A system to determine neuron morphology in 3d two-photon microscope images

Issac Buenrostro
Massachusetts Institute of Technology

Abstract

A system was developed that could find the locations and borders of the neurons and astrocytes in a 3d image obtained *in vivo* from a two-photon microscope. The system denoises the image and assigns probability scores to many possible neuron borders for every cell found in the volume. The borders corresponding to the maximum score establish the morphology of the cell. The result were compared with a more reliable segmentation in 2d. It was found that more than two thirds of the neurons could be identified by the system, finding even some neurons that could not be found in 2d. The high amount of noise in the volume was responsible for most of the false positives and false negatives. The sizes of the neurons were also compared, finding that the 3d segmentation gives mostly smaller cells, with a mean error of 41.9%. A 3d processing, instead of segmenting each layer separately, was determined to be essential for the correct segmentation of the cells.

Introduction

Two-photon microscopy is a powerful technique to measure the activity of multiple neurons in a brain region. Using this, the interactions between close neurons can be inferred by detecting similar tunings in cells.

Usual microscopes use a laser to excite dyes applied to the sample. The excited dyes then return to their ground state producing a photon of a slightly different wavelength than the laser. These photons are directed by lenses and collected by a photon detector. Although useful, the scattering of the laser in the sample makes it difficult to produce sharp images when a large magnification is needed.

Two-photon microscopes excite the dye using only half the required wavelength, therefore two concurrent photons are necessary to actually excite the molecule. After one photon has stroke, there are a few femtoseconds in which a second photon can strike. Usually this is highly unlikely. Two-photon microscopes use a very high power femtosecond pulse laser, which is focused by lenses in the precise part of the sample that is to be measured. Only where the ray is focused can the density and synchronization of photons be enough for two of them to strike a molecule in a few femtoseconds. At the end, in a microscope with magnification 20, only a volume a couple micrometers wide, and about three times as high is excited, giving a very precise intensity measure. The laser is focused to every pixel in the image, obtaining at the end a complete image of the region.

Since the refocusing takes time, and roughly every micrometer needs to be read, the images are slow to produce. In 3d images where the lens must be moved up and down, it is even slower, limiting the number of times the region can be measured. After obtaining the volume many times, all the images can bea averaged to increase the signal-to-noise ratio.

When the images are in a single plane, requiring no movements in the z axis, our microscope was able to scan the whole image from 15 to 20 times per second. However, when scanning a volume, the maximum number of scans that it could do were one hundred.

Currently, to have a good temporal resolution, all the experimental measurements are done in 2d images. It is possible to select some regions in the image, where the computer calculates the average intensity. The computer graphs the resulting average intensities. If dyes are applied to the tissue, and absorbed by neurons, the graphs will show the change in dye concentration in the selected neurons. Usually, dyes which attach to Ca^{2+} ions are used, so the measurement will show the concentration of Ca^{2+} over time.

According to results from Kerr et. al. [1], and our own similar results, calcium concentration can indicate the spiking activity of neurons. Calcium bursts, which are seen as increases in intensity, have a strong positive correlation with spikes measured by a clamp applied directly to the neuron. Therefore, the firing rate of a neuron can theoretically be calculated from the changes in the intensity, giving us the ability to measure the activity of multiple neurons at a time in a small region.

All the recordings are made from the soma, which is around 10 micrometers wide. Working at that scale in vivo requires to limit the movement of the animal as much as possible. Mice are anesthetized, and the head is fixed in the microscope, however respiration and blood flow still create movements, which, even if small, can potentially affect the measurements by loosing focus on the neuron. A way to overcome this is to find the precise borders of the cells, so that the reference points can be chosen near the center, giving a range of a few microns for movement.

When working with 2d images, finding the centers in the x and y directions is easy. However, the z direction is also critical, as this is the direction where the most diffusion occur, and therefore more precision is needed. It is essential to know the extent of the neurons over and under the plane that is being observed, and for this 3d volumes need to be analyzed.

