VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF ELECTRONICS
DEPARTMENT OF ELECTRONIC SYSTEMS

Brain lesion detection using neural networks

Neural networks course project

Done by: EKSfmu-16 gr. st. Arnas Butkus
Checked by: prof. dr. Artras Serackis

Vilnius, 2017

1 Introduction

Magnetic resonance imaging (MRI) scans allows seeing the situation within the body. It is primary means of seeing if there are any issues with the brain. The scan in itself does not tell if there is an issue and a trained medic needs to determine if there is an issue. One of such issues is the blood spill in the brain lesion for short. There are cases where lesions are small and hard to notice and require an expert to spot them; sometimes they are a huge glob.

So the goal here is to see if it is possible to implement neural networks to at least classify if there is an issue, and ideally mark the lesion area. And here I look through some MATLAB solutions to at least similar problems proposed by others, looking for a method that might work to some degree.

I explore existing methods in MRI scan segmentation and/or delineation. Starting with a method utilizing Statistical Parametric Mapping (SPM12) toolbox for MATLAB, working on 3-dimensional MRI scans. Then explore few other methods 2D images, slices, of MRI scans.

2 Analysis of existing volumetric delineation algorithm

Theres is a variety of automatic and semi-automatic algorithms for some form of lesion or some other specific brain region delineation [2, 31, 34–42]. I started of by analysing Joseph C. Griffis algorithm, `lesion_gnb.m`, for lesion area delineation [2]. It is run using MATLAB and, as provided [45], works with any MRI scans presented as .nii files while supplying extra configuration variables during runtime. Algorithm utilizes functions provided by SPM12 toolbox, which is available for free on the internet [46]. When running the script at first it throws an input box asking, should the segmentation be performed on the MRI scan. Input is either "Y" or "N". If "N" then MRI scan segmentation is skipped.

In general, when running for the first time, user would have to perform the segmentation, thus selecting "Y". Next dialogue asks to pick a directory where all the files will be put. The one after that asks to provide the MRI scan file. Thus processing of the MRI scan begins with its segmentation into grey matter, white matter and cerebrospinal fluid (CSF). For this purpose unified segmentation algorithm, which is provided by the SPM12 toolbox, is used.

2.1 Unified segmentation

When attempting to segment brain images into certain classes two approaches can usually be taken: tissue classification or registration with a template. Tissue classification method assigns voxels to a tissue class according to their intensities. For this to work the intensities of each tissue class needs to be characterized, which is usually achieved by choosing specific voxels to represent each class [24–29, 32]. Automatic way this is done is by first mapping the brain to some standard space and automatically selecting voxels that have a high probability of belonging to each of the class. A similar approach is to model the intensity distribution by a mixture of Gaussians, while using tissue probability maps to weigh the classification according to Bayes rule [11, 16, 19]. Registration with a template method involves some specific type of registration where template brain is warped to match the brain T1w scan to be segmented [9, 18, 44]. Not necessarily the volume matching methods are to be used here as methods that are based on matching surfaces [30, 47, 48] would also work in this category. These methods work by overlaying regions that are predefined on the templates, thus allowing different structures to be identified automatically. Unified segmentation uses both the tissue classification and registration with template methods for more accurate segmentation of an MRI scan.

To start tissue classification the images need to be registered with tissue probability maps [49]. After registration these maps represent the prior probability of different tissue classes being found at each location in an image. Bayes rule can then be used to combine these priors with tissue type probabilities derived from voxel intensities to provide posterior probability [5]. This procedure is circular registration requires initial tissue classification and tissue classification requires initial registration. To resolve this a single generative model is used. This model also includes parameters accounting for the image intensity nonuniformity for both segmentation and registration. To find these parameters algorithm alternates between classification, bias correction and registration steps which provides better results than serial application of each component.

To account for image nonuniformity parametric bias correction is used. Many bias correction models are based on modelling the intensities of different tissues as a mixture of Gaussians. There are three commonly used models of how the bias interacts with noise. First is when the resulting signal (y_i) is an original signal (μ_i), scaled by some bias (ρ_i) with added Gaussian noise (n_i) that does not depend on bias [50]. This assumes that the noise if from MRI scanner itself (1).

$$y_i = \mu_i/\rho_i + n_i \tag{1}$$

Second model, which is used by the unified segmentation, is similar to first one except the noise is added before the signal is scaled, which implies that the noise is due to variation in tissue properties (2). There is an option of accounting for both tissue and scanner noise, which is likely to be a better option especially for the images that have large amounts of bias. However, unified segmentation uses in the SPM12 uses single source model.

$$y_i = (\mu_i + n_i)/\rho_i \tag{2}$$

Third method applies logarithmic transformation to the data first which then allows multiplicative bias to be modeled as an additive effect in logarithmic space. The cost function for these approaches is related to the entropy of the distribution of log-transformed bias corrected data. As with the non-parametric model based on log-transformed data, low intensity voxels have to be excluded to avoid numerical problems. The generative model is of a form similar to one given in equation (3) which then gives a exponential multiplication in resulting signal function (4)

$$\log y_i = \log mu_i - \log \rho_i + n_i \tag{3}$$

$$y_i = \mu_i e^(n_i)/\rho_i \tag{4}$$

Another parameter used in segmentation is priors probabilities determination. Rather than using stationary values based on mixing proportions, additional information is used utilizing information from other subjects brain images. Usually priors are generated by registering a large set of brain images and averaging resulting tissue classes. This gives a set of tissue probability maps white matter, gray matter and CSF representing probabilities any of the matter types of being in specific areas of a brain.

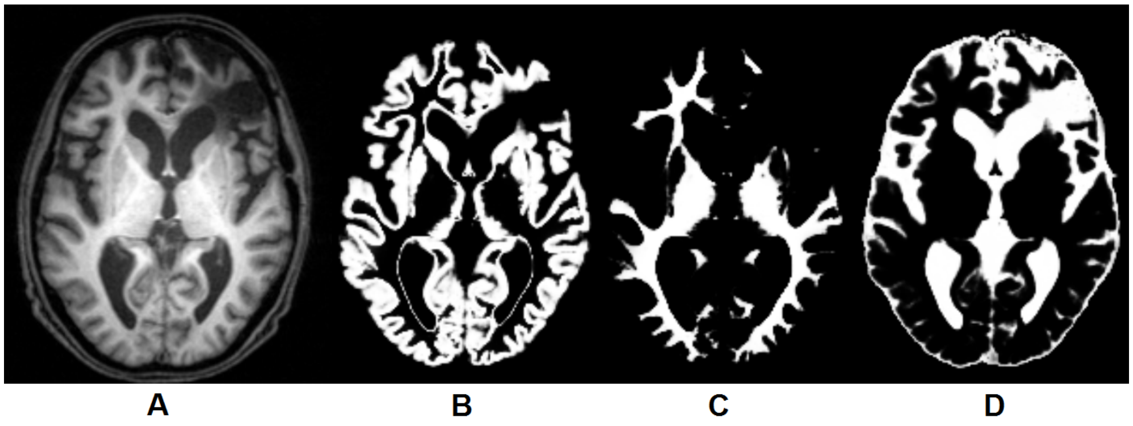


Figure 1: Research results obtained using unified segmentation: patient MRI scan slice (A), gray matter TPM (B), white matter TPM (C) and CSF TPM (D)

2.2 Generation of lesion probability map

After unified segmentation is applied to the brain, multiple images are saved to the provided directory. To be exact it contains 20 files, including the original MRI scan. Of these initially 3 files are used - warped gray matter tissue probability map (TPM), warped white matter TPM and warped CSF TPM. These TPMs are warped to match a default template provided by SPM toolbox. Warping is used because subsequent processing requires a comparison with a healthy brain half and/or standard template and for that the rough shape of the scans must align. This is part of the reason why this algorithm does not work for small lesions.

Next dialogue asks to provide directory containing MRI scan segments. This dialogue is thrown because this is the next piece of code to be executed after sectioning or skipping sectioning in the initial dialogue. This is the first of many optimization issues present in the script. So, in any case, selecting the directory with the segments throws a dialogue with the main settings selection for the algorithm.

This new dialogue has 6 fields to be edited. First one is for naming folder in which results will be placed, whose function is self-explanatory. Next is a selection whether to search for lesion is in the left hemisphere, indicated by providing character ”L”, or in the right hemisphere, indicated by character ”R”. This points out one more limitation of this algorithm neither can it automatically determine which hemisphere has the lesion, nor find lesions that span both hemispheres. Third field is for smoothing kernel full width at half maximum (FWHM) selection. Default suggested value is 8, which is used for performance/results reference later. Next field asks if whether the unaffected hemisphere should be used for lesion area detection. Fifth field is for selecting prior lesion probabilities. Last field is implicit mask for smoothing value which is 0 by default.

After gathering these values and doing some shuffling into a single class variable, subscript for feature extraction is run. Processing in this subscript starts with smoothing of all the elements to be used (TPMs and SPM12 template/PPM) applying the smoothing kernel FWHM value provided in earlier dialogue. Following more variable shuffling, brain mask from SPM12 toolbox is loaded. This mask is used to filter out noise that can appear outside of brain before and after some processing steps; for example: to remove data points appearing outside of brain after smoothing TPMs. Next up all required layers are made for the affected hemisphere. As an example, assuming that the left hemisphere is affected, three left hemisphere probability maps are created for gray matter, white matter and CSF each. First is the left hemisphere only, all other data points filters out, of the smoothed TPMs made by segmenting. Second is right hemisphere only, smoothed TPMs flipped to overlay the left hemisphere. This one is used only if selected in the previous settings dialogue, however is created either way. Third is left hemisphere of the smoothed SPM12 template/PPM.

Once all the TPMs and PPMs are ready feature maps for missing tissue (F_1) and abnormal tissue (F_2) are created. The missing tissue map provides information about areas where brain tissues are missing. To find this area a feature of SPM12 segmentation is used segmentation algorithms tend to classify chronic stroke lesions as CSF due to missing tissue voxels being assigned low gray or white matter probability values [22, 23]. So in the end the missing tissue area is obtained from the average of two image volumes using Eq. 5 and Eq. 6.

$$(CSF_{Affected} - CSF_{Unaffected}) * ((GM_{Unaffected} + WM_{Unaffected}) - (GM_{Affected} + WM_{Affected})) \quad (5)$$

$$(CSF_{Affected} - CSF_{Prior}) * ((GM_{Prior} + WM_{Prior}) - (GM_{Affected} + WM_{Affected})) \quad (6)$$

Both of these equations are used if user indicates to use unaffected hemisphere in previous dialogue. Otherwise only Eq. 6 is used. When using both equations, before saving result as missing tissue map both the equations results are averaged. Averaging is rationalized proving two points: using them as separate predictors would be sub-optimal since both volumes contain highly redundant information as both volumes are expected to have lesion values in same areas; averaging them retains the values of concordant voxels, while reducing the values of discordant voxels (e.g. false positives due to inter-hemispheric or inter-individual variability)[2].

Once the missing tissue map is created, next is abnormal tissue map (F_2). It provides information about abnormal tissue and is motivated by the observation that SPM12 segmentation tends to classify these tissues as gray matter due to the T1w signal intensities being similar to those observes in healthy gray matter [14]. Overall system is same as in missing tissue extraction, just uses different equations (7, 8). An example of resulting missing and abnormal tissue maps is in figure 2.

$$(GM_{Affected} - GM_{Unaffected}) * (WM_{Unaffected} - WM_{Affected}) \quad (7)$$

$$(GM_{Affected} - GM_{Prior}) * (WM_{Prior} - WM_{Affected}) \tag{8}$$

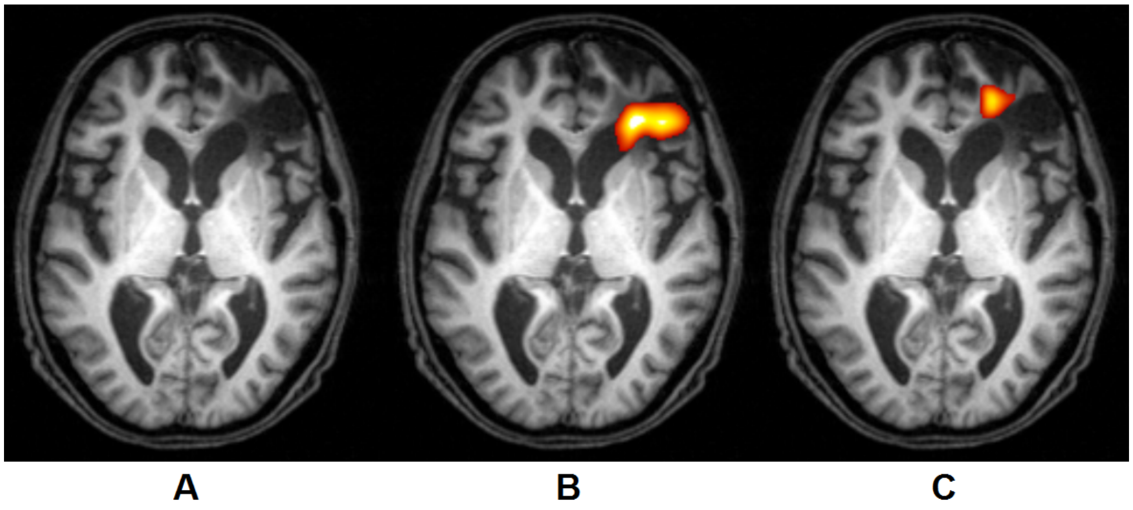


Figure 2: Patient MRI scan slice (A), with missing tissue map overlaid (B) and abnormal tissue map overlaid(C)

At the end of feature extraction subscript, both feature maps are saved to files along with MATLAB features magnitudes variable and their corresponding indexes variable. This is followed by classification subscript. By default this uses previously trained classifier to predict full area affected by the lesion [17]. Training was performed with 29 cases and tested on a single case left out of the training [2]. Prediction is executed utilizing MATLAB function "predict", supplying it with the trained system and features magnitudes variables saved in feature extraction subscript. This function computes the k-step ahead prediction returning, in this case, labels and posterior. However, these resulting feature maps are noisy and imprecise (Fig. 3B). To refine the results some post-processing algorithms are employed next.

