Quanti-Patch: a Quantification Method for ODC Patchiness

A Documentaiton & Protocol presented
by

Baihan Lin

to
Olavarria Lab
Department of Psychology

in fulfillment of the data analysis in Olavarria Lab
in the subject of
Computational Neuroscience

University of Washington
Seattle, Washington, USA

December 2016

Mentors:
Prof. Jaime Olavarria

Author:
Baihan Lin

Abstract:

The thesis is given birth due to the need to analyze a set of data and the curiosity of finding the best way to generate and present the result. Thus purpose of the thesis is to explore the solutions of presenting the patchness of ocular dominance column (ODC) on V1 of ipsi and contra cortices to determine the critical effect of visual input developmentally and temporally. The thesis discusses the advantage and challenges of using traditional method to analyze patches with Patch Index defined as $PI = 1 - SM/SI$, in which is SM = x max, SI = length of the curve in one dimension of the normalized image color scale. This thesis also points out several constructive points to be further researched upon and finished developing.

Comment:

This documentation is mainly for the purpose of recording my thoughts and attempts in data analysis task, therefore not in a standard of submission in any kind. Feel free to comment and help improve this protocol in customized way.

Keyword:

MATLAB, ODC, Visual Cortex, Patchiness Analysis

Contents:

Traditional Method:

(Challenges Faced)

To determine the patchness, the operational definition might be the patchy darkness difference appeared on V1 of ipsi and contra cortices after tracer injected and HRP reaction. Therefore using ImageJ plot profile is the most approachable choice.

To do that, we need to sort out the most obvious slices from the covered slides or scanner results, and this process is done by Photoshop. Another advantage of using Photoshop is that we could set the color contrast level to make the patches more obvious.

After setting the image more obvious and saving them to TIFF files, we could analyze the TIFF with line tool in ImageJ. Draw a line across the most patch-like area. Analyze plot profile. Then we found the unsatisfactory fluctuating data. The traditional smoothening method is to draw a curve then analyze the length of the curve.

The challenges faced in the entire process might cause some confounding variables of our human error into the data. These challenges include:

a) **Darking/Contrasting Process**

   Patches is what we measured and this process determines the visibility of the patches. If every slice were processed into different visibility of patches, our data measured should not be as accurate.

b) **Utilization of Gaussian Blurring via ImageJ or Photoshop**

   Pure graphical blurring is not as constant as directly smoothening the data itself. Especially in the case of data scanned and have potential minor contamination, they are more vulnerable to the distortion of image data.

c) **Arbitrary Drawing of line in the V1**

   The most patch-like area might not be the area of interest, the bad-quality data area might be what we focus upon. Thus human error is very likely. The trade-off of quality and importance is always an eternal concern.

Traditional solutions are listed here as a reference:

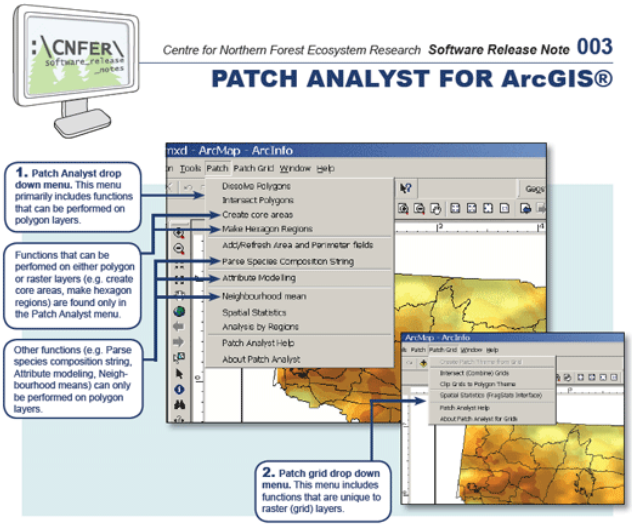
a) Solution 1: Implementing the traditional (via Java or MatLab)

As discussed in chapter 1, we determine the patchness based on the comparative darkness using Photoshop and ImageJ. And based on the plot profile we calculate SM = x max, SI = length of the curve, then we obtain Patch Index (PI) = 1 – SM/SI. This is very effective in determining the patchness, but it also faces the challenges stated before.