While fifteen thousand averages give a very sharp image to extract the neurons, the limit of at most one hundred averages for volume measurements imposes more challenges. Images are noisy, sometimes so much that even as a human it is difficult to pick out good borders for the cells. Since the mouse is only alive for some hours, it is important to use as less time as possible in the initial imaging and cell extraction.

A program was developed that could find, denoise, segment and label the neurons found in a volume. The main objectives of the program were to find most of the neurons that could be found in a 2d image of the same region, draw borders as similar as possible to those that could be drawn in the 2d image, and process a complete volume in less than five minutes.

Methods

Obtaining the images

The images are obtained in vivo from V1 in mice cortex. Mice are anesthetized with Urethane. A craneotomy is performed in the mouse to expose an area a few millimeters wide of the cortex. The dura is removed.

Two dyes are applied to the region of interest. Oregon Green Bapta -1 AM (OGB-1) is absorbed by gray and white matter, and only after absorbed attaches to Ca^{2+} ions. When attached, the dye fluoresces at 517 nm when excited with 488 nm light. The fluorescence of the dye increases 14-fold when saturated with calcium (as reported by the distributor), making detection of calcium concentration possible.

The other dye applied is Sulforhodamine-101 (SR-101) which stains only astrocytes. Its fluorescence is red, and therefore its signal can be separated from the signal of OGB-1.

A dichroic mirror is used to separate the OGB-1 signal, which is measured as channel 1 of the image, and the rest of the signal is measured as channel 2. In this way, in channel 2 only astrocytes can be seen, but channel 1 shows both neurons and glia.

A lens with magnification 20x was used to obtain images of 160 by 140 by 60 voxels. Each voxel measured 1.17 micrometers in the x and y directions, and 1 micron in the z direction. Each transverse section was taken five times, and the stack repeated four times and all the images averaged. In this way, the final result was an average of 20 images.

Also, a 2d image was obtained from close to the central transverse section. This image was averaged 3620 times to increase the signal-to-noise ratio. In this case, before averaging, the image passed through an alignment algorithm that attempted to correct linear transformations of the image due to movements of the animal minimizing the sum of squared errors.

All the processing was first done in channel 1, and channel 2 is only taken into account for the final cell differentiation.

Denoising the image

Before segmenting, the volume was denoised.

First an anisotropic diffusion was applied. A code written by Peter Kovesi from the School of Computer Science & Software Engineering in the University of Western Australia was used. This diffusion is based on results by Perona and Malik [2]. The algorithm diffuses noise in the image, while preserving the borders, by making the diffusion inversely proportional to the laplacian in each voxel. Five iterations of the diffusion were applied, with a conduction coefficient of 40, and a diffusion speed of 0.15.

After this, another image is generated in which each voxel is the average of a region 30 by 30 by 5 voxels centered in that particular voxel. The original image is divided by this image to normalize the intensity values throughout the volume (therefore every region in the volume has roughly the same average).

Calling the program

The program takes as mandatory argument the 3d volume. Additional to this, other arguments can be supplied. The following parameters can be given when calling the program:

- iter- amount of iterations the program will do, default: 20
- magnification- magnification that the image has, default: 1
- lens- lens the microscope had, default: 20
- start- lower cutoff for the cells, default: 0.5
- useMask- supplies a 2d mask for reference, default: 0
- reference- supplies the 2d image when using a 2d reference, default: 0
- correct- runs the neuron correction code, default: 1

When referencing these parameters, the word *params* is added before the name of the parameter. Therefore, during the program, *iter* is called *params.iter*.

Pre-processing of the image

The image is first normalized so that the average for every transverse section was the same. This is necessary because the microscope gets dimmer signal from deeper layers in the cortex.

The image is then further normalized to avoid having negative values, and to make the range of the image from 0 to 1. Since the program uses only the order of the values (ratios of the values make no difference in the segmentation), we are free to modify the values in any way that preserved order. Matlab uses four decimals double values, so it gives enough precision in the range $[0, 1]$.

