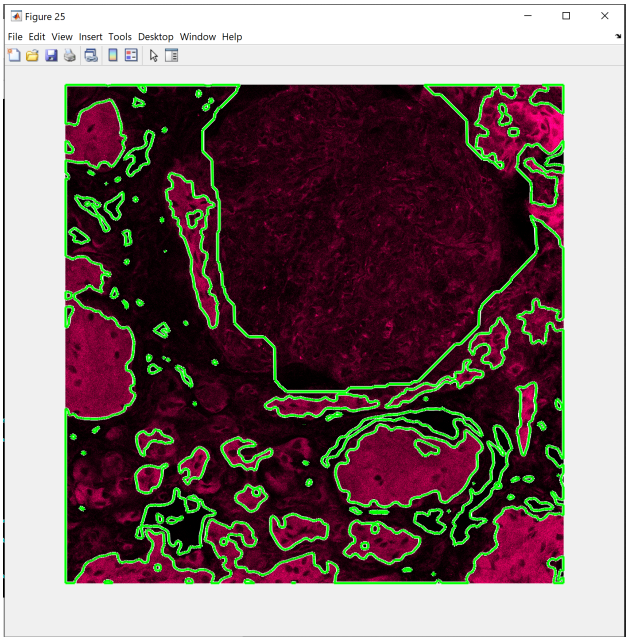

Figure 17: Interstitium bounds on RGB structural image

3.9 Lipid Quantification

The last stage of the code is the lipid quantification algorithm. Using the segmentation areas defined previously. Applying the same bound box areas/masks onto the lipid quantification image we may generate lipid specific structural information.
The lipid image is first converted to a binary image with a threshold defined as ¿ 0. Figure 18 demonstrates this as the original lipid droplet image was had such poor contrast it was not well visualized for the purposed of this manual. Therefore the second phase (binary conversion) is the first lipid droplet image showcased here.
Iterating on the 3 structural masks of interest calculated as above we can apply these same masks to the lipid droplet image (Fig. 19-21). All pixels within that mask serve as the denominator for the morphological structure and all lipid droplets that == 1 within that same mask will serve as the numerator for a decimal percent calculation of total lipid droplets per structure.