Thus I look online and considered about alternative ways to determine patchness.

b) Solution 2: Patch analysis via Patch Analyst (New possible alternative)

Patch Analyst 5



Based on the website description: "Patch Analyst is an extension to the ArcView® GIS system that facilitates the spatial analysis of landscape patches, and modeling of attributes associated with patches. It is used for spatial pattern analysis, often in support of habitat modeling, biodiversity conservation and forest management. Please register with mailing list to be kept informed of changes, bug fixes, updates, etc"

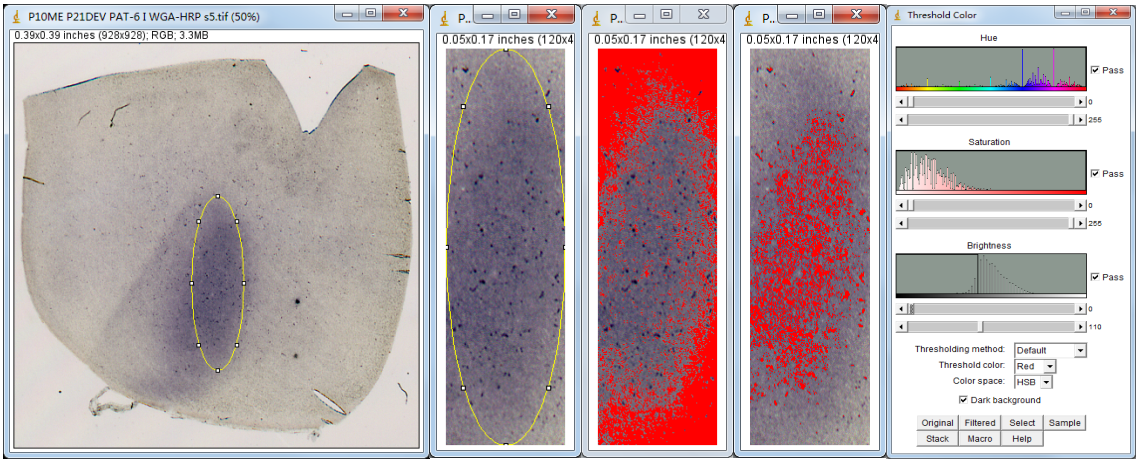
My understanding is that although it is used in forestry, it can be applied to ODC. The program includes capabilities to characterize patch pattern and the ability to assign patch values based on combinations of patch attributes, which means it can distinguish the noise better by recognizing specific patch pattern attribute. Due to time limitation, I haven't tried it yet. **(This is in my plan of exploration in summer).**

The operational definition for patchiness is the ratio of 3D surface area to the flat area, analogous to our traditional ratio of 2D grey value curve length to line length

Reference: http://www.cnfer.on.ca/SEP/patchanalyst/

c) Solution 3: Particle analysis via ImageJ (New possible alternative)

ImageJ can set threshold, then do the particle analysis, normally used in counting objects. I tried: crop, adjust threshold, analyze particle.



The result seems usable, but I just need to figure out what each parameter refers to so as to determine how to show patchness given these data.

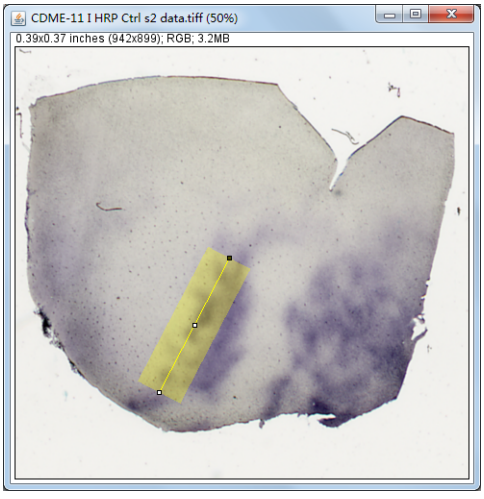
(This is in my plan of exploration in summer).