Thresholding

The program calculates thresholds based on *params.start* and *params.iter*. For that, it creates a linear space of *params.iter* elements between *params.start* and 1, and then calculates the value in the image corresponding to that quantile. In this way, *params.iter*-many values are obtained, each of which will work as a threshold.

For each threshold, going from low to high, a binary image is produced, where every value in the original image below the threshold is a zero, and every value above the threshold is a one. In the binary image, the connected regions (requiring a common face to be connected) are calculated, and labeled as potential neurons. The image is combined with the previous binary image in this way: for every neuron in the previous image, if it disappeared completely disappeared in the new one, the previous neuron is taken. If it didn't disappear, but only shrank, the new one is taken.

The program then goes through all the potential neurons, calculating probability values for each one. Each neuron is then found in the previous image, which also has probability values, and all the way to the first threshold. All these probability values are assigned to the neuron, giving a graph of the probability values as different thresholds are applied.

At the end, for each tentative neuron the program found using the thresholds, it will have a probability graph with *params.iter* data points.

Calculation of probabilities

To calculate the probabilities, each neuron is isolated in a binary image, and three different characteristics of the proposed border are take into account:

- Volume: The volume of the cell is calculated by adding the values in the isolated cell. The volumes are divided by the magnification times the lens squared to normalize with respect to the x and y axis, and by 20 (default lens) to normalize in the z direction. The volume that a sphere with radius 5.5 voxels in a magnification 1, lens 20 image would have is subtracted from this, and the result divided over $1/10^3$. With this we are assuming that the mean radius of a neuron is 5.5 voxels and the standard deviation about $1/10^3$ microns(determined empirically from the images obtained). The result from this is therefore in normals.

  To avoid finding neurons just by size, only neurons which are too big or too small actually use this probability values. If the absolute value of the normal is lower than 15, it is set to 15. If it is outside this bound, it keeps its value. After that, the result is divided over 10, and its sign shifted. The end result are values which are bigger and closer to -3/2 for neurons of reasonable sizes, and sharply drop for unreasonable sizes.

- Shape: Since masks are defined in voxels, they are orthogonal. To calculate the probability value due to the shape, the surface area to volume ratio is taken into account. The surface area is estimated by counting the amount of faces of voxels that are borders of the cell. The surface ares is divided by the raw volume obtained by adding the values in the mask. Since a quadratic term is being divided by a cubic term, the result is multiplied by the geometric mean of the diameters of the figure in the x, y, and z direction.

  Since a neuron is expected to be fairly spherical, the value that a sphere would give is subtracted. Therefore, 3 is subtracted from the previous result, and then it is divided by 3, so that a cube will give 1 as a result, and the sign is changed. The result is that almost all convex shapes will give values between -1.5 and 0.

- Border: The gradients in the border are also taken into account. The perimeter of the binary image is calculated. The laplacian for each voxel in the border is estimated using three kernels, one for each direction. The estimate of the laplacian is the quadratic mean of these three values.

The laplacians in the border are averaged and multiplied by the magnification times the lens, because bigger amplifications will produce slower gradients. The results are usually between 0 and 3.

The three values and then added to obtain the probability score. Because of the signs the values have, the best probability scores are obtained by cells within the reasonable volumes, with a sphere-like shape, and strong laplacians in the border.

Finding the correct borders

For each proposed neuron, the maximum in the probability scores is found. The border corresponding to that score is recalculated, and drawn in a new volume the same size as the original. Every neuron is labeled with a unique positive integer to make them easy to find for any other program that might need them. The maximum probability score for the neuron is kept, so that the user can choose to display only those neurons which have a score above a certain threshold (which are more likely to be correct neurons).

Neuron correction

Sometimes, the maximum probability will cause a neuron to have a threshold low enough to invade a close neuron. To avoid this, if *params.correct* is set to 1, the program executes a correction step.