Next dialogue to be thrown informs that delineation is complete and suggests applying post-processing. Standard "Y/N" choice is provided and if not performing any post-processing script exits. Results at that point can be found in `f1.nii`, `f2.nii`, `lesion_labels.nii` and `lesion_posterior.nii` files. If selecting to apply post-processing, next dialogue asks whether FWHM smoothing should be performed and what size smoothing kernel should be applied; default is 8. Then a dialogue for picking if the implicit masking should be used. Third dialogue is for clustering algorithm should minimum lesion cluster size algorithm be applied and what is the minimum size per cluster. This comes with a recommendation of 100 voxels per cluster, which is another reason why this algorithm, using default values, could not find small lesions.

These dialogues are thrown intermittently throughout the code that is being executed in the post-processing step. First to be executed is smoothing. Lesion probability map (`lesion_label.nii`) is processed according to the supplied by the user smoothing kernel FWHM and then thresholded to retain values in voxels with magnitudes above 0.25. This is intended to close gaps, smooth rough edges and degrade small isolated lesion clusters given by predictor [2]. This is followed by clustering algorithm, which is used to further remove small clusters of lesions. By default minimum cluster size is 100, which means if a total number of voxels in a cluster is less than 100, those voxels are removed from the lesion map.

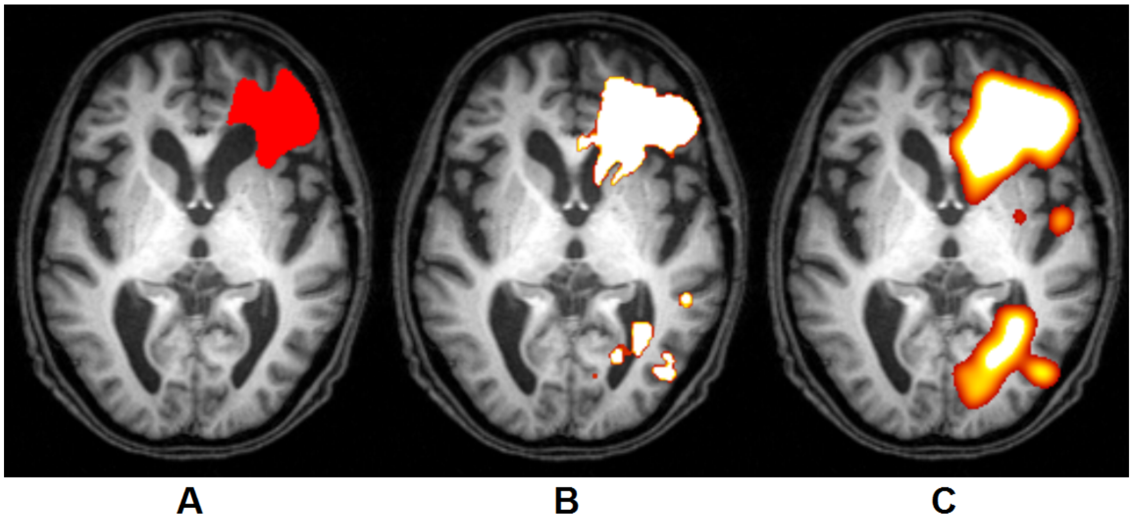


Figure 3: Griffis lesion delineation algorithm results: depiction of hand drawn lesion area (A), lesion area pre post-processing (B), lesion area after post-processing (C)

This concludes the run of Griffis lesion delineation algorithm with default values using his trained predictor system. For the first test case, upon visual inspection, it finds excessive amounts of areas affected by the lesion (Fig. 3C). Following subsection analyzes obtained results when running the algorithm "as is", how the algorithm can be improved and comparison between the two.

2.3 Default results, algorithm improvement and comparison between them

Griffis lesion delineation algorithm is neither accurate, nor is it optimized. Accuracy depends on the type of data fed to the algorithm and practice in picking settings like smoothing kernel FWHM. I call the script not optimized, because it often performs calculations that are irrelevant to subsequent calculations and even saves some of this extraneous data onto the hard drive. This significantly increases processing time and wastes storage space in above mentioned case all data generated by this algorithm for one patient takes 334 MB (including 8MB of initial MRI scan).

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{9}$$

The main metric used for these results evaluation will be Dice-Sorensen Coefficient, otherwise known as Dice Similarity Coefficient, or DSC for short [51], which shows the similarity between two sets of data. X and Y in Eq. 9 refer to the two sets of data. DSC is obtained by multiplying the number of overlapping points by two and dividing that by the sum of all data points in both sets. In analysis of lesion delineation algorithm results, first data set corresponds to manually delineated lesion area three dimensional matrix which has values of either 0 or 1, with 1 indicating that the lesion is present in that voxel; second data set is algorithm delineated lesion area three dimensional matrix, which is thresholded with, an arbitrarily chosen, 20% cutoff, meaning that values smaller than 20% of maximum value in all set are assigned 0 value and others are assigned 1. Since both sets are binary, with maximum magnitude of 1, resulting DSC values ideally should be equal to 1. That would mean that all data point all compared data points coincide. For analysis DSC itself in calculates for each MRI scan slice going vertically instead of one DSC value for all. This allows to identify slices where algorithm found lesion area accurately, and where it produced just false data.

For testing 124 MRI scans were prepared, however due to limitations of the algorithm majority were rejected. This algorithm due to smoothing involved and due to brains being unsymmetrical cannot find small lesions, it can find only lesion areas that damaged a significant portion of the brain. To pick out which brains have small lesions, their manually delineated lesion regions were used and MRI scans with total lesion area of less than 5000 voxels were rejected. Another feature for rejection is lesion location. This algorithm cannot delineate the lesion if it spans both hemispheres. That is why only lesions that affect single hemisphere were selected. After this filtering, 49 MRI scans were left and algorithm was run for all of them.

DSC results are plotted as a graph for each slice. Slices where there is no data in either set has no line as the DSC equation (Eq. 9) produces a division by zero in all plots there are slices with DSC of 0. This is due to the automatic algorithm finding larger of just different areas than in manually delineated case (Fig. 5). These plots also have red dashed line representing relative size of manually delimited lesion area in that slice and green dashed line showing relative size of automatically delimited lesion area in that slice. These lines are used to indicate MRI scans which in the end got seemingly good DSC values in some slices, but that is only due to automatic algorithm delimiting huge areas of brain as affected by lesion.

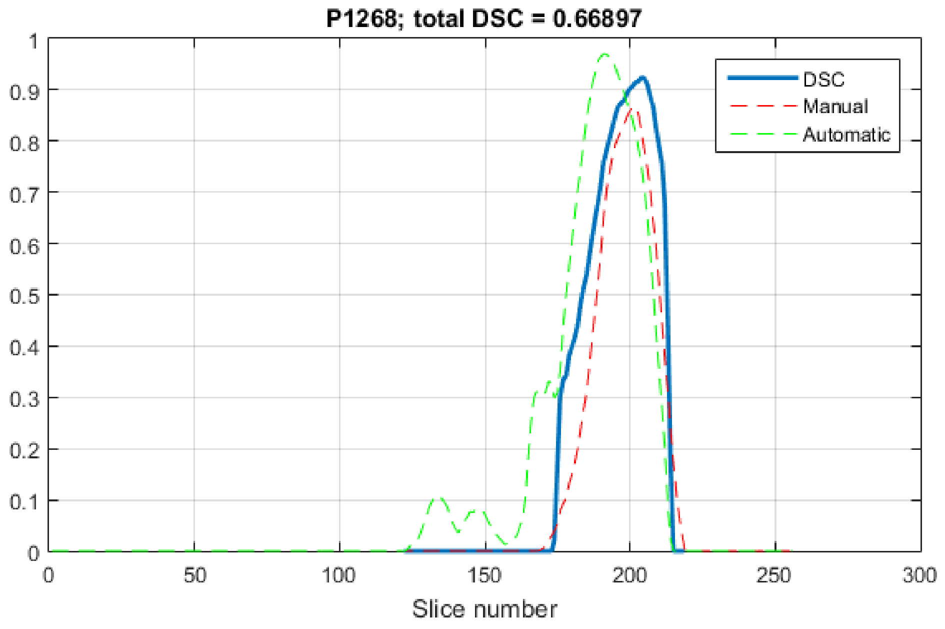


Figure 4: Best attained per slice DSC results after running algorithm for 49 patients.

None of the MRI scans in any of the slices attained DSC value of 1, which would be the ideal case, however in one case DSC of 0.9 in a slice was attained (Fig. 4). 13 more got DSC values around 0.8 for

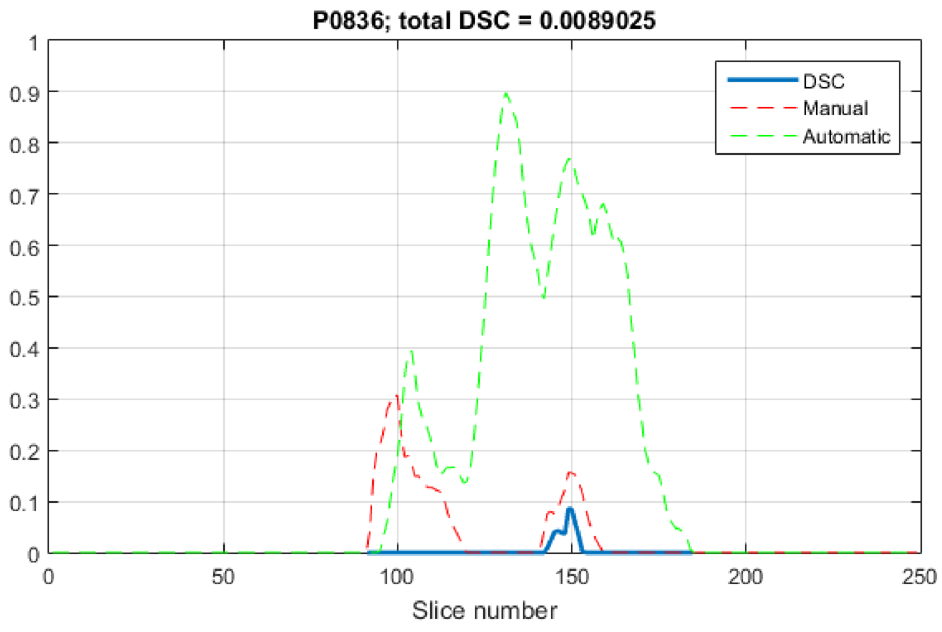


Figure 5: Worst attained per slice DSC results after running algorithm for 49 patients.

a slice, which shows that this algorithm can delineate some lesions with a degree of precision. However, these 14 patients constitute only 29% of the test cases. DSCs for 3 patients do not reach magnitudes of even 0.4 per slice. That is when algorithm was unable to determine where precisely is the lesion and just marked large swaths of brain as affected by lesion. Patient MRI scans in between these two extremes get some decent DSC values in some slices because of same reason algorithm delimiting large areas as affected by lesions and marked voxels in sets at particular slice just happen to coincide. In this case total DSC value shows that algorithm failed. As an example, for patient P0743 (Fig. 6) one of the slices has DSC over 0.7, but automatic algorithm delimited large areas of the brain as lesion even though there was none, which brings total DSC to 0.17. This same issue is present in few of the results, which looking at the per slice DSC would imply "good" result. In case of patient P1712 (Fig. 7), maximum per slice DSC reaches 0.8, but the total DSC is only 0.28 implying a bad result. And really in that case automatic algorithm finds large areas of lesioned brain where there is none (Fig. 8). In the end there are cases where automatic algorithm fails completely (Fig. 5). However, this case is illustration of algorithm being incapable of detecting small lesions (Fig. 9). Analysing MRI scan and manual lesion delineation one can see that only small areas scattered around the right hemisphere are affected by the lesion. That is the worst case for the algorithm lesions that are small and on edges of the brain. After smoothing and removal of areas outside of presumed brain area information becomes indistinguishable.

So in the end, according to the algorithm author, lesion delineation with total DSC of 0.60 or higher can be considered good [2]. By this estimate out of my tested 49 cases only 7 delineations are "good" mere 14% of cases. This is likely due to predictor system that was trained by the original author just does not work with the MRI scans I have tested with. By training your own predictor system it should be possible to increase the accuracy and improve DSC scores, but that is not likely work later, when running for other cases. Author had 30 MRI scans to work with, used 29 for training and 1 to test the

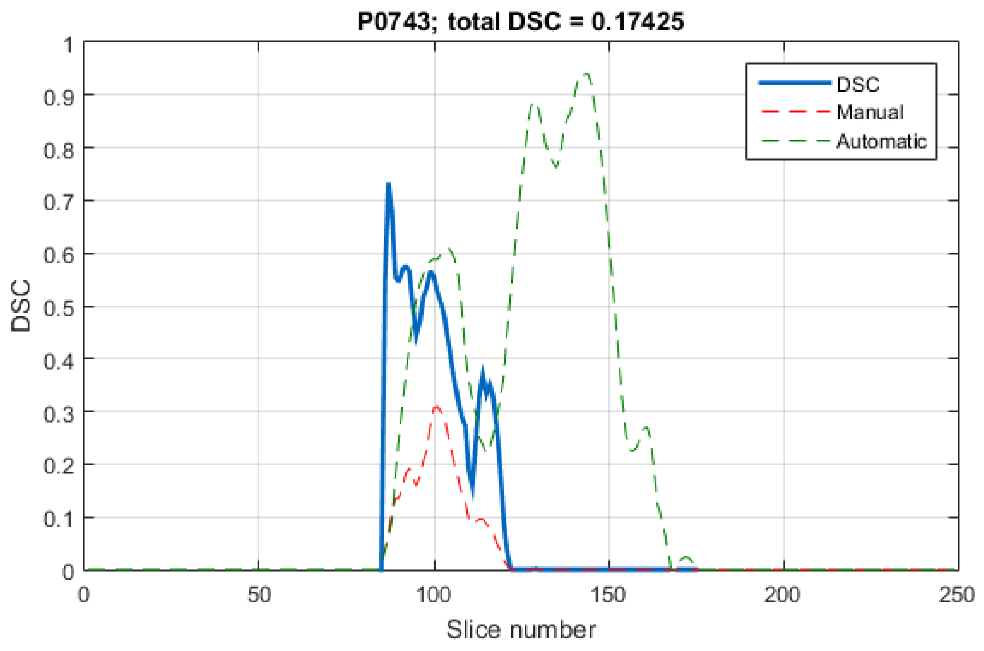


Figure 6: DSC plots showing results of lesion delineation for patient P0743.