Results

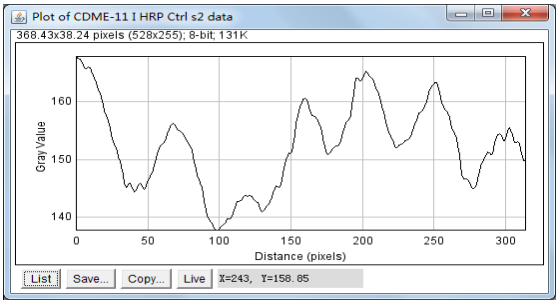
File Edit Font Results

	Area	Mean	Min	Max
795	1.736E-7	103	103	103
796	1.736E-7	107	107	107
797	1.736E-7	88	88	88
798	1.736E-7	71	71	71
799	1.736E-7	102	102	102
800	1.736E-7	101	101	101

d) Solution 4: width of line (Improvement on traditional)



When using ImageJ line measurement and generate plot profile, if we broaden the width of line, the noise will be evened out. This is what I adopted in the data of my program finally in Patch Processor 2.0 (solution 7).



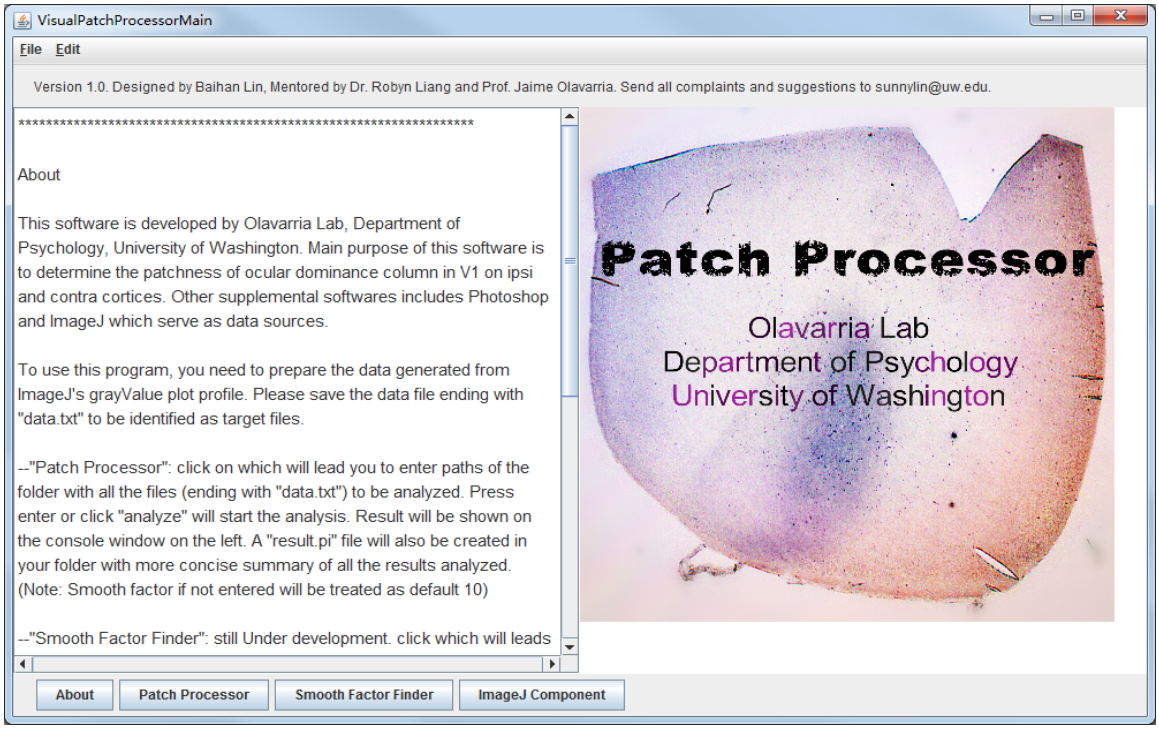
e) Solution 5: Cluster Index

The Cluster Index (CI), used to quantify clustering in the cell plots, is derived from Hopkins' statistic for spatial randomness (Hopkins, 1954; Ripley, 1981). This statistic is based on two basic measurements: (1) the nearest-neighbor distance between the points in the data set (defined as w) and (2) the distance between a randomly selected location in the field and the point from the data set closest to it (defined as x). Hopkins' statistic is defined simply as: (x2)/ (w2). Because this is a ratio, ranging from 0 to infinite, it is mathematically simpler to deal with the log of this statistic, the distribution of which is very nearly Gaussian.