If the program finds that after drawing a neuron, a voxel is part of two neurons, it will send both neurons to the correction algorithm. This algorithm uses twice as many thresholds values. For each of the two neurons, it finds the maximum point of the neuron (which is labeled as the center of that neuron). Then, starting from the highest threshold, the proposed borders that include that center with that threshold are calculated. The program determines if the proposed border is invading the center of the other cell, and stops one threshold before that happens.

From this, the program has two proposed neurons which don't invade each other. For each one it adds the original proposed borders and the new proposed borders using the logical operator AND, so that the smallest border is chosen. The result from this will be the highest probability borders that don't invade the neighboring cell.

The correction is executed until there are no more overlapping neurons.

Using 2d references

The program can use a 2d image as a reference to find the neurons. Since 2d images can be averaged many more times, the segmentation is easier in those images.

A 2d image is taken, and the program attempts to align it using a linear transformation with every transverse section. The code that does this was developed by Andreas Hoenselaar from Andreas Tolias Lab. The output of this code is a 3 by 3 matrix with the best linear transformation it found, and the sum of squared errors after aligning. The minimum in the errors is found, from which the z coordinate for the reference is deduced.

A segmentation code previously developed by James Cotton from Andreas Tolias Lab, and which is the one currently used for all the data analysis, is then run in that 2d reference, reliably finding all the neurons in it. The program takes the 2d image in *params.reference* and the segmentation in *params.useMask*. The optimal linear transformation is applied to the 2d segmentation, to match the neurons with the 3d alignment in the 3d volume.

After this, for each neuron in the 2d image, the maximum value in that region in the 3d volume is found. The highest threshold less than or equal to that value is found, and applied to the whole image. The connected region which includes that voxel is selected. The program looks for all the local maxima in that region, if there is more than one local maximum, the region is discarded, and the program determines that the neuron can't be segmented (because it leads to at least other two neurons, assuming each local maximum is part of a different neuron).

If there is only one maximum, the probabilities for the connected region that includes that maximum in each threshold are calculated, and then the neuron segmented in the same way as the program would normally do. The label assigned to the neuron is the same as the label that the 2d reference had, so that they can be compared easily. After having the 3d morphology of each neuron in the 2d reference, the z coordinate of the slice can be calculated.

Differentiating neurons from glia

When the segmentation is done, the system takes the volume with the neurons drawn, and for each cell it determines if it is actually a neuron or it is glia.

For each neuron, it calculates the average intensity of the region in the channel 1 image, and divides it by the average intensity of the region in the channel 2 image. It collects all the values obtained this way, and finds the biggest difference between two consecutive values. Every value above that is labeled as a neuron, and every value below that is labeled as an astrocyte. Since in channel 2 two only astrocytes fluoresce, this is a reliable way of identifying the different types of cells.

Results

To determine the quality of the results, the segmentation obtained by the program was compared to other segmentations. Due to the amount of noise in the volumes, and the fact that it is in 3d, segmenting the same volume wasn't a reliable way of comparing them.

A 2d image was taken of a slice in the volume. The image was taken 3620 times and averaged, obtaining a very good signal-to-noise ratio. Both the volume and the frame were multiplied by a coefficient to make their range go from 0 to 1 (for each one). The 2d image was determined to be in the coordinate 25 of the z axis of the volume. The root mean square of the differences between corresponding pixels was 0.0485, which is a 4.9% difference between the frames, attributable to noise.

An iterative algorithm tried to find a linear transformation that minimized the root mean square. A matrix was produced with the optimal linear transformation.

A code that is usually used in the lab for 2d segmentation was applied to the 2d image, to obtain the neurons (Figure 4). The previously found linear transformation was applied to the segmentation, obtaining neurons that theoretically should fit in the 25th layer of the volume.

The 3d volume (Figures 1 and 2) was segmented with 40 iterations and neuron corrections, and the 25th layer of the result compared to the 2d neurons (Figure 3). The 2d code found 67 neurons. The new system found 80 neurons in the same region. Of the 67 neurons that the 2d code found, 48 were also identified by the 3d code, which gives a 71.6% accuracy in identifying the neurons that the 2d code finds.