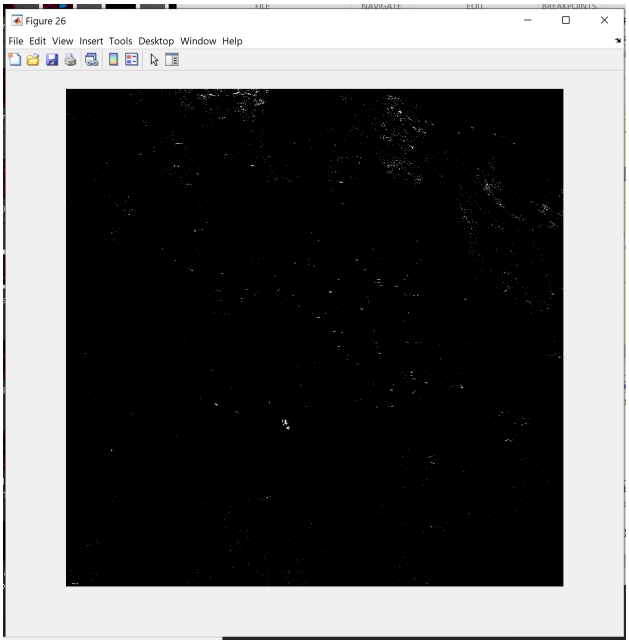

Figure 18: Binary Lipid droplet image

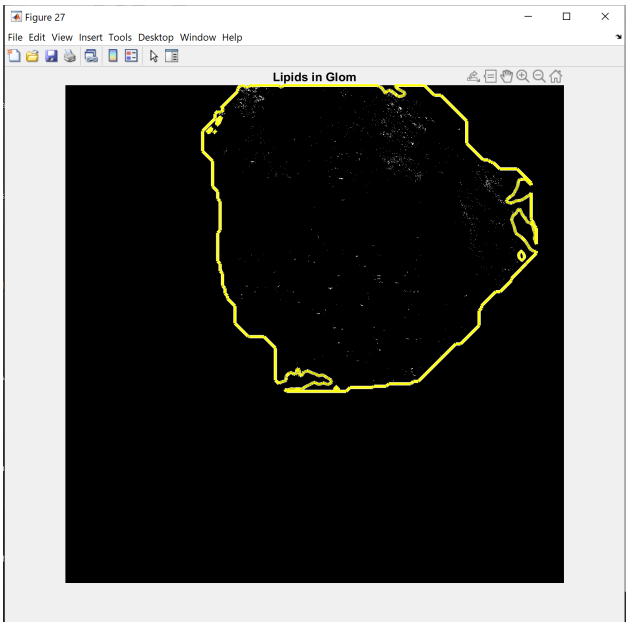

Figure 19: Lipid droplets within the glomeruli mask

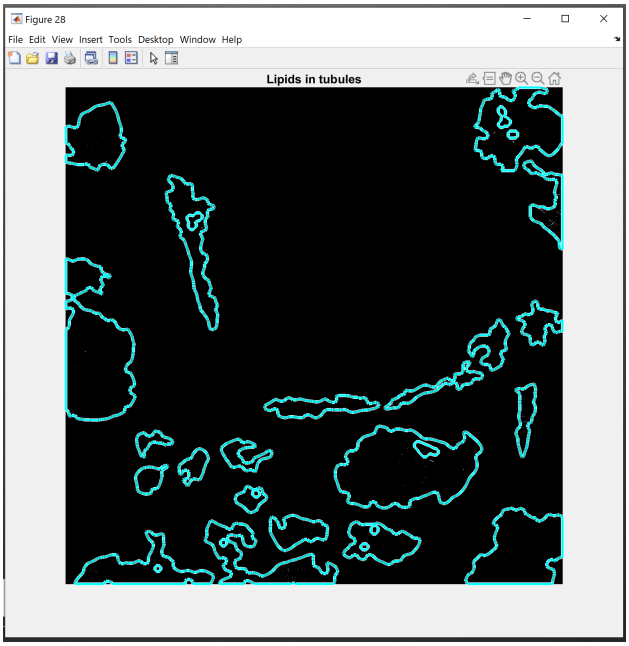

Figure 20: Lipid droplets within the tubule mask

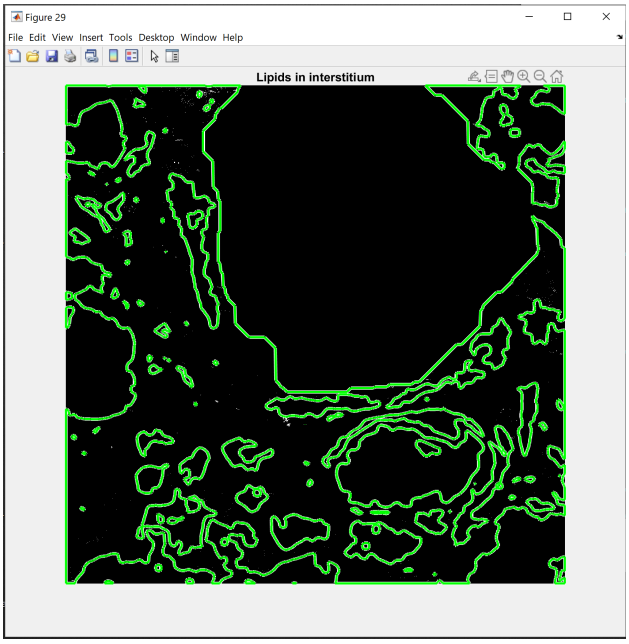

Figure 21: Lipid droplets within the interstitium mask

3.10 Output

Output in the demo code include: image generation of each phase of the algorithm, all the variables generated to create these images and perform the calculation and lastly an Excel spreadsheet with the outputs of decimal ratios of lipid quantification.
The demo code is resultantly slower as it takes more time to output each image associated with each step of the code. However is done to allow the user to save what steps they like for presentations, publications etc.
Figure 22 shows the workspace. This is where all the variables currently in use are stored. This is done to allow the use to investigate or scrutinize for troubleshooting or if these require further images and application of the masks to specific images.
The code as seen in Figure 23 converts the numeric decimal lipid data into a cell format. We define column headers using 3 strings: 'Glomerulus','Tubules','Interstitium'. These will be the column headers in the excel output file. We then join together the column headers and the data creating a 2x3 matrix and use the name outlined by the *savename* convention at the beginning of the code. We then concatenate the *savename* with the format we want to save the file in - in this case .xlsx (for Excel). Lastly we use the function `writecell` to generate the Excel file with the data we have indicated and the name we wish to save it with as input arguments.
The Excel spreadsheet with automatically generate based on the naming convention defined in the earlier part of the code. Opening this excel spreadsheet should look something like Fig. 24.