f) Solution 6: Machine Learning

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. Both for classification and regression, it can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. It would be useful is determine the difficulty of simulating centers of patches as an indicator of patchiness.

g) Solution 7: Patch Processor 2.0 by Baihan Lin, Olavarria Lab, UW



For more information, please refer to my other documentation: "Patch Processor: Indication and Optimization of ODC Patch Index"

Definition of Patch Index

In our method, Patch Index is defined as PI = 1 – SM/SI, in which is SM = x max, SI = length of the curve in one dimension of the normalized image color scale.

Consider an image (Figure 1), it can be described as a function z(x, y), where z is its grey scale level. This makes a landscape. The patchier it is, the more hills the landscape have. If we striatify the image, for example, horizontally (Figure 2), it can be displays as a one dimensional function with respect to each y value. Similar can be done to x axis, which is considered a vertical striatification.

Figure 1. Ctrl-ODC-1-original_1 trim-grey.tif

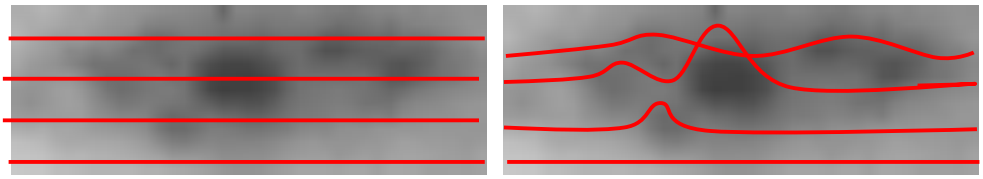


Figure 2. Striate plots from Ctrl-ODC-1-original_1 trim-grey.tif

Therefore, the patchier the image is, the longer the curve line from grey scale function will be, relative to the straight line during the striatification process. After both vertical and horiztonal striatification, we take an average to obtain the average patch index of the given image, in our case, the coverslip image of ocular dominance columns of Long Evans rat primary visual cortex.

Implementation details can be found in Code summary and comments. You can also access the latest updates of this project form my github repository: https://github.com/doerlbh/OLab_patch_processor_3.0.

Normalization of ODE coverslip image

One systematic error in our method is from the previous image processing procedures, which is the internal difference of grey scale exposure. The patch index calculated from the same patchiness can be different if the two images are processed in different grey scale.

To account for this error, we introduced a normalization procedure, in which all the grey scale information are divided by their mean grey scale value. In this way, the patchiness information is still retained, while the grey scale dependency is exempted.

Finally, to make sure the patch index calculated can be meaningful and distinguishing, our method multiply the color scale level by 50, which is an arbitrary number that can be modified in further exploration.

Implementation details can be found in Code summary and comments. You can also access the latest updates of this project form my github repository: https://github.com/doerlbh/OLab_patch_processor_3.0.

Frequently Asked Questions (FAQ)

(for more, email sunnylin@uw.edu or Google)

1. **Why can't my tif files be recognized? Why does it show error message saying file don't exists?**

This can be due to different reasons. A common mistake is to put the parent folder instead of the innermost subfolder as your input. If so, the program can mistakenly assign wrong attributes. Be sure to choose the innermost subfolder to start with. Another common reason can be the naming issue. It is very important that there should be no spaces inside the file name for your starting image files. Other possibilities will need more debugging.

2. **How do I know whether the analysis is finished so I can close the program?**

When you view the console and see ">>" in the front, you are all set to close.

3. **Where can I find the output?**

The log is located inside a folder name after "output" followed by date. More conveniently, all the information is printed in the console window when you run the MATLAB code.