In the 2d segmentation, there were 20 neurons that could be identified by a human and the code had missed. From those neurons, the 3d system found 12, giving a 60% accuracy in identifying the neurons that can be seen in the 2d image but are missed by the 2d code.

It is possible for the 3d algorithm to determine that the borders of a certain neuron are slightly smaller than they should be, and therefore a neuron that can be seen in a certain transverse section could be found only in nearby sections. The mask was dilated by 1 voxel (Figure 7)in the z direction, getting an accuracy of 73.9% in identifying neurons found by the 2d algorithm, and 70% accuracy in identifying neurons not found by the other code. If the dilation was of two voxels (Figure 8), the first accuracy increased to 78.3%.

The 3d mask was modified to display only neurons which had a positive probability score (Figure 5). This time, only 69 neurons were found in the 25th layer. 46 neurons in the 2d mask were also found in this version of the 3d one, giving a 68.7% accuracy for only neurons with positive probability scores. 45% of the missed neurons in the 2d mask were still found with this modification.

Proposed neurons which where too small to be neurons (less than 125 voxels in this case) where dropped to produce a new mask (Figure 6). Only 64 neurons were kept in layer 25, which means that 20% of the original neurons were either not of the correct size or probably not neurons (and are easy to rule out). The accuracy for neurons in the 2d mask dropped to 65.7%.

False positives are determined only by visual exploration. 2 false positives were found in the original mask, out of 80 proposed neurons. This gives a 2.5% false positive rate. The rest of the found cells, which haven't been counted before, are cells which are mostly in other layers, and therefore they can't be reliably found in the 2d section analyzed.

The code was then run using the 2d image and segmentation as references. From the 67 neurons that were in the 2d image, only 48 could be reliably identified in the volume. From the rest, 4 corresponded in to locations in the volume that were too dark to be neurons, and for 15 it was impossible to determine which neuron corresponded to that mask between two or more in the 3d volume. The success rate for the neuron segmentation using the 2d image as a reference was therefore 71.6%.

Assuming the segmentation in the 2d mask was correct, the area of the neurons in that transverse section was compared between the 3d and 2d masks. Only neurons that could be identified in both segmentations were considered. The absolute difference was calculated, and divided by the area in the 2d case. The mean error was 41.9% of the area, with a median of 30.4%, and a standard deviation of 40.3%. The ratio between cells that were bigger in 3d and cells that were smaller in 3d was 0.32, meaning that most of the cell segmentations gave a smaller cell in the 3d case.

Discussion

Assuming the 2d segmentations are trustworthy, the 3d system is finding around two thirds of the neurons in the volume. Most of the missed cells happen due to noise, where the neuron is difficult to identify even by a human, or completely invisible in the volume. The sample used was averaged 20 times, and was compared against another one averaged 3620 times. The signal-to-noise ratio difference is evident in Figures 3 and 4.

The 2d code was run in 25th layer of the volume to have a further comparison. Most of the cells were not convex and had holes, showing how much noise can affect the result, and that the information provided by the third dimension is valuable.

The false positives were generated mainly due to local maxima generated by the denoising algorithm, but their number was so low, and the effects of the denoising so important, that no change was applied.

Regarding the size of cells, it depends mainly in the amount of thresholds applied and the probability scoring. If few thresholds are applied, the system doesn't have the opportunity to analyze enough possible borders to find the best one. However, increasing the number of thresholds increases both the execution time and number of false positives.

The probability scorings could be modified with more reliable data regarding the shape, volume and gradient parameters that neurons have. However, with respect to gradient, since it all depends on the system being used for imaging, it is unlikely that useful literature will be found. A learning system could be implemented in the algorithm, which could use the parameters for successful neurons previously segmented to find the optimal means and standard deviations.

The system was successful in its objectives, being able to determine the location in the z axis where the measurements are being done.