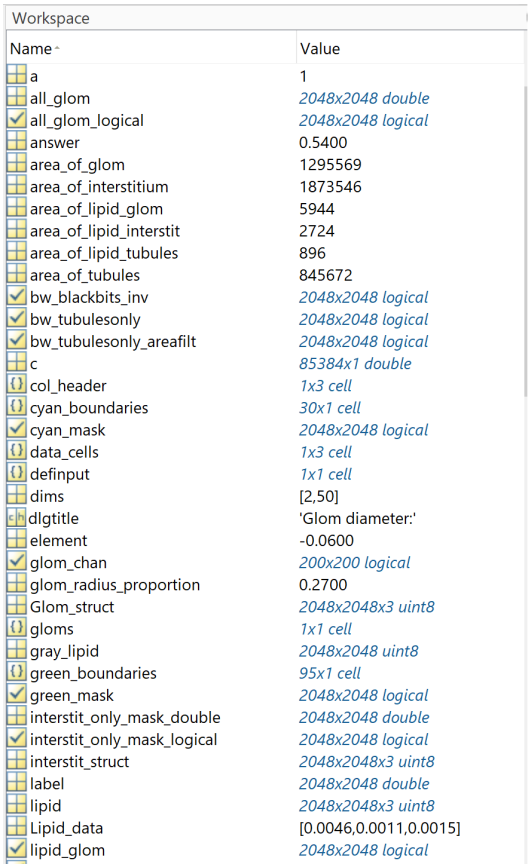

Figure 22: Workspace variables - Matlab

```
%% Output to Excel
data_cells = num2cell(Lipid_data);      %Convert data to cell array
col_header={'Glomerulus','Tubules','Interstitium'};      %Row cell array (for column labels)
output_cell=[col_header; data_cells];%Join cell arrays
save_name = strcat(save_name, '.xlsx');
writecell(output_cell, save_name);
```