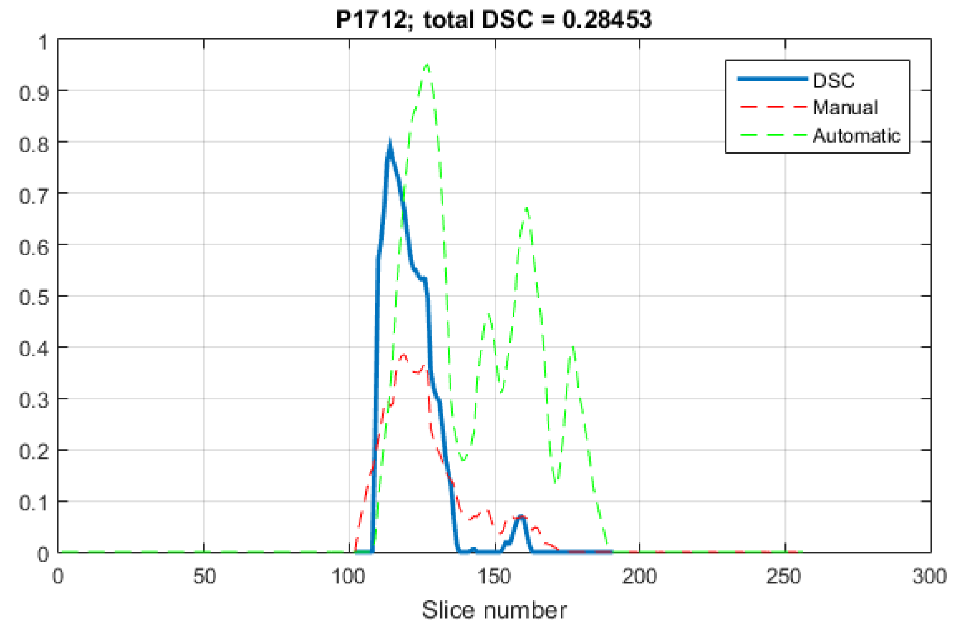


Figure 7: DSC plots showing results of lesion delineation for patient P1712.

obtained prediction system. However, later on when testing the system he, presumably, applied it to the same 30 MRI scans. This is why his results are better 66% of the lesion delineations are "good" [2]. This is to be expected as the system was trained on those MRI scans. And that is why predictor trained there does not work in many of 49 cases I have tested on, since these vary greatly by placement, size, clustering and intensity.

However the results and overall algorithm operation can be tuned and improved editing the script and default values. So at this point I have created an edited version of authors original `lesion_gnb.m` scrip, naming it `et_lesion.m`. Main focus of this script is increase number of things done automatically;

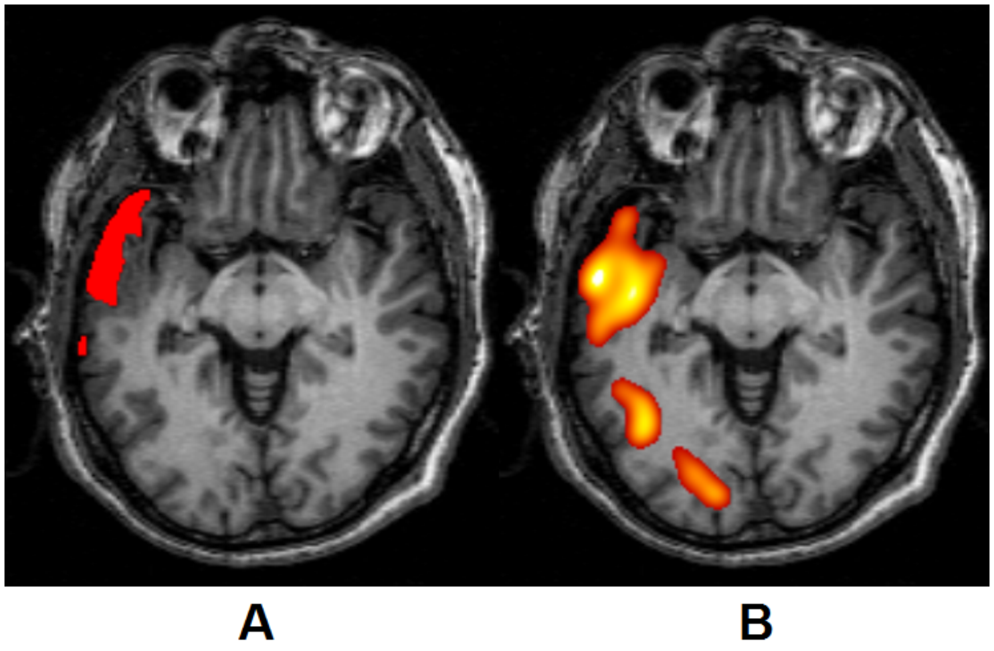


Figure 8: Manual delineation (A) comparison with result of automatic delineation (B) for patient P1712.

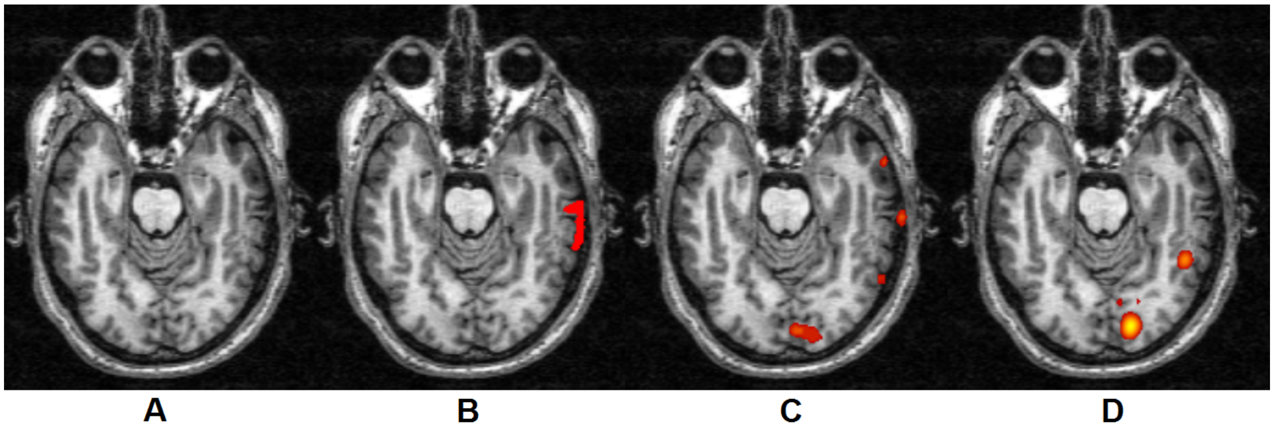


Figure 9: Patient P0836 MRI scan (A) manual delineation (B) comparison with result of automatic delineation (D). Also showing found missing tissue TPM (C).

make the script run for multiple patients without user interaction. This was especially appealing, as even without any user interaction, using default parameters and scripts, `lesion_gnb.m` for the 49 test cases runs around 3.76 hours (Table 1). `lesion_gnb.m` requires a variety of settings on each MRI scan, however only one depends on the MRI scan itself. As such they can be set at the beginning and reused for all cases. The one case where this does not apply is indication which hemisphere of the brain is lesioned. My code determines this by MRI scan filename last symbol in filename is either "L", signifying that the left hemisphere is lesioned, or "R", signifying that the right hemisphere is lesioned. Patient MRI scans are taken from a single directory, which is expected to contain only patient MRI scans. Results are placed in one directory, set at the beginning, which in the end contains subfolders named after patients MRI scan filenames into which all resulting segmentations, feature maps, delineations are placed.

In terms of saving storage space, 8.4 GB of space were saved (Table 1), roughly 40%. However, storage space saving was not the real focus of optimization, more of a side product of improving algorithm execution speed. Most time consuming part is MRI scan segmentation in to TPMs using SPM12 toolbox. Complex segmentation algorithms strive for high accuracy [5], thus all the lengthy repeated calculations that are used. Also big contributor to the long processing times is that these algorithms cannot fully utilize multiple-core/multiple-thread CPUs. Usual load during processing on Intel i5-4460 4 core, 4 thread processor by MATLAB process most of the time is 40%, and 80% in some parts. So segmentation is a time consuming process, which means the amount of processing done during segmentation overall should be minimized.

Originally Griffis lesion delineation algorithm during segmentation saves to storage all native space TPMs gray matter, white matter, CSF, bone, soft tissue and air/background. In addition it also saves warped modulated and unmodulated TPMs of gray matter, white matter, CSF and bone. Each TPM requires, not equivalent, but significant amount of processing. In the end, lesion delineation script utilizes only warped unmodulated TPMs of gray matter, white matter and CSF, thus first improvement of mine is generating, and saving, only the required warped TPMs gray matter, white matter, CSF. When saving only these, segmentation function throws a warning in the cleanup process stating that it cannot be performed. Cleanup can be performed when at least one of each TPM is generated. Thus, my code also saves native space TPMs for bone, soft tissue and air/background. Initial test showed that cleanup is not necessary - tested by simple differences comparison: subtracting TPM generated by Griffis configured segmentation algorithm from TPM generated by segmentation configured by me. Maximum absolute amplitude difference equals 0, meaning there is no difference to the segmentation results from cleanup algorithm. Omission of cleanup and calculation of those extra native space TPMs would save 2 minutes per patient. However, upon manual inspection of few other cases it became apparent, that sometimes TPMs returned from segmentation have false positive values in the air/background, bone, soft tissue regions. To avoid these issues, i kept the cleanup process. In the end, segmentation configured by me, running code for all 49 patients, saves on average only 19 seconds (Table 1) of processing time per patient.

Table 1: `et_lesion.m` vs `lesion_gnb.m` execution times and results size comparison.

	average segmentation time, s	average other processing time, s	average full processing time, s	full run time, h	results size, GB
`et_lesion.m`	245	4	249	3.58	13.3
`lesion_gnb.m`	264	12	276	3.76	21.7

More time can be saved by implementing a feature that is hinted in the comment in original code, but not actually used testing if smoothed prior/template already exists, before attempting to smooth it. In feature map calculation equations (Eq. 6 and Eq. 8), mentioned before (Section 2.2), prior TPM, which is referred to as template in parts of the code, is smoothed with smoothing kernel FWHM defined by the user. As the original non-smoothed prior file used for calculations does not change, smoothing could be performed once per FWHM value and, upon code reruns, previously smoothed TPM could be reused. Check if previously smoothed TPM exists and subsequent omission of smoothing saves 8 seconds per patient in my optimized algorithm execution time changes from 12 seconds down to 4 seconds (Table 1). This 4 seconds average is attained, when all runs already had smoothed prior TPM and were skipping smoothing process. Average time of the `lesion_gnb.m` indicates how long processing would take if prior TPM would not be smoothed with a particular FWHM, effectively, showing how long it would take to when running script for the first time.

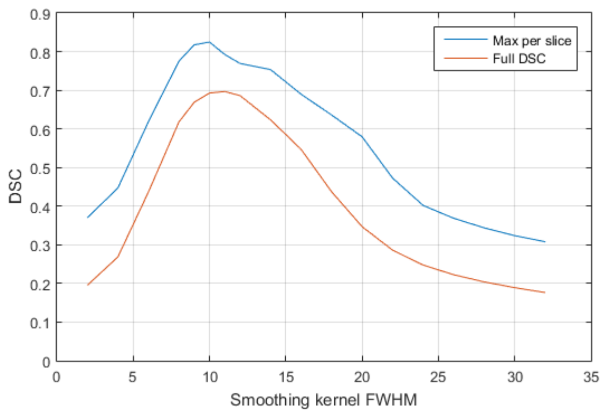
Further script optimization has comparatively small impact on processing speed or storage space taken and mostly deals in increasing code readability and minimizing redundancy in variables. However, other parameters can be optimized. My focus was on 3 of them smoothing kernel FWHM, prior coefficient and thresholding limit of post-processed lesion delineation. Both, full DSC value and maximum per slice accuracy, values are used for result comparison purposes. Higher priority is given for full DSC as it shows full lesion delineation accuracy rather than maximum partial, per slice, accuracy.

First I started with most obvious one smoothing kernel FWHM. This affects the three dimensional Gaussian blur of each voxel. Test were run for four patients P1857, which had "good" lesion delineation using default values; P1712, which had "good" maximum per slice DSC, but full DSC of just 0.28; P0836, whose lesion delineation had full DSC of 0.0089 worst of all test cases. FWHM values were taken in range from 2 to 32 with a step of 2, except for P1857 and P1712 for whom FWHM of 9 and 11 are included to better pinpoint curve peak. This gives a sizeable range which also includes the default FWHM value of 8. Resulting DSC dependence on FWHM plots show the possibility to find an optimal FWHM value (Fig. 10).