References

[1] Kerr, J, Greenberg, D, Helmchen, F. *Imaging input and output of neocortical networks in vivo.* 2005.

[2] Perona, P, Malik, J. *Scale-space detection using anisotropic diffusion.* 1990.

Annexes

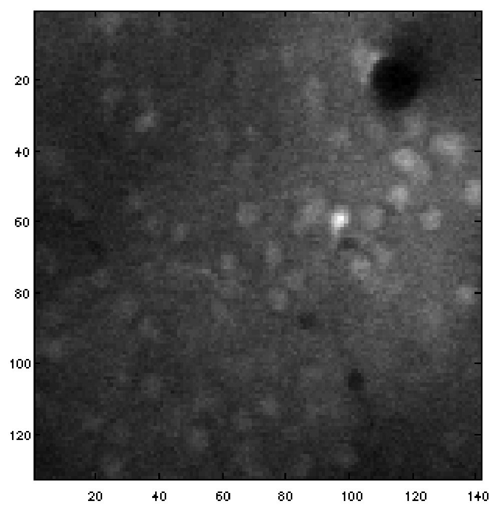

Figure 1: Transverse section of the raw image

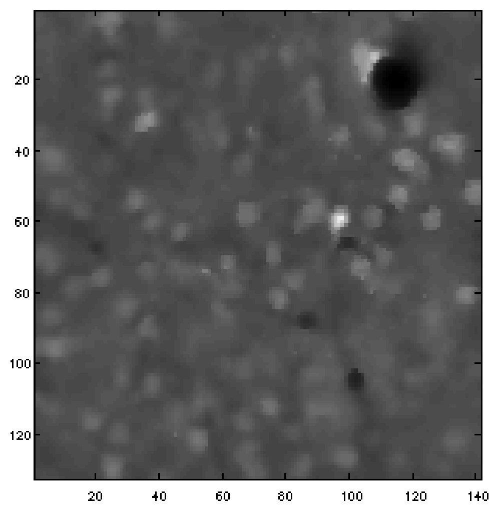

Figure 2: Image after the denoising algorithm

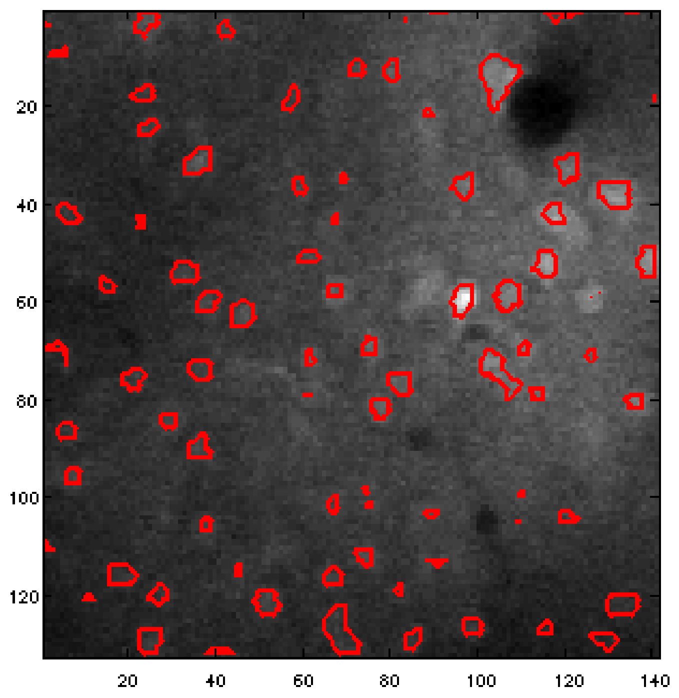

Figure 3: Segmented volume

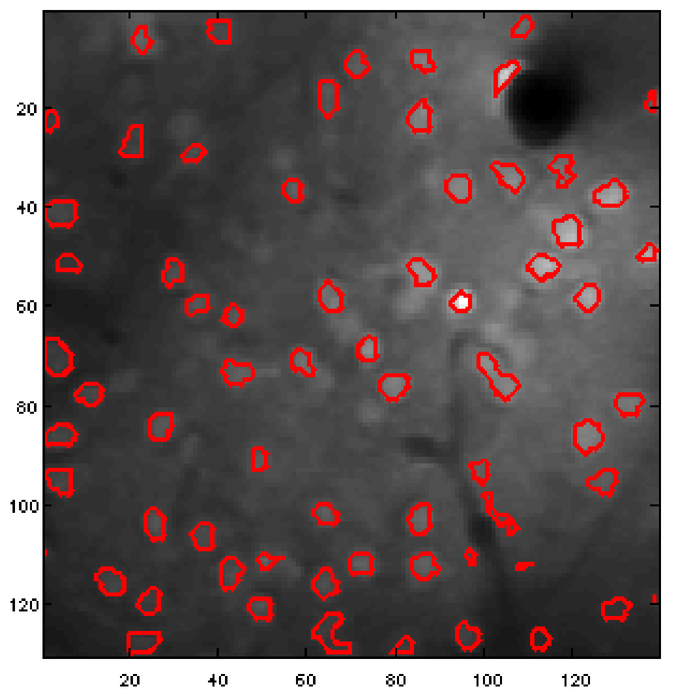

Figure 4: Segmented 2d image

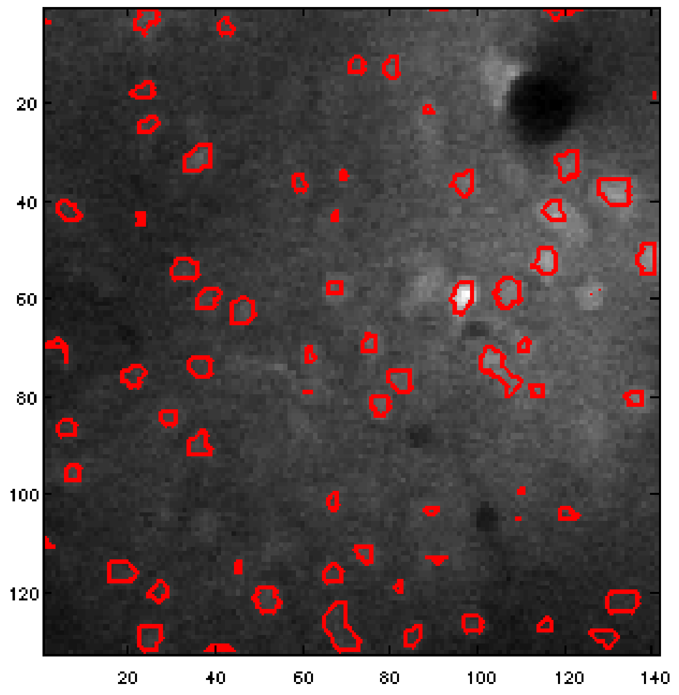