Figure 23: Code that defined Excel output

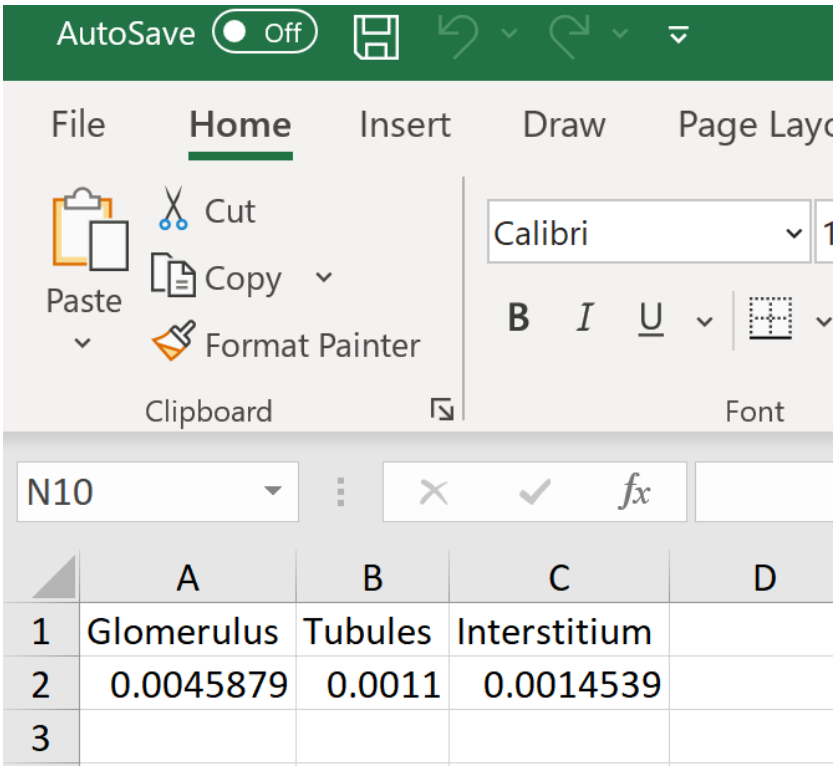

Figure 24: Excel Output

4 Deployment w.r.t. batch processing code

4.1 Pipeline code - about and when to use

This code is for processing a batch of paired structural nile red (691) spectra and lipid (548) files. Not all stages of the image processing pipeline are shown and is whats makes it a pipeline!
To process the files in a batch make sure all image file pairs are in the same folder as the functions and script. You can navigate to the folder in Matlab over on the left hand side of the screen for file directory. The naming convention is important because it will assist with the separation and subsequent different image processing of the pairs for segmentation and lipid quantification. It also generate the file name under which it will be saved. This includes normal versus disease state, sample number and spectra type.
To run the pipeline file all files that you wish to be processed should be contained in the same folder as the code and functions. Keep in mind this means copying files from wherever else they are stored - dropbox or otherwise. The files you wish to process for lipid quantification should be pairs of files and have the name structure: patientnumber_withinpatientsample_spectrachannel.tif. This is important for the name of the files as they are processed so the 691 ad 548 channels of the same sample can remain affiliated as well as save to an excel spreadsheet with the associated file name based on the input data. Should you wish to vary this you will need to alter the code accordingly.
Outputs include an Excel spreadsheet where the amount of lipid droplets contained in the 3 biological morphology - glomerular, tubule and interstitium is reported as percent area (based on bounding areas as they exist in the segmentation portion of the code. This can be changed to automatically save a particular image from the pipeline (i.e. segmentation, or lipid area within each category), but you will have to contact me to have this changed or do so yourself. Should you only want a couple of the images from a set you are analyzing I would advise using the demo code which outputs each stage of the image analysis. These images can be saved to a particular format from the image pop up under the 'file' 'save as'. In the dialogue box of the image and select an image format (.tif, png, jpg etc.). Note the default image save format for images processing in Matlab is .mat (matrix file) so that it can be opened in Matlab.

4.2 Loading in files

Currently the code is set to import any files with a `*.tif` ending within your current working directory or folder. This can easily be changed to accommodate other file types or extend for multiple file types. This is pictured in Figure 25. the variable `allFiles` is created which has created a list of all of the files we will be handling and processing in our pipeline.

```
% store the names of all .tif fi
allFiles = dir('*.tif');
%% Load the Data in
%
```

Figure 25: File type to be loaded in

4.3 Naming files

The naming convention used for the demo code and the pipeline code are the same. The name is split at '_' and '.' (underscores and periods). The first two partitions undergo a string concatenation to form the variable `shortname`. The type of the image I.e. structural or lipid which corresponds to the third partition of the name under which the image is initially loaded is allocated to the `type` subset. See Fig 26.
The structure `Maindata` is created in order to store the names and data of the files being processed. It is accessible in the workspace area. In it you can see the Maindata with all the image names and double clicking here will bring you to a pair of images - structural and lipid.

```
%% create name for processing!
for i = 1:length(trial)
    temp = strsplit(trial{i},{'.','_'}); % split strings at period to drop .tif
    shortname{i} = strcat(temp{1,1},temp{1,2}); % concatenate strings strings at underscore to make name shorter
    type{i} = temp{1,3};
    clear temp
end
clear i % clearing the variable i

%% Import Data and Separate trial types
for i = 1:length(trial)
        data = imread(trial{i});
        Maindata.(shortname{i}).(type{i}) = data; %.data;
clear data
end
clear counter i %clearing variables
```

Figure 26: Naming for processing

4.4 Maindata

The same code as per the demo is run for each pair of images within Maindata. The exception here is the only outputs evident to the user are the user defined stages. This is to save time and more provide computational efficiency when processing many files at once.
The user will still perform glomeruli location, glomeruli size and tubule subsample for each image, though this will occur in rapid succession as there will be no need to wait for all the demonstrations of the algorithmic steps to run since they will all occur now in the background.

4.5 Outputs

Fewer outputs are available in the pipeline version of the code. This is for streamlining and as variable names are repeated and reapplied to new image pairs I have cleared them at the end. The only entities left in the workspace will be the Maindata structure and a results structure (Fig. 27). The user can double click on the Results structure and see the results of the lipid analysis per structure for each image pair. The results mimic the Excel spreadsheet that is also generated in the same fashion as described previously for the demo version of the code (Fig. 28 and 29). The user will see Excel spreadsheets generate within their directory for each image pair continuously as the code runs.