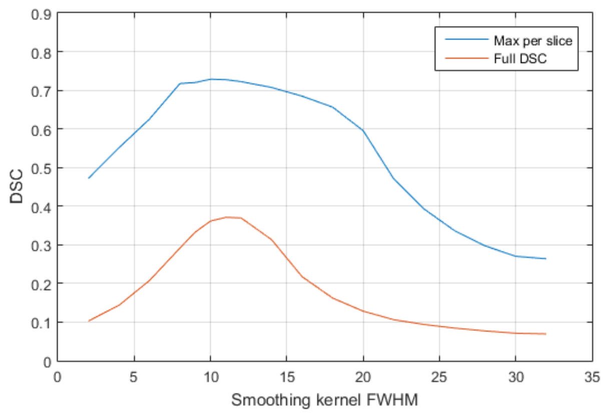
Analyzing first case, P1857 MRI scan, already shows that FWHM of 8 does not give the best possible results (Fig. 10A). Total DSC peak is at smoothing kernel FWHM value of 11. Maximum per slice DSC is at its peak at FWHM of 10 and quickly dips below 0.8 at FWHM 11. Thus, this gives two values to pick from. Full lesion area DSC more indicative of overall lesion delineation accuracy, thus FWHM of 11 can be stated as being best in the current testing setup. Using FWHM of 11 instead of 8 increases full DSC by 11% from 0.6181 to 0.6966. Theory of FWHM of 11 being better than 8 is consistent with curves obtained for P1712 MRI scans (Fig. 10B). Maximum per slice DSC curve peaks at FWHM of 10 and full DSC curve peaks at FWHM 11. Few extra checks were carried out best DSC values vary between FWHM 10 and 11, more often than not, 11 being better one. Thus `et_lesion.m` will be using smoothing kernel FWHM of 11.

Further tests show that no FWHM value can help find "good" results in cases where the algorithm fails to find decent results using default parameters. In case of P0836 MRI scan, at smoothing kernel FWHM of 6 maximum per slice DSC does reach values greater than 0.5, however overall DSC is still less than 0.05. As mentioned before, delineation fails here, because this patients MRI scan has mostly small lesions and they are at the edges of the brain. For my untrained eye they were mostly invisible. Even looking at segmentations, though somewhat visible knowing where the lesions should be, they are hard to distinguish (Fig. 11), thus it is no surprise that blurring of the image even further obfuscates the useful information.

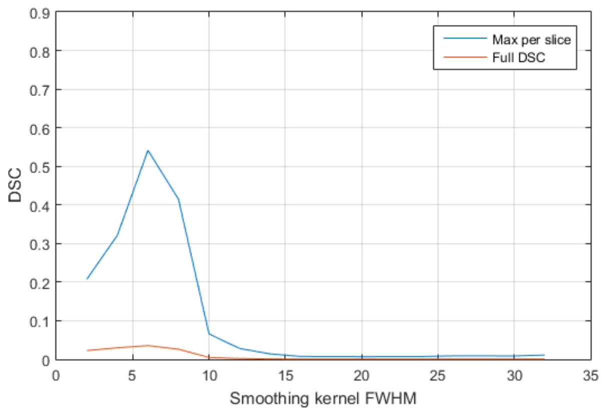
Next tested parameter set by user that could improve the result is prior probability coefficient. This value is used to make a two element vector, which is then used as one of parameter in prediction system.



(a) P1857



(b) P1712



(c) P0836

Figure 10: FWHM impact on DSC testing results for three patients; (a) - ”good” lesion delineation, (b) - erroneous lesion delineation, (c) - completely failed lesion delineation.

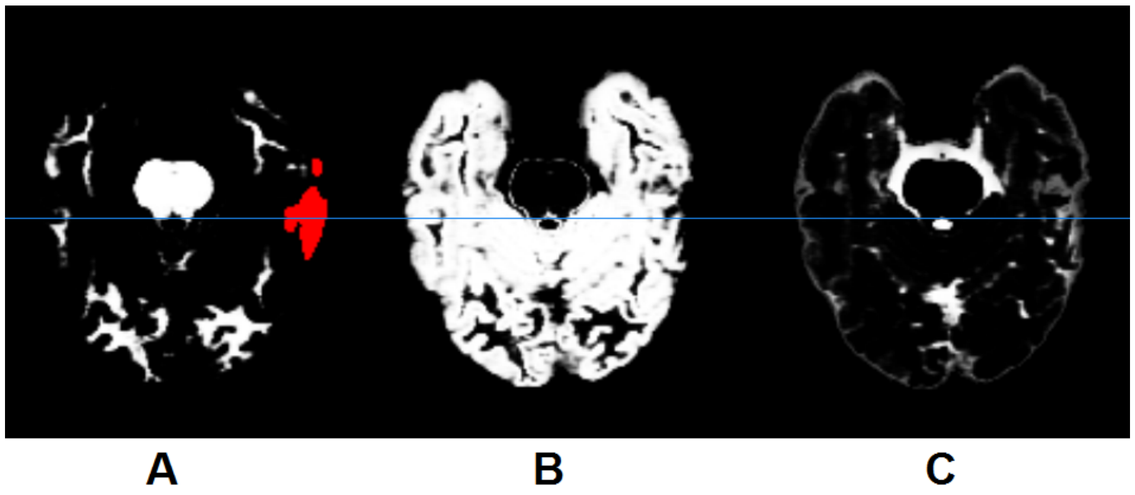
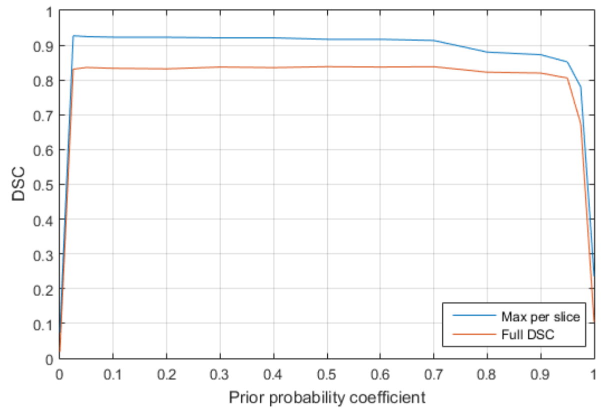
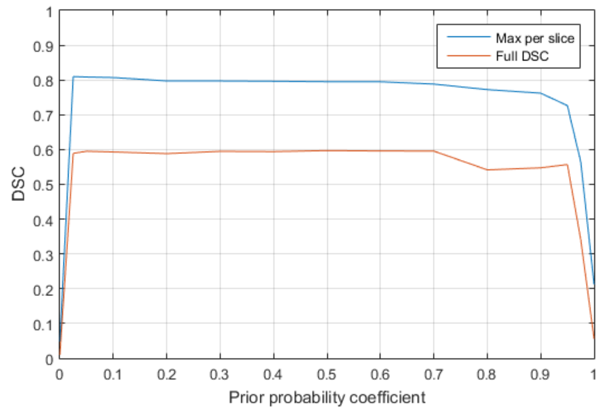


Figure 11: Most obvious slice of segmentations for patient P0836 MRI scan with emphasis on lesion area: gray matter TPM with manually delimited lesion area overlaid (A), white matter (B), CSF (C)

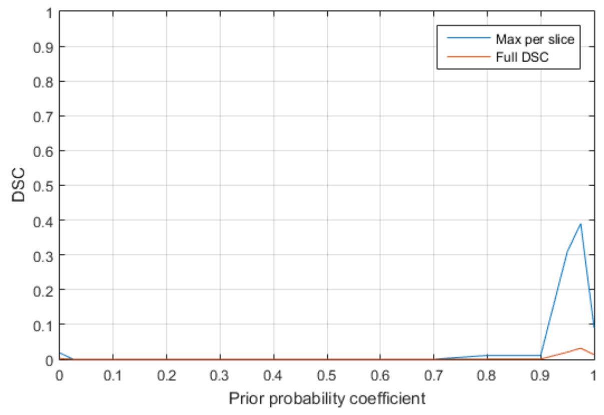
First value in the vector is 1 minus the given coefficient, second value is coefficient itself. Analyzing the source code for the script I found that default value for prior probability coefficient of 0.1064 is used by him. However, he suggests 0.5, which is the value I used in the control run testing how his script performs for my given 49 patient MRI scans. Testing of prior coefficient is performed on same three patients MRI scans from smoothing kernel FWHM impact tests, plus P1419 who had somewhat different curves. FWHM value for these tests is 11 in hopes to find a way how to further improve the result, given best found FWHM. Range of prior probability coefficients is from 0 to 1 with step size of 0.1, including extra points at both ends (0.025, 0.05, 0.95, 0.975) to further determine the point of DSC value drop off. Plots themselves show maximum per slice DSC and full lesion delineation DSC dependence on prior probability coefficient. Resulting curves are in Figure 12.



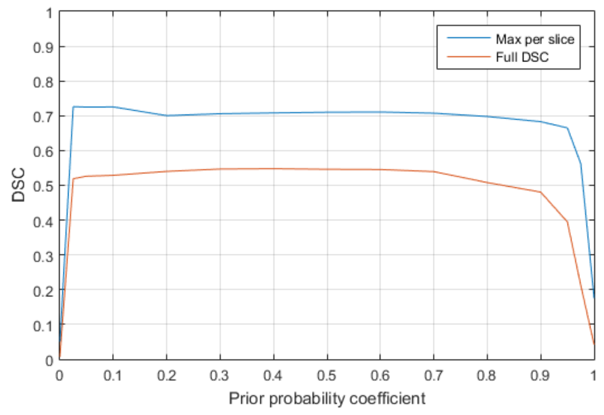
(a) P1857



(b) P1712



(c) P0836



(d) P1419

Figure 12: Prior probability coefficient impact on DSC testing results for three patients.

Firstly, prior probability coefficient is not good at 0 or 1. For example for patient P1857 using 0 as prior probability coefficient, gives empty lesion delineation no results; using 1 as prior probability coefficient, gives all hemisphere as lesion delineation area too many results. Thus, values in between must be taken.

Curves themselves are quite stable, having small results variation in a wide range, from 0.025 to 0.7 prior probability coefficient values. For patient P1857, with "good" results, shown here full lesion DSC varies from 0.8311 to 0.8383 maximum difference of 0.0072, which does not give a significant difference. In case of patient P1712 maximum difference is 0.0087 still not a significant difference over a wide range.

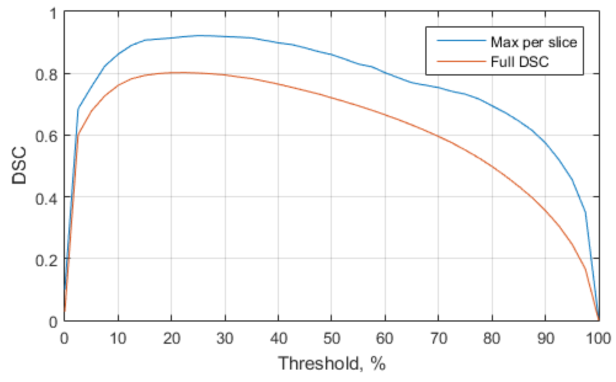
This insignificant difference persists with many other patients who got "good" or at least decent lesion delineations running with default parameters. However, some patient have more pronounced "better" regions where either maximum per slice DSC or full lesion delineation DSC values are higher than in other regions (Fig. 12D). Full DSC and per slice DSC regions do not overlap in this case, however, based on same logic as before during FWHM testing, full lesion delineation DSC values are more important as such values should be picked from that range. As such in `et_lesion.m` prior probability coefficient value is maintained 0.5 the same as in default case, since it does fall in the "good" range.

Finally, prior probability coefficient does not help with the case of patient P0836, where lesion delineation is complete failure. In the presumed "good" range, 0.025 to 0.7, all values, both per slice and full DSC, are flat zeroes no result. Some data did happen to overlap at prior probability coefficient value of 0.975, but even then the DSC values are bad and as such this data is not considered to be influential on the decisions made in picking prior probability coefficient to be used in `et_lesion.m`. And as just a test if specially tailored FWHM and prior probability coefficient values could produce a decent result another quick test was run. Maximum per slice DSC of 0.7 is attained at 0.05 prior probability coefficient value, which coincides with other results, implying better maximum per slice DSC in lower ranges of prior probability coefficient. But, in the end maximum full lesion delineation DSC was still only 0.0398.

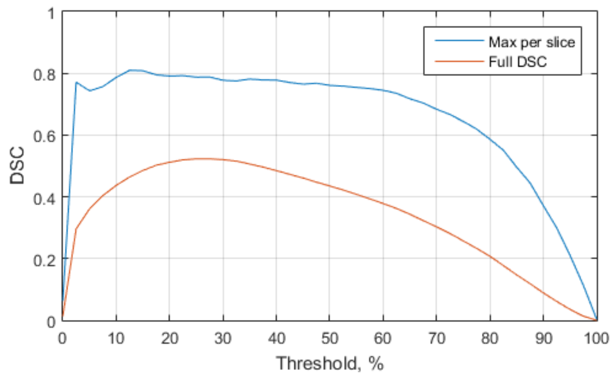
Third, and final, parameter whose impact on the resulting lesion delineation I have tested is the threshold used to convert results to binary values from floating point ones. Original `lesion_gnb.m` script makes binary valued lesion delineation, but after post-processing this delineation once again contains gradients based on given smoothing kernel FWHM. Before DSC calculations this smoothed lesion delineation is once again converted back in to binary values. In the algorithm, threshold value is dynamic and varies per patient. Up to this test the value was 20% of maximum value in all lesion delineation. For example, if maximum value is 1 then all values bellow 0.2 assigned 0 and all value above and including 0.2 would be assigned 1. In practice, however, maximum value is never 1 and in some cases (patient P0836 failed delineation) as low as 0.18 magnitude.

Test was performed by changing threshold percentage in full range, from 0% to 100%, with a step of 2.5%. All other test parameters are kept the same as in default algorithm, except for FWHM which is set to be 11. Test objects are post-processed lesion delineation of all 49 patients. For result accuracy evaluation once again maximum per slice DSC and full lesion delineation DSC was used.