Figure 5: Only neurons with positive probabilities

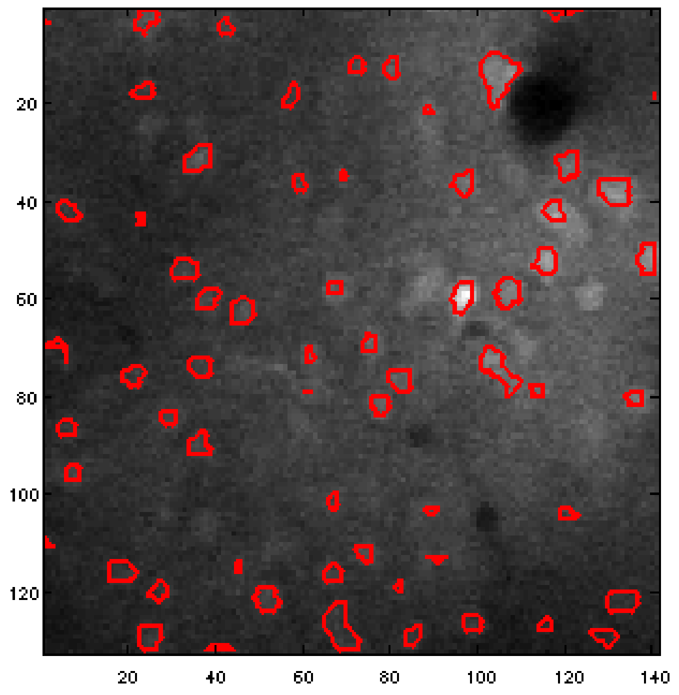

Figure 6: Small cells dropped

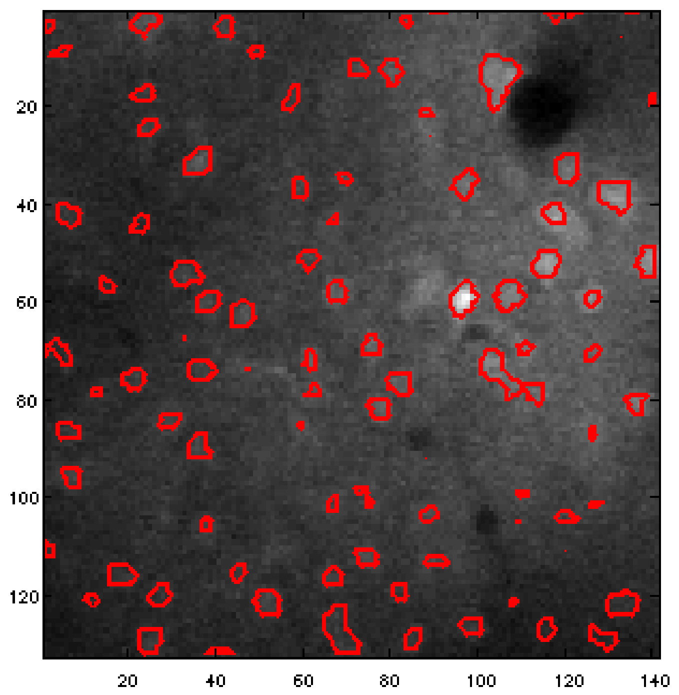

Figure 7: Dilation of 1 voxel in z direction

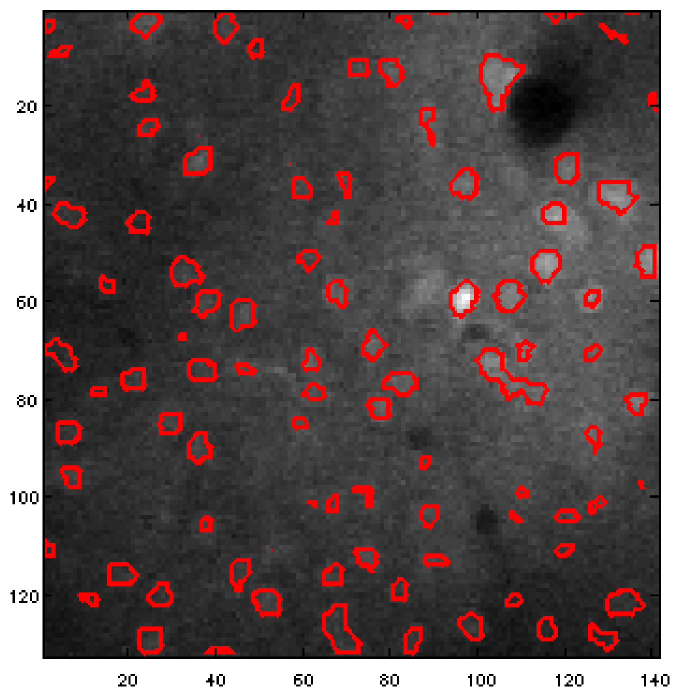

Figure 8: Dilation of 2 voxels in z direction