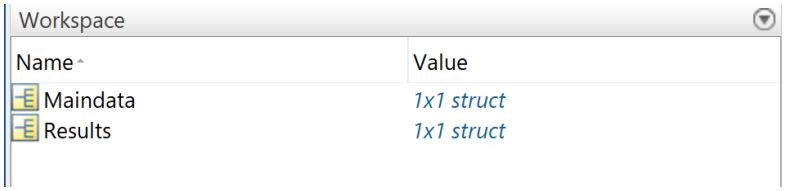

Figure 27: Workspace of Pipeline

```
%% Output to Excel and Results structure
data_cells = num2cell(Lipid_data);      %Convert data to cell array
col_header={'Glomerulus','Tubules','Interstitium'};      %Row cell array (for column labels)
output_cell=[col_header; data_cells];      %Join cell arrays
name = sprintf(shortname{e}); % use trial type to generate the name of the file
save_name = strcat(name, '.xlsx'); %append .xlsx so writecell knows what format to save in


Results.(shortname{e}).data_stats = output_cell;
writecell(Results.(shortname{e}).data_stats, save_name);
```

Figure 28: Workspace of Pipeline

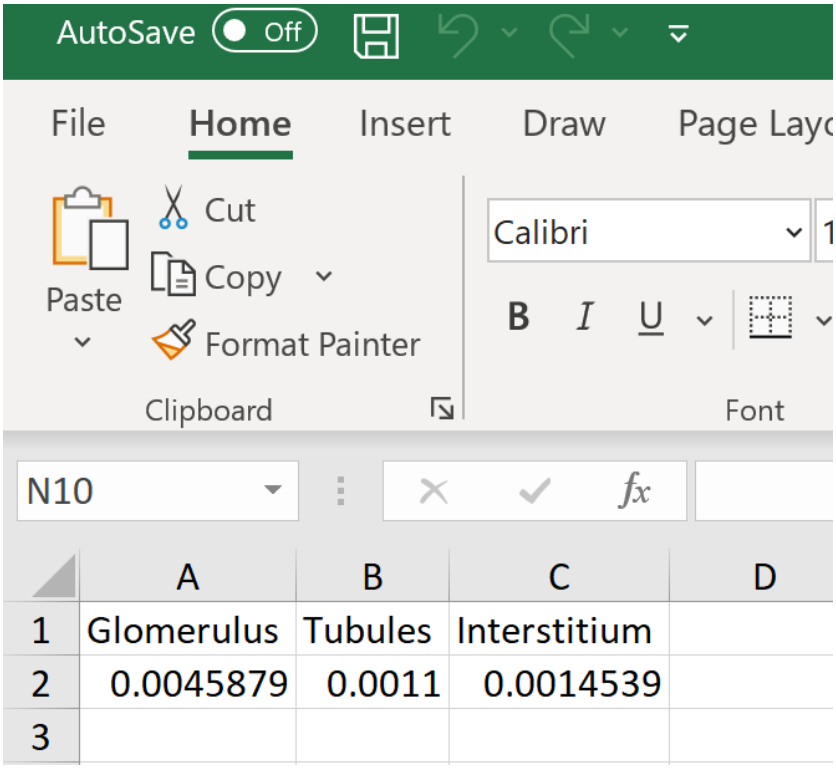

Figure 29: Excel Output

5 Final Comments

The present instance of the code showcases a semi-automated segmentation tool that allows for rapid and reproducible segmentation of glomeruli, tubules and interstitium in Nile Red stained human kidney biopsy specimens and lipid droplet quantification per structure. Nile Red is a fast and inexpensive tool to investigate structural differences making it ideally suited for investigating any changes in glomeruli, tubules and interstitium of healthy versus disease states. Continued research into this important area relies on the correct segmentation and subsequently classification of different morphological renal structures/compartments that are expected to vary with disease state such as diabetes.