Firstly, I checked the results for the same three patients as used in precious tests as they represent three distinct results of lesion delineation (Fig. 13). Patient P0836 is not included in the figure, because, as seen in previous test, using FWHM of 11 and prior probability coefficient of 0.5 resulting DSCs are 0. Thus thresholding does not change this result as there is no overlapping data to begin with. And only point where DSC rises above zero is at 0%. This is, because with a threshold of 0% all data set becomes ones, due to all values at or above threshold value being set to 1, and this means that inevitably some data does overlap. This feature of the algorithm is noticeable for all patients everyone have same but small DSC values at threshold of 0%. Inversely, due to the algorithm both DSC values are near zero with threshold at 100%, because, in theory, only one voxel, with the highest magnitude, would be in the



(a) P1857



(b) P1712

Figure 13: Thresholding point impact on DSC testing results for two patients from before.

comparison set.

Other two patients do have peaks in their curves implying that there is an optimal value for thresholding. For patient P1857 this peak is at 22.5%. Comparing this to the preciously used threshold of 20% this gives an increase in overall DSC of $2*10^{-4}$, which is not a significant difference. Slightly higher difference is in the case of patient P1712: optimal threshold is at 27.5% and magnitude difference is 0.01 (better result by 2%). These two optimal thresholds do not coincide, as such, different approach is necessary to get best case threshold.

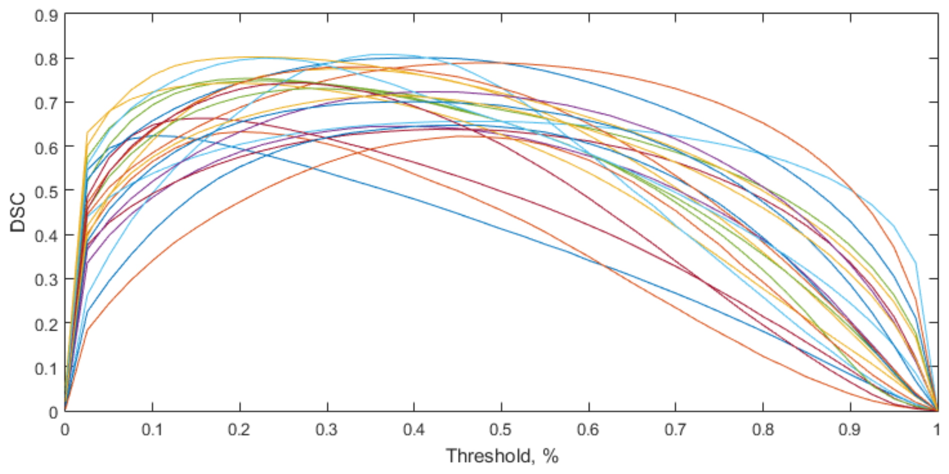


Figure 14: Curves of 24 patients full lesion delineation DSC dependence on thresholding point.

DSC dependence on threshold calculations are performed for all patients, however not all results should be considered further. Results that do not, at any point, reach "good" values [2] are omitted. This leaves 24 patients half of all patients. However, the resulting curves still have highly varied peaks (Fig. 14). To gleam a possible result I have averaged results for these 24 patients for each threshold and looked for a peak in the resulting curve (Fig. 15). Maximum magnitude is reached at 32.5% threshold, which, comparing to magnitude attained using 20% threshold in previous tests, gives an increase in DSC of 0.025 (3.6%).

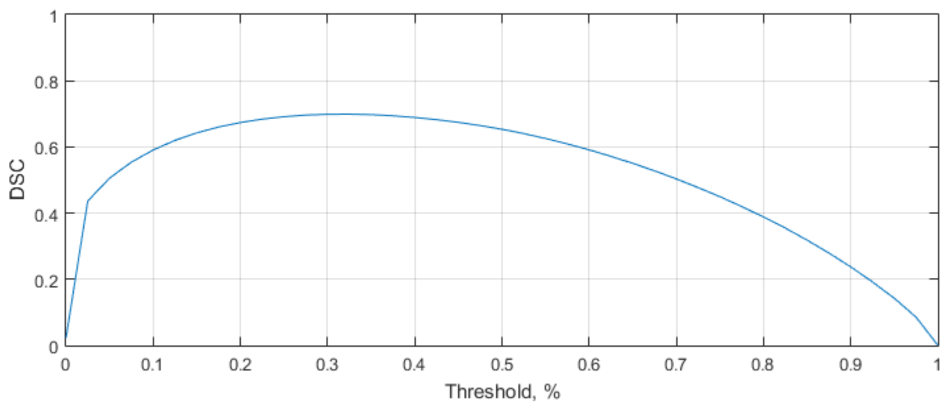
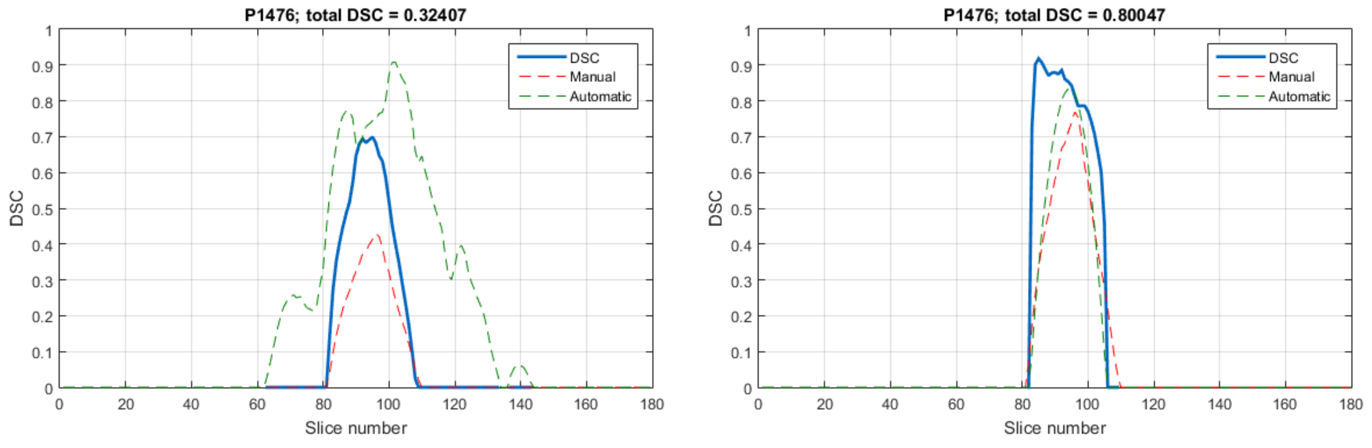


Figure 15: Curve of averages of curves in figure 14.

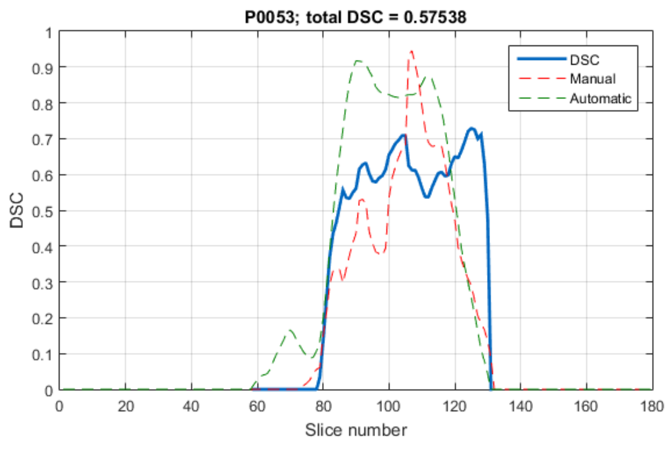
Running my modified `et_lesion.m` script and changing two parameters in calculations did improve end results. For results comparison all patients full lesion delineation DSCs were used. In roughest terms, DSC improved by 6.28, meaning an average increase of 0.128 DSC per patient. Largest increase in lesion delineation DSC was for patient P1476, with the DSC increase of 0.476 and also maximum per slice DSC increase of more than 0.2 (Fig. 16). This significant increase appears, because the algorithm is able to effectively find the actual area of lesion during feature extraction and after prediction, lesion delineation is similar to the manual delineation. For 44 patients DSC did increase, by some amount. For five patients lesion delineation DSC fell in comparison to DSCs obtained with default parameters. Largest DSC loss was for patient P0053 with magnitude of 0.05 (Fig. 17). This loss is, due to script now giving significantly smaller lesion volumes, but in this case was not more precise than with default parameters. For patient P0056 lesion delineations total labeled lesion voxels by `lesion_gnb.m` is 65 141, number of voxels labeled by `et_lesion.m` is 27 344, as also seen in graphs comparison. These smaller lesion volumes are also what give better results in case of other patients lesion delineations.



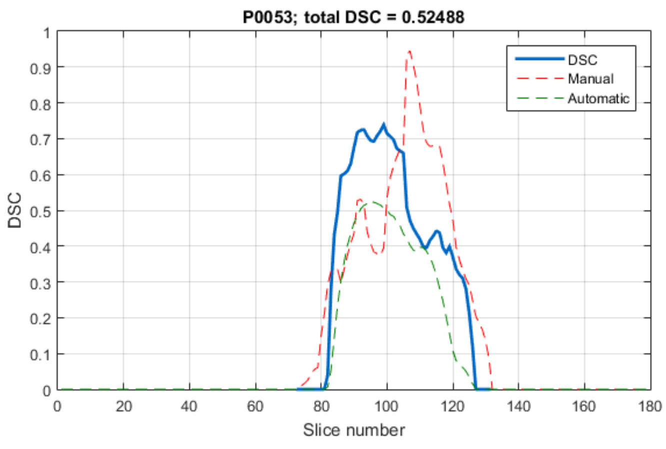
(a) (b)

Figure 16: DSC graphs for patient P1476 obtained running `lesion_gnb.m` (a) and `et_lesion.m` (b).

Total number of voxels marked as lesion area in `et_lesion.m` decreased by at least 25.9% for every patient; on average, decreased by 61.6%, with a maximum decrease of 93.2%. This decrease is mostly



(a)



(b)

Figure 17: DSC graphs for patient P0053 obtained running `lesion_gnb.m` (a) and `et_lesion.m` (b).

caused by thresholding level increase, but is also affected by prediction algorithm working differently when given slightly different feature maps that are influenced by new smoothing kernel FWHM. This decrease in lesion delineation volume gives higher precision, better DSC, in cases where extracted missing and abnormal tissue probability maps manage to find cores of lesions. Otherwise prediction system further muddles the information marking large swaths as lesion area and at that point removal of voxels decreases DSC.

So in the end, `et_lesion.m` manages to get ”good” [2] lesion delineations of 21 patients, compared to 7 obtained using script with default parameters. It is still only 43% of patients in the specially selected test group giving good results. As such it is still not usable for general application on any MRI scan and I would like to implement some self-evaluation algorithms that would provide the user with an indication of what accuracy in the result should he expect.

3 Algorithms attempting segmentation in 2D images

3.1 Image Segmentation tutorial

Started off by looking what’s available online as far as implementation in MALTAB is concerned. Specifically for lesion delineation there is few options, and fewer that implement neural networks. The ones I look at don’t really use neural networks I guess, but whatever.

So first off was image segmentation tutorial [1]. And really that is what I went most in depth with. This shows region growing methods to find nickels and dimes in the image.

It starts off by finding minima in the image. These points are grown using binary images. These are obtained by thresholding since in the grayscale image histogram clearly seen that some objects have clearly higher luminosity than others. In this case the coins have higher luminosity than the wooden background.

Once the regions are obtained they then are classified using overall blob area whether area is larger

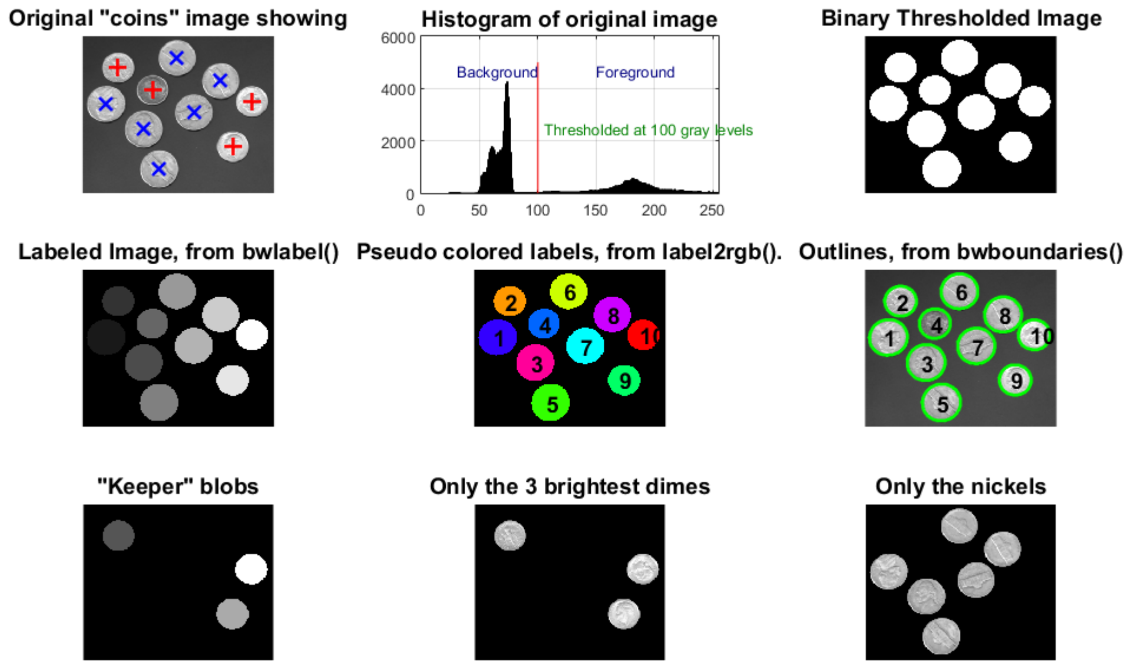


Figure 18: Results shown by the algorithm

or smaller than a pre-picked value. This part could be made with a neural network instead. Though in either case algorithms would fail if other image provided would be from a different distance and neither the hardcoded size thresholds, neither trained neural network wouldn't be able to say with certainty whether this is nickel or dime. So this points out that area alone is not sufficient feature for neural network even in this simple case.