Data Summary

(for 5 cases available now)

---
file:/Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test/Ctrl-ODC-1-original_1_trim-grey.tif
patch index:0.27542
---
file:/Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test/P20MDn-12-ipsi3,4-clean-trim-grey.tif
patch index:0.65883
---
file:/Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test/P20MDnd-73-ipsi13-scale-trim-grey.tif
patch index:0.75417
---
file:/Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test/P21MDd-100-ipsi5-scale-4pxOutlierRemoved-trim-grey.tif
patch index:0.80021
---
file:/Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test/ctrl-ODC-4-section-2-original-trim-grey.tif
patch index:0.55323

As shown above, the difference between the control group and the MD group is obvious. More analysis should be done and t-test should be applied with bigger sample to draw a more solid conclusion.

Bibliography:

Driscoll, T. A. (2009). *Learning MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Feng, J. (2004). *Computational neuroscience: Comprehensive approach*. Boca Raton: Chapman & Hall/CRC.

Special Acknowledgement:

Thank Prof. Jaime Olavarria and Dr. Adrian Andelin for giving me the opportunity to analyze his data! In order to present his the most convincing data, I am lucky to review, utilize and summarize my ability of information searching, computational modeling and conception actualization. In the entire process of developing ideas into software, I learnt a lot from the questions and reflections at every step and had the fortune to enjoy the excitement of tackling the challenges one by one! His tolerance on the deadline and encouragement on the difficulty give me the strongest fortitude to proceed!

Thank Prof. Jaime Olavarria for introducing me with insightful lectures into this dynamic field of biopsychology and neuroscience and letting me getting involved in the challenging projects of his warm lab! (It was also my honor to be able to be peer TA in his class. Thank him to give me courage to challenge Psychology Honors Program.)

Thank University of Washington for giving us the platform to scientifically explore practical academic problems interdisciplinarily!

I will continue the voyage of exploring the infinite realm of computational neuroscience in my academic career fearlessly.

Baihan Lin
December 2016

Code Summary:

(in order of importance from high to low)

1. quantipatch.m

```
% Main Function: quantipatch
% To quantify the patchiness in a 2-dimensional space

% author:   Baihan Lin, for Olavarria Lab
% date:     Dec 2016

% Starting a new analysis so we want to eliminate all old variables

clear all
close all

% I decided to use automatic method for data collection:

prompt = 'What is your folder?: ';
disp('e.g.  /Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test');
path = input(prompt,'s');

% For example, /Users/DoerLBH/Dropbox/git/OLab_patch_processor_3.0/test

% So that I can specify a folder to access all data files.

[~,list] = system(['find ' path ' -type f -name "*.tif"']);

system(['mkdir ' path '/output-' date]);
system(['cd ' path '/output-' date]);

diary(strcat(path, '/output-', date, '/report_', date, '.out'));
diary on;
disp(path);
disp(date);

files = strsplit(list);
```

```
length(files);

files = unique(files);
files = files(~cellfun('isempty',files));

% calculate the patch index

for f = 1 : length(files)

    file = files{f};
    pi = patchindex(file);
    disp('----------------------');
    disp(strcat('file: ',strcat(file)));
    disp(strcat('patch index: ', num2str(pi)));

end
```

2. patchindex.m

```
% Dependent Function: patchindex
% To calculate the patchindex in a 2-dimensional space

% author:   Baihan Lin, for Olavarria Lab
% date:     Dec 2016

function pi = patchindex(file)

image = imread(file);
image = cast(image,'double');
image = normimag(image);

xindex = striatindex(image.');
yindex = striatindex(image);

pi = (xindex + yindex)/2;

end
```

3. striatindex.m

```
% Dependent Function: patchindex
% To calculate the patch index in a 1-dimensional space as striate

% author:   Baihan Lin, for Olavarria Lab
% date:     Dec 2016

function si = striatindex(X)

Xsize = size(X);
flat = Xsize(2);

vec = X(:,2:end) - X(:,1:end-1);

veclength = sqrt(vec.^2 + ones(Xsize(1), Xsize(2)-1));
curve = sum(veclength, 2);

si = 1 - flat/mean(curve);

end
```

4. normimag.m

```
% Dependent Function: patchindex
% To normalize the image to be same exposure level

% author:   Baihan Lin, for Olavarria Lab
% date:     Dec 2016

function normalizedImage = normimag(image)

normalizedImage = 50*image/mean(mean(image));

% 50 is an arbitrary number to choose, but it does a good job.

end
```