Glomerulus segmentations often included the glomerular tuft. This is an area that is sometimes included in the glomerulus but other times not [10]. With this knowledge, however, the user could change the starting size of the semi-supervised segmentation so as reduce the expansion of the Chan-Vese algorithm over the curve set. Additionally, depending on the amount of transection of tubules, a doughnut or 'blob' shape may emerge if the transection is in yz plane versus xy or xz. Such lumens were excluded by the program instead of attributing the area to the intersititum. Limitations with this method include that segmentation of the tubules is based on applying a threshold and therefore is tailored to this particular type of image capture. Chan-Vese segmentation is also prone to errors in starting position so it is crucial that the user defines the approximate centers for the glomeruli at the outset.

Despite these shortcomings, this code as it is deployed now is a prime example of using a human-in-the-loop automation. All deep learning based segmentations require training, this method works by offloading the segmentation work of pathologists without losing their expertise in structure identification [11]. Secondly, should users suffer from too small a sample size (as we did) for deep learning this represents an alternative to investigating morphological differences in kidney disease states. The rapid rise of image analysis as part of the artificial intelligence suite of methodologies has barely begun to be fully exploited in medicine [12]. However, "incorporation of new tools into traditional histopathological evaluation of renal tissue is needed to improve diagnostic precision and predictive value of renal biopsy in kidney disease" [13], [**?**]. It also offers an alternative to direct deep learning segmentation, which is both computationally and temporally expensive to

train, thus greatly limiting its potential for use.

6 Repo and Additional information

6.1 Github:

By clicking on the hyperlink below it will direct you to the repo for this on github.

https://github.com/adriennekline/renal-pathology-segmentation

I will post this later in the summer of 2021 to my github repository after Dr. Chun's lab has had a chance to get their manuscripts underway.

6.2 Contact info

This segmentation algorithm and associated code is tailored to images that fit this profile and image capture protocol. However can be adapted to accommodate other image segmentations.
Should you find yourself having questions please feel free to contact me at: askline1@gmail.com.

References

[1] S. Amoueian et al., "Renal Biopsy Interpretation," *in Renal Biopsy and Pathology*, M. Mubarak, Ed., Rijeka, Croatia: InTech, 2012, pp. 45-64

[2] P. Greenspan et. al, "Nile Red: A selective fluorescent stain for intracellular lipid droplets", *J. Cell Biol.*, vol.100, no.3, pp.965–73, 1985

[3] P. Walker, et. al, "Practice guidelines for the renal biopsy", *Mod Pathol.*, vol. 17, no.12, pp.1555-1563, 2004

[4] L. Barisoni, et. al, "Digital pathology imaging as a novel platform for standardization and globalization of quantitative nephropathology", *Clinical Kidney Journal*, vol. 10, no. 2, pp.176–187, 2017

[5] L. Oni, et al, "Inter-observer variability of the histological classification of lupus glomerulonephritis in children". *Lupus*, vol. 26, no. 11, pp.1205-11, 2017

[6] W. Kang, et. al, "The Comparative Research on Image Segmentation Algorithms," *First International Workshop on Education Technology and Computer Science*, Wuhan, Hubei, 2009, pp.703-707

[7] C. Jayapandian, et. al, "Development and evaluation of deep learning–based segmentation of histologic structures in the kidney cortex with multiple histologic stains", *Kidney International*, vol. 99 no. 1, pp.86-101, 2021

[8] M. Hermsen et. al, "Deep Learning–Based Histopathologic Assessment of Kidney Tissue", *J. Am. Soc. Nephrol.*, vol. 30, no.10, pp.1968–1979, 2019

[9] D. Muruve, "The biobank for the molecular classification of kidney disease: research translation and precision medicine in nephrology", *BMC Nephrol.*, vol. 18 no.1, pp.252, 2017

[10] K. Zou et. al, "Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index", *Acad. Radiol*, vol 11, no. 2, pp.178-189, 2004

[11] SM Sheehan et al. "Automatic glomerular identification and quantification of histological phenotypes using image analysis and machine learning", *Am. J. Physiol. Renal Physiol.* vol 315, pp.1644-1651, 2018

[12] A Holzinger, "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" *Brain Inform.*, vol.3, pp.119-131, 2016

[13] J. Topol et. al, "High performance medicine: the convergence of human and artificial intelligence Nature Medicine", vol.25, pp.44-56, 2019

[14] SM Bagnacio, "Beyond the microscope: interpreting renal biopsy findings in the era of precision medicine". *Am. J. Physiol. Renal Physiol.* vol. 315, pp. 1652-1655, 2018