3.2 Tumor detectors

Of the ones that worked theres two - Automatic segmentation of brain tumor in MR images[3] and, seemingly a rip-off from aforementioned, Brain tumor detection from MRI images using anisotropic filter and segmentation image processing[4]. As for why I called it a rip-off, second one is two years later and uses identical customer helper function, but, again, whatever.

In theory these try to find tumors in the brain so I guess it is no surprise that neither first one (Fig. 19) nether the second one (Fig. 20) is able to find lesion area. So I moved on.

4 Feature extraction for neural network

This is where I spent most of my time. Created 4 functions that would be useful if I will get stupid enough to ever go back to this approach. First one is `NNCP_Image_Segmentation_edgetech.m` customized version of Image Segmentation Tutorial mentioned before. And in the end creates three structures that I intended to use as inputs when training neural network. These structures contain largest area blob definition as given by MATLAB function `regionprops`. That gives a lot of numbers (Fig. 21) which is good when going for precision in neural networks, not so much when thinking of

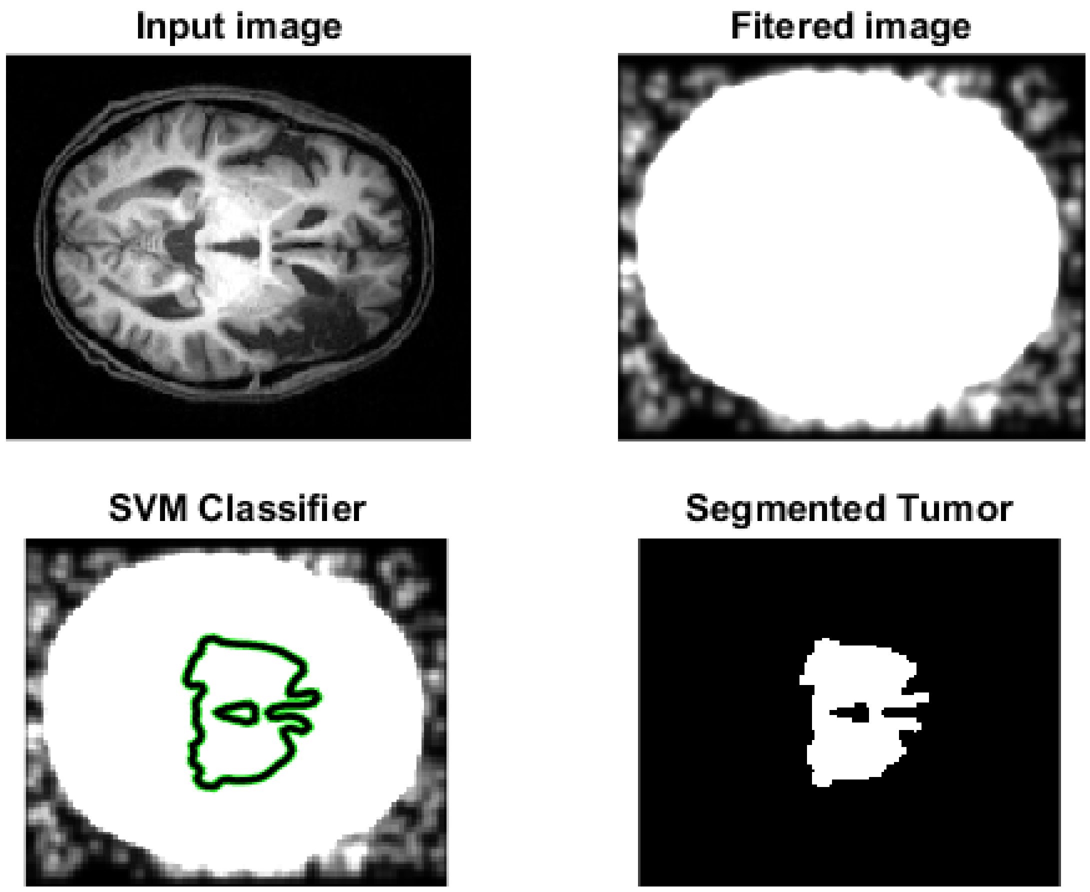


Figure 19: Older tumor detection algorithm[3] results for lesion detection.

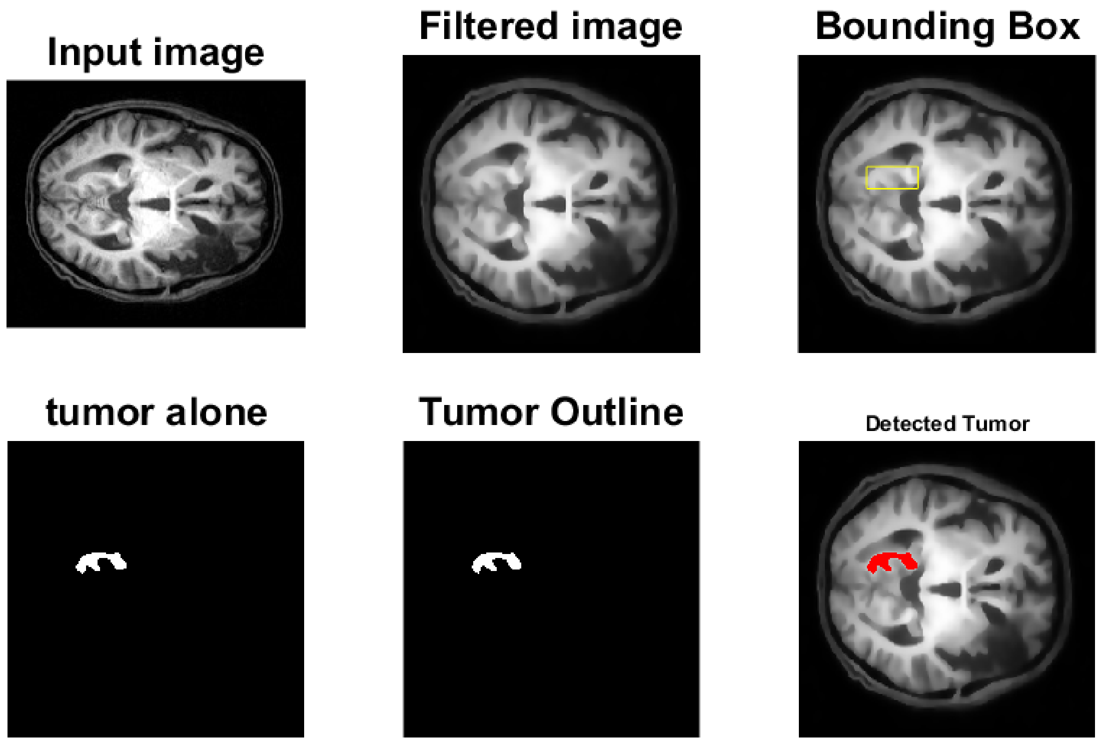
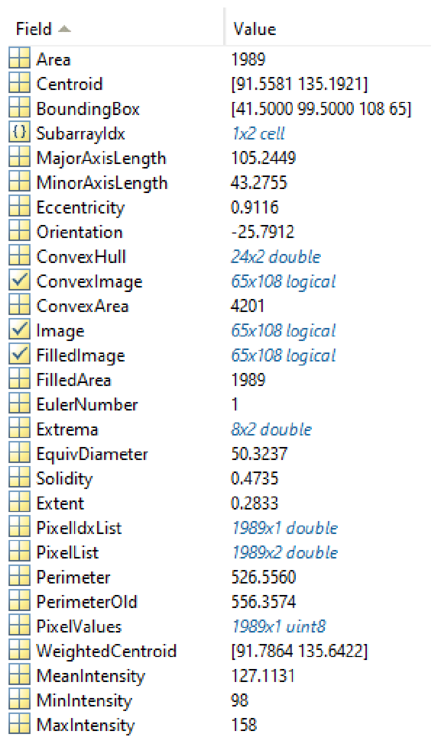


Figure 20: Newer tumor detection algorithm[4] results for lesion detection.

training speed.



Field	Value
Area	1989
Centroid	[91.5581 135.1921]
BoundingBox	[41.5000 99.5000 108 65]
SubarrayIdx	*1x2 cell*
MajorAxisLength	105.2449
MinorAxisLength	43.2755
Eccentricity	0.9116
Orientation	-25.7912
ConvexHull	*24x2 double*
ConvexImage	*65x108 logical*
ConvexArea	4201
Image	*65x108 logical*
FilledImage	*65x108 logical*
FilledArea	1989
EulerNumber	1
Extrema	*8x2 double*
EquivDiameter	50.3237
Solidity	0.4735
Extent	0.2833
PixelIdxList	*1989x1 double*
PixelList	*1989x2 double*
Perimeter	526.5560
PerimeterOld	556.3574
PixelValues	*1989x1 uint8*
WeightedCentroid	[91.7864 135.6422]
MeanIntensity	127.1131
MinIntensity	98
MaxIntensity	158

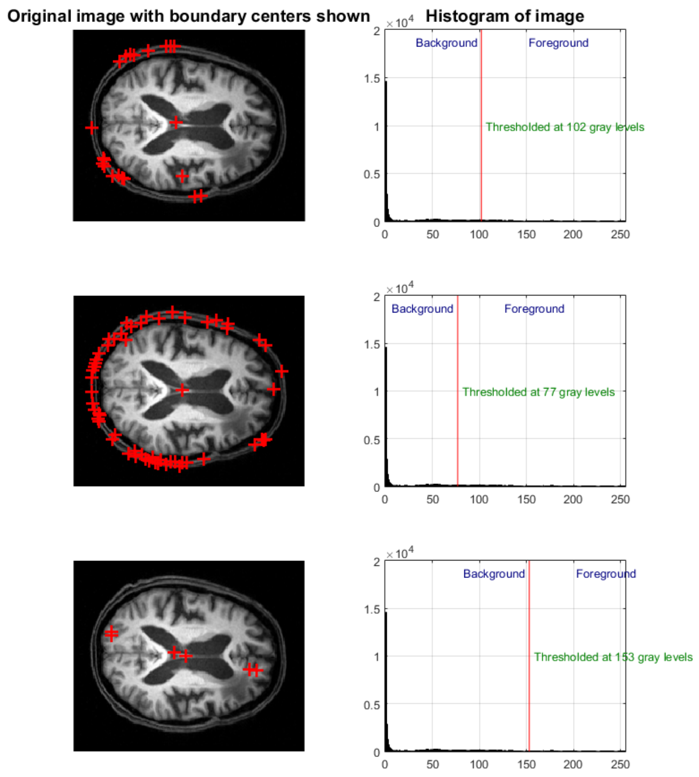
Figure 21: My feature extraction/segmentation results.

As to why three structures it's because I threshold binary image at three different normalized levels - 0.3, 0.4 and 0.6 - just some random numbers. Did not give much thought to these, just eyeballed them since in the end brain grayscale image is not nearly as clear cut as with coins on a table. So these threshold give varying blobs in cases which is good enough. In this case I'm thinking of one case in particular that I noticed while testing out: at one threshold level this large lesioned area is within the blob; at another level it is entirely outside; so I figure neural network would see this sudden disappearance of large blob area and would think that that is due to lesion.

Anyway, I made the script be able to optionally draw some graphs (Fig. 22). These show blob centers on the left column, image histogram with threshold level depicted in the middle column, blob boundaries and numbers in the right column. In this case I show off the idea I had about the thresholds making blob encompass the lesion or not.

Other functions are variable/data processing shifting around and overall repetition speed up and automatization. Like the second one: `NNCP_ImageCycling.m`. This one in essence takes a .nii file, which is 3D, and makes a bunch of 2D slices saving those as separate images (thanks to `nifti2slices.m`. Then it goes through each of those slices and sends them off to `NNCP_Image_Segmentation_edgetech.m` to extract features. It concatenates those features for all slices it made and sends them on to whoever called this function. There's other bells and whistles that ease the life but, once more, whatever.

Third function, which time wise came to be first of these four, is: `nifti2slices.m`. Returns image data of specified slice or slices back to the function that requested it. Lots of fun with automation, but, effectively, ineffective use of time.



Original image with boundary centers shown
Histogram of image
Outline bwboundaries()
Background
Foreground
Thresholded at 102 gray levels
Thresholded at 77 gray levels
Thresholded at 153 gray levels

Figure 22: My feature extraction/segmentation results.

And lastly the intended front-end function: `NNCP_ET_NN.m`. This is what initiates feature gathering from, at this point, 6 .nii files and then parses them into a double type values matrix, since that is what I though neural network trainers would want. Also it generates the goals vector, currently gave it values by visual inspection, but I might add script for getting values from the lesion .nii files that I have.

And at this point I hit `nntool`. At best getting error that it does not like my inputs matrix (Fig. 23). I formatted data, input and output, like for apples and pears from way back when. Here functions didn't like that and I don't have motivation, energy, drive or whatever to bash my head against it till something eventually works.

5 Conclusions

So this is not even like the embedded systems laboratory works, where in conclusion I just said "It works". Here nothing works and I do not have in me whatever is needed to make it work. Working on images of neural networks with neural networks destroys too many neural networks by way of stress and at times anger so, for final time, whatever.

Here is the thing neural networks are a difficult construct, especially so when trying to find a region within an image. Even more so when said image is grayscale and asymmetric and the region to look for may be there, may not be, may be nigh a circle, may be a small noodle on the side. What I am getting at is that in my situation neural networks are way too difficult a concept for me to implement in a way that would yield some conclusive results. I admit being at fault here since I left myself two days here instead of working intermittently over all semester, but eitherway from what i understand i didn't have enough data to train proper image recognition neural network. And utilizing some great MATLAB toolboxes, like SPM12, gives better results with less hassle; also allows 3-dimensional work instead of 2D (referring to J.C. Griffis idea [2](Sec).

6 Source Codes

```
function [expFeat, goals] = NNCP_ET_NN()
% NNCP_ET_NN - NN training  geatures and goals extraction function
%
% ET 2017.12.17 "edgetech"  All Rights Reserved.

global extractedFeatures
start = tic;

if isempty(extractedFeatures)
    inputFileDir = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\';
    inputFileExt = '.nii';
    outputDir = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\NNCP_ET_NN\1␣-␣Copy␣(';

    sampleNum = 1;
    inputFileName = 'darkLeftHemKaput';
    features = NNCP_ImageCycling([inputFileDir inputFileName inputFileExt],[
       outputDir ns(sampleNum) ')\']);

    sampleNum = 2;
    inputFileName = 'frontEffed';
    features = [features NNCP_ImageCycling([inputFileDir inputFileName inputFileExt
       ],[outputDir ns(sampleNum) ')\'])];

    sampleNum = 3;
    inputFileName = 'leftHemKaput';
    features = [features NNCP_ImageCycling([inputFileDir inputFileName inputFileExt
       ],[outputDir ns(sampleNum) ')\'])];

    sampleNum = 4;
    inputFileName = 'mainTest';
    features = [features NNCP_ImageCycling([inputFileDir inputFileName inputFileExt
       ],[outputDir ns(sampleNum) ')\'])];

    sampleNum = 5;
    inputFileName = 'pretendFine';
    features = [features NNCP_ImageCycling([inputFileDir inputFileName inputFileExt
       ],[outputDir ns(sampleNum) ')\'])];

    sampleNum = 6;
    inputFileName = 'pretendFine2';
    features = [features NNCP_ImageCycling([inputFileDir inputFileName inputFileExt
       ],[outputDir ns(sampleNum) ')\'])];

    extractedFeatures = features;
else
```

```
    features = extractedFeatures;
end

[rows, columns] = size(features);
for i = 1:columns
    for k = 1:rows
        structure = features(k,i);
        expStruct = [
            double(structure.Area) ...
            double(structure.Centroid) ...
            double(structure.MajorAxisLength) ...
            double(structure.MinorAxisLength) ...
            double(structure.Eccentricity) ...
            double(structure.ConvexArea) ...
            double(structure.EquivDiameter) ...
            double(structure.Extrema(:,1)') ...
            double(structure.Extrema(:,2)') ...
            double(structure.Solidity) ...
            double(structure.Extent) ...
            double(structure.WeightedCentroid) ...
            double(structure.MeanIntensity) ...
            double(structure.MinIntensity) ...
            double(structure.MaxIntensity)
            ];
        if exist('exp3Struct','var'); exp3Struct = [exp3Struct expStruct];
        else exp3Struct = expStruct;
        end
    end
    exp3Struct = exp3Struct';
    if exist('expFeat','var'); expFeat = [expFeat exp3Struct];
    else expFeat = exp3Struct;
    end
    clear exp3Struct
end

% normalize them all
[rows,~] = size(expFeat);
for i = 1:rows
    expFeat(i,:) = expFeat(i,:)./max(expFeat(i,:));
end

% hand picked lessioned slices a.k.a. expected results
% sample 1
ss  = 122;
lss = 137;
les = 183;
es  = 186;
```

```
s1 = cat(ss,lss,les,es);
% sample 2
ss  = 119;
lss = 126;
les = 181;
es  = 181;
s2 = cat(ss,lss,les,es);
% sample 3
ss  = 122;
lss = 122;
les = 186;
es  = 186;
s3 = cat(ss,lss,les,es);
% sample 4
ss  = 86;
lss = 86;
les = 107;
es  = 131;
s4 = cat(ss,lss,les,es);
% sample 5
ss  = 119;
lss = 119;
les = 119;
es  = 181;
s5 = cat(ss,lss,les,es);
% sample 6
ss  = 122;
lss = 122;
les = 122;
es  = 186;
s6 = cat(ss,lss,les,es);

goals = [s1 s2 s3 s4 s5 s6];

fprintf(1,'NNCP_ET_NN␣time:␣%3.2f\n',toc(start));

end

function rez = cat(ss,lss,les,es)
rez = [0 zeros(1,lss-ss) ones(1,les-lss) zeros(1,es-les)];
end
```

```
function rez = nifti2slices(path,output,slices)
% NIFTI2SLICES - extract images from .nii file
%
%   path    - full or relative path to the .nii file
%   output  - full path to output directory including filename (none or
%                  empty str disables output)(no file extension plis)
%   slices  - number or vector of slice numbers to output (none outputs one
%                slice at 75% height)
%
% ET 2017.12.16 "edgetech"  All Rights Reserved.

% dbg:
if nargin == 0
    disp('[nifti2Slices]:␣using␣debug␣values');
    path = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\pretendFine.nii';
    output = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\slices\slc';
    slices = [0 1];
end;

outputFlag = 0;
if exist('output','var')
    if ~isempty(output)
        outputFlag = 1;
    end
end

global globalNii globalNiiName
if ~strcmp(globalNiiName, path)
    disp('[nifti2Slices]:␣Loading␣.nii')
    globalNii = load_nii(path);
    globalNiiName = path;
end
nii = globalNii;
[~,~,maxSlice] = size(nii.img);
rez = [];

if exist('slices','var')
    if length(slices)>1
        if slices(1) >= 0 && slices(1) < 1
            disp('[nifti2Slices]:␣Normalized␣slice␣range␣provided.')
            slices(1) = floor(slices(1)*maxSlice);
            if slices(1) == 0; slices(1) = 1; end
            slices(2) = floor(slices(2)*maxSlice);
        else
            disp('[nifti2Slices]:␣Numbered␣slice␣range␣provided.')
        end
        for i = slices(1):slices(2)
            if i>maxSlice; disp(['.nii␣has␣only␣' ns(maxSlice) '␣slices']);break; end;
            rez(:,:,i+1-slices(1)) = nii.img(:,:,i);
            if outputFlag; imwrite(im2uint16(double(nii.img(:,:,i))/mmx(double(nii.img(:,:,i)))),[output '_' ns(i) '.png']); end;
        end
    else
        disp('[nifti2Slices]:␣Single␣slice␣provided,␣getting␣single␣slice.')
        if slices>maxSlice
            disp('.nii␣has␣only␣'+ns(maxSlice)+'␣slices');
        else
            rez = nii.img(:,:,slices);
            if outputFlag; imwrite(im2uint16(double(rez)/mmx(double(rez))),[output '.png']); end;
        end
    end
else
    disp('[nifti2Slices]:␣Slices␣not␣provided,␣getting␣single␣slice␣at␣75%.')
    [~,~,z] = size(nii.img);
    rez = nii.img(:,:,round(z/4*3));
    if outputFlag; imwrite(im2uint16(double(rez)/mmx(double(rez))),[output '.png']); end;
end

if nargin == 0; rez = maxSlice; end
disp('[nifti2Slices]:␣Done')
end
```

```
function rez = NNCP_ImageCycling(inputFilepath,outputDir,outputFName,slices)
% NNCP_IMAGECYCLING - generates features and stuff
%
%   inputFilepath - path, filename, extension of input .nii file
%   outputDir     - slice image output directory
%   outputFName   - common name for the slices
%
%   Note: if any variables not given, defaults are used
%
% ET 2017.12.17 "edgetech"  All Rights Reserved.

start = tic;

if ~exist('inputFilepath','var'); inputFilepath = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\src.nii'; end
if ~exist('outputDir','var'); outputDir = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\slices\'; end
if ~exist('outputFName','var'); outputFName = 'slice'; end
if ~exist('slices','var'); slices = [.48 .73]; end
nifti2slices(inputFilepath,[outputDir outputFName],slices);
content = dir(outputDir);
for i = 1:length(content)
   if content(i).isdir; continue; end; %skip non files
   ret = NNCP_Image_Segmentation_edgetech([outputDir content(i).name]);
   if exist('rez','var'); rez = [rez ret];
   else rez = ret;
```

```
    end
end

disp(['NNCP_ImageCycling␣took:␣' ns(toc(start))]);

end
```

```
function rez = NNCP_Image_Segmentation_edgetech(filename)
% NNCP_IMAGE_SEGMENTATION_EDGETECH - generates features and stuff for
%   single slice
%
%   filename - full path name and extension to MRI image (.png/.jpg)
%
% ET 2017.12.17 "edgetech"  All Rights Reserved.

draw = 0;
if exist('filename','var'); fullFileName = filename;
else fullFileName = 'D:\Games\MATLAB␣R2015a\_et␣files\NNCP\slices\slice_75.png';
    figure; draw = 1;
end

global originalImage
originalImage = double(imread(fullFileName));
originalImage = uint8(originalImage*(255/mmmx(originalImage)));

start = tic; % Start timer.
rez = [process(1,.4,draw) process(2,.3,draw) process(3,.6,draw)]';
elapsedTime = toc(start);
fprintf(1,'NNCP_Image_Segmentation_edgetech␣time:␣%1.2f␣[%s]\n',elapsedTime,
    fullFileName);
if draw; fprintf(1,'\n'); end

end

function rez = process(row, threshold, draw)

global originalImage
row = (row-1)*3;
captionFontSize = 14;

% Check to make sure that it is grayscale, just in case the user substituted their
    own image.
[~, ~, numberOfColorChannels] = size(originalImage);
if numberOfColorChannels > 1
  originalImage = rgb2gray(originalImage);
end

if draw
```

```
    % Display the image.
    subplot(3, 3, 1+row);
    imshow(originalImage);
    % Force it to display RIGHT NOW (otherwise it might not display until it's all done, unless you've stopped at a breakpoint.)
    drawnow;
    caption = sprintf('Original␣image␣with␣boundary␣centers␣shown');
    if row == 0; title(caption, 'FontSize', captionFontSize); end;
    axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.

    % Just for fun, let's get its histogram and display it.
    [pixelCount, grayLevels] = imhist(originalImage);
    subplot(3, 3, 2+row);
    bar(pixelCount);
    if row == 0; title('Histogram␣of␣image', 'FontSize', captionFontSize); end;
    xlim([0 grayLevels(end)]); % Scale x axis manually.
    grid on;
end

% Threshold the image to get a binary image (only 0's and 1's) of class "logical."
% Method #1: using im2bw()
  normalizedThresholdValue = threshold; % In range 0 to 1.
  thresholdValue = normalizedThresholdValue * max(max(originalImage)); % Gray Levels.
  binaryImage = im2bw(originalImage, normalizedThresholdValue);       % One way to threshold to binary
% Method #2: using a logical operation.
%   thresholdValue = 100;
%   binaryImage = originalImage > thresholdValue; % Bright objects will be chosen if you use >.
% ========== IMPORTANT OPTION ============================================================
% Use < if you want to find dark objects instead of bright objects.
%   binaryImage = originalImage < thresholdValue; % Dark objects will be chosen if you use <.

% Do a "hole fill" to get rid of any background pixels or "holes" inside the blobs.
binaryImage = imfill(binaryImage, 'holes');

if draw
    % Show the threshold as a vertical red bar on the histogram.
    hold on;
    maxYValue = ylim;
    line([thresholdValue, thresholdValue], maxYValue, 'Color', 'r');
    % Place a text label on the bar chart showing the threshold.
    annotationText = sprintf('Thresholded␣at␣%d␣gray␣levels', thresholdValue);
```

```matlab
    % For text(), the x and y need to be of the data class "double" so let's cast both to double.
    text(double(thresholdValue + 5), double(0.5 * maxYValue(2)), annotationText, 'FontSize', 10, 'Color', [0 .5 0]);
    text(double(thresholdValue - 70), double(0.94 * maxYValue(2)), 'Background', 'FontSize', 10, 'Color', [0 0 .5]);
    text(double(thresholdValue + 50), double(0.94 * maxYValue(2)), 'Foreground', 'FontSize', 10, 'Color', [0 0 .5]);
end

% % Identify individual blobs by seeing which pixels are connected to each other.
% % Each group of connected pixels will be given a label, a number, to identify it and distinguish it from the other blobs.
% % Do connected components labeling with either bwlabel() or bwconncomp().
labeledImage = bwlabel(binaryImage, 8);     % Label each blob so we can make measurements of it

% Get all the blob properties.  Can only pass in originalImage in version R2008a and later.
blobMeasurements = regionprops(labeledImage, originalImage, 'all');
numberOfBlobs = size(blobMeasurements, 1);

if draw
    % bwboundaries() returns a cell array, where each cell contains the row/column coordinates for an object in the image.
    % Plot the borders of all the coins on the original grayscale image using the coordinates returned by bwboundaries.
    subplot(3, 3, 3+row);
    imshow(originalImage);
    if row == 0; title('Outlines,␣from␣bwboundaries()', 'FontSize', captionFontSize); end;
    axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
    hold on;
    boundaries = bwboundaries(binaryImage);
    numberOfBoundaries = size(boundaries, 1);
    for k = 1 : numberOfBoundaries
        thisBoundary = boundaries{k};
        plot(thisBoundary(:,2), thisBoundary(:,1), 'g', 'LineWidth', 2);
    end
    hold off;
end

mArea = 0;
mAreaIdx = 0;
textFontSize = 14;  % Used to control size of "blob number" labels put atop the image.
```

```
labelShiftX = -7; % Used to align the labels in the centers of the coins.
blobECD = zeros(1, numberOfBlobs);
% Print header line in the command window.
if draw
    if row == 0; fprintf(1,'Blob␣#␣␣␣␣␣␣Mean␣Intensity␣␣Area␣␣␣Perimeter␣␣␣␣Centroid␣␣␣␣␣␣␣Diameter\n');
    else fprintf(1,'------------------------------------------------------------------------\n');
    end
end
% Loop over all blobs printing their measurements to the command window.
for k = 1 : numberOfBlobs % Loop through all blobs.
    meanGL2008a = blobMeasurements(k).MeanIntensity;    % Find the mean of each blob.
  blobArea = blobMeasurements(k).Area;                % Get area.
    blobCentroid = blobMeasurements(k).Centroid;    % Get centroid one at a time
    if draw
        if blobArea > 50                                    % Don't output insignificant data
            blobPerimeter = blobMeasurements(k).Perimeter;    % Get perimeter.
            blobECD(k) = sqrt(4 * blobArea / pi);       % Compute ECD - Equivalent Circular Diameter.
            fprintf(1,'#%2d␣%17.1f␣%11.1f␣%8.1f␣%8.1f␣%8.1f␣%␣8.1f\n', k, meanGL2008a, blobArea, blobPerimeter, blobCentroid, blobECD(k));
        end
        % Put the "blob number" labels on the "boundaries" grayscale image.
        text(blobCentroid(1) + labelShiftX, blobCentroid(2), num2str(k), 'FontSize', textFontSize, 'FontWeight', 'Bold');
    end

    % find largest blob
    if blobArea > mArea
        mArea = blobArea;
        mAreaIdx = k;
    end
end

% Now, I'll show you another way to get centroids.
% We can get the centroids of ALL the blobs into 2 arrays,
% one for the centroid x values and one for the centroid y values.
allBlobCentroids = [blobMeasurements.Centroid];
centroidsX = allBlobCentroids(1:2:end-1);
centroidsY = allBlobCentroids(2:2:end);

if draw
    % % Plot the centroids in the original image in the upper left.
    % % Dimes will have a red cross, nickels will have a blue X.
    subplot(3, 3, 1+row);
    hold on; % Don't blow away image.
    for k = 1 : numberOfBlobs           % Loop through all keeper blobs.
        plot(centroidsX(k), centroidsY(k), 'r+', 'MarkerSize', 10, 'LineWidth', 2);
    end
end

rez = blobMeasurements(mAreaIdx);

end
```

```
function rez = et_dsc()

origdir = cd;

srcNiiDir = 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\mri_filtered\source'; % folder with FMRI scan .nii files (needs L or R at the name end to indicate damaged hamisphere)
labelsDir = 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\mri_filtered\labels'; % folder with hand drawn lesion areas for patients (name: ['l' patient_nii_file_name] <- that is small L)
wrkDirRoot= 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\etOutput\fullRun2\'; % folder where folders with generated data will be placed
pngDir =    'D:\Games\MATLAB␣R2015a\_et␣files\Masters\DSCplots\fullETrun2\et_dscrun\'; % folder for DSC plot placement

fwhm = 11;
srcNii = dir(srcNiiDir);
sub = 0;
for i =1:length(srcNii)
    [~, fname, fext] = fileparts(srcNii(i).name);
    if srcNii(i).isdir; disp('Not␣.nii'); sub = sub + 1; continue; end
    disp(['---␣Cycle␣#' num2str(i) '␣[' fname ']␣---']);

    workDir = [wrkDirRoot fname];
    cd(workDir)

%     if ~exist(['ws' fwhm 'lesion_labels_clustered_500.nii'],'file')
%         matlabbatch=[];
%         matlabbatch{1}.spm.util.defs.comp{1}.def = {['iy_' fname fext]}; % file path for inverse deformation field
%         matlabbatch{1}.spm.util.defs.out{1}.pull.fnames = {'s8lesion_labels_clustered_500.nii'}; % file path for reverse normalization
%         matlabbatch{1}.spm.util.defs.out{1}.pull.savedir.savepwd = 1; % save in current directory
%         matlabbatch{1}.spm.util.defs.out{1}.pull.interp = 7; %7th degree b-spline interpolation (values 4-7 can be used for 4th to 7th degree b-spline)
%           matlabbatch{1}.spm.util.defs.out{1}.pull.mask = 0; % no implicit masking
%           matlabbatch{1}.spm.util.defs.out{1}.pull.fwhm = [0 0 0]; % Gaussian smoothing kernel FWHM (can be applied to mask e.g. 8 8 8)
%           spm_jobman('run',matlabbatch); % run job
%       end

    lbl = load_nii(fullfile(labelsDir, ['l' fname fext]));
    l = lbl.img;
    wf1 = load_nii(['ws11' fname '_labels_clustered.nii']);
    f = wf1.img;

    cutoff = max(max(max(f)))*.325;
    f(f>=cutoff) = 1;
    f(f<cutoff) = 0;
    f = uint8(f);

%       sliceCnt = size(f,3);
%       ratio = reshape((2*summ(f&l))./(summ(f)+summ(l)), 1, sliceCnt);
%       areal = reshape(summ(l), 1, sliceCnt); %note: might find better whay than reshaping
%       areaf = reshape(summ(f), 1, sliceCnt);
%       mx = max([areal areaf])+100;
%       areal = areal./mx;
%       areaf = areaf./mx;
%
%       plot(1:sliceCnt, ratio, 'LineStyle', '-', 'Color', [0 .4 .75], 'LineWidth', 2); hold on
%       plot(1:sliceCnt, areal, 'LineStyle', '--', 'Color', [1 0 0]);
%       plot(1:sliceCnt, areaf, 'LineStyle', '--', 'Color', [0 .5 0]); hold off
%       xlabel('Slice number'); ylabel('DSC');
%       legend('DSC','Manual','Automatic')
%       title([strrep(fname(1:end-2),'_','') '; total DSC = ' num2str((2*summm(f&l))./(summm(f)+summm(l)))]);
%       ylim([0 1]);
%       grid on
%
%       printPlot(fname,pngDir,0,0);

%       break;
%       if i > 10; break; end

%       rez(i-sub)=(2*summm(f&l))./(summm(f)+summm(l));
    rez(i-sub)=(summm(f)*100)/(size(f,1)*size(f,2)*size(f,3));
end

cd(origdir)
disp('␣␣Done.')
```

```matlab

function y = summ(x)
y = sum(sum(x));

function y = summm(x)
y = sum(sum(sum(x)));
```

```matlab
function ratios = et_lesion()
% ET_LESION - automatic lesion detector
%
%   TBA
%
% DN 2016.11.15 "edgetech" All Rights Reserved.

origdir = cd;

srcNiiDir = 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\1Semester\testDir\src'; % folder with FMRI scan .nii files (needs L or R at the name end to indicate damaged hamisphere); without '\' at the end; was 'D:\Games\MATLAB R2015a\_et files\Masters\mri_filtered\source'
labelsDir = 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\1Semester\testDir\lbl'; % folder with hand drawn lesion areas for patients (name: ['l' patient_nii_file_name] <- that is small L); without '\' at the end; was 'D:\Games\MATLAB R2015a\_et files\Masters\mri_filtered\labels'
wrkDirRoot= 'D:\Games\MATLAB␣R2015a\_et␣files\Masters\1Semester\testDir\'; % folder where folders with generated data will be placed; with '\' at the end
pngDir =    'D:\Games\MATLAB␣R2015a\_et␣files\Masters\1Semester\testDir\png'; % folder for DSC plot placement; without '\' at the end
lesionDir = fullfile(fileparts(which('spm')), 'toolbox', 'lesion_gnb');


spm_jobman('initcfg');
fwhm = 11;
im_mask = 0;
times = [];
segTimes = [];
srcNii = dir([srcNiiDir '\*.nii']); % poll for .nii files
fullTime = tic;
ratios = {};


for i = 1:length(srcNii)
    [~, fname, fext] = fileparts(srcNii(i).name);
    disp(['---␣Cycle␣#' num2str(i) '␣[' fname ']␣---']);

    %     fname = 'P0089_R'; % debug

    workDir = [wrkDirRoot fname];
    if ~isdir(workDir); mkdir(workDir); end
    copyfile(fullfile(srcNiiDir,[fname fext]), workDir);
    cd(workDir)

    full = tic;

    % ---------------- segmentation -----------------------------------
    if ~exist(['wc1' fname fext],'file')
        seg = tic;
        run_sectioning(fullfile(workDir,[fname fext]))
        %          run_sectioningFull(fullfile(workDir,[fname fext]))
        t = toc(seg);
        segTimes = [segTimes t];
        disp(['Sectioning␣time:␣' num2str(t) '␣seconds.']);
    end

    % ---------------- feature extraction ------------------------------
    if ~exist('gmta','var') || ~exist('wmta','var') || ~exist('csfta','var')
        cd(fullfile(lesionDir,'volume_files'));
        if ~exist(['s' num2str(fwhm) 'TPM.nii'], 'file')
            my_smooth_spm('TPM.nii', [fwhm fwhm fwhm], im_mask); % if it doesn't
                exist, create it
        end
        smoothed_tpm_template = load_nii(['s' num2str(fwhm) 'TPM.nii']); % load
            smoothed TPM volume

        gmta = smoothed_tpm_template.img(:,:,:,1);
        wmta = smoothed_tpm_template.img(:,:,:,2);
        csfta = smoothed_tpm_template.img(:,:,:,3);
        clear smoothed_tpm_template

        % go to subject directory
        cd(workDir);
    end

    %ET: wont be used in the end, i guess, but now for reruns it is useful:
    if ~exist(['s' num2str(fwhm) 'wc1' fname fext], 'file')
        my_smooth_spm({['wc1' fname '.nii']; ['wc2' fname '.nii']; ['wc3' fname '.
            nii']}, [fwhm fwhm fwhm], im_mask); %ET: creates blurred wc1-3_Scans
    end
    %ET: kruva load'inimo ir isskirstymo kintamaisiais
    gmf = load_nii(['s' num2str(fwhm) 'wc1' fname fext]);
    wmf = load_nii(['s' num2str(fwhm) 'wc2' fname fext]);
    csff = load_nii(['s' num2str(fwhm) 'wc3' fname fext]);

    gma = gmf.img;
    wma = wmf.img;
    csfa = csff.img;

    % get brain mask
    if ~exist('outside_index','var')
        b_mask = load_nii(fullfile(lesionDir, 'volume_files', 'mask_ICV.nii'));
        b_mask = b_mask.img;
        outside_index = find(b_mask == 0); %ET: netestavau, bet panasu isrenka
            indexus tasku kurie yra uz kaukoles template
    end

    % smooth and clip segmentations
    gma(outside_index) = 0; %ET: ir pagal ta outside_index
    wma(outside_index) = 0; % ... isvalo duomenis ...
    csfa(outside_index) = 0; % ... kaukoles isoreje.

    gmu = flip(gma,1);
    wmu = flip(wma,1);
    csfu = flip(csfa, 1);

    l_bound = ceil(0.5*size(gma,1));
    affected = fname(end);
    if strcmp(affected, 'L') == 1
        idxFrom = l_bound;
        idxTo = size(gma,1);
    elseif strcmp(affected, 'R') == 1
        idxFrom = 1;
        idxTo = l_bound;
    else
        error('What␣Hemisphere?');
    end

    % create single hemisphere subject volumes %ETedit
    gma(idxFrom:idxTo, :,:) = 0;
    wma(idxFrom:idxTo, :,:) = 0;
    csfa(idxFrom:idxTo, :,:) = 0;
    gmu(idxFrom:idxTo, :,:) = 0;
    wmu(idxFrom:idxTo, :,:) = 0;
    csfu(idxFrom:idxTo, :,:) = 0;
    gmt = gmta; gmt(idxFrom:idxTo, :,:) = 0;
    wmt = wmta; wmt(idxFrom:idxTo, :,:) = 0;
    csft = csfta; csft(idxFrom:idxTo, :,:) = 0;


    ct = (csfa - csft).*((gmt + wmt)-(gma + wma)); % missing tissue feature map 1
    cm = (csfa - csfu).*((gmu + wmu)-(gma + wma)); % missing tissue feature map 2

    f1 = (cm + ct)./2; % get average of the two volumes
    f1 = round(f1.*100)./100; % round result to two significant digits
    f1(f1<0)= 0; % set voxels with values < 0 to 0

    gm1 = (gma-gmu).*(wmu-wma); % abnormal tissue feature map 1
    gt1 = (gma-gmt).*(wmt-wma); % abnormal tissue feature map 2

    f2 = (gm1 + gt1)./2; % get average volume
    f2 = round(f2.*100)./100; % round result to 2 significant digits
    f2(f2<0) = 0; % set voxels with values < 0 to 0

    gmf.img = f1; %ET: he asigns working images to my_gm so that save_nii could work. ...
    save_nii(gmf, [fname '_f1.nii']); % ... It aparently needs some of those extra variables/parameters.
    gmf.img = f2;
    save_nii(gmf, [fname '_f2.nii']);

    tb_mask = b_mask;
    tb_mask(idxFrom:idxTo,:,:) = 0;
    indexes = find(tb_mask); % extract indices of non-zero voxels from mask
    features = [f1(indexes) f2(indexes)];

    %      save my_features features %ET: used in classification
    %      save my_ind indexes
    %      load my_features features


    % ---------------- clasification ------------------------------------
    if ~exist('predict_lesion','var')
        load(fullfile(lesionDir, 'trained_gnbc', 'predict_lesion.mat')) %ET: some sort of his prediction; 1x1 struct ClassificationNaiveBayes
        x = .5; % default priors he suggests when inputing values
        priors = [1-x x];
        predict_lesion.Prior = priors; %[0.8936    0.1064] original priors
    end

    % predictExample() %ET: my try to figure out what predict needs, gives and does
    [labels, posterior] = predict(predict_lesion, features); %ET: where the magic happens
    gmf.img(:,:,:) = 0;
    gmf.img(indexes) = posterior(:,2);
    save_nii(gmf, [fname '_posterior.nii']);

    gmf.img(:,:,:) = 0;
    gmf.img(indexes) = labels;
    save_nii(gmf, [fname '_labels.nii']);

    % ---------------- post-processing ---------------------------------
    my_smooth_spm(fullfile(pwd, [fname '_labels.nii']), [fwhm fwhm fwhm], 0);
    slnii = load_nii(['s' num2str(fwhm) fname '_labels.nii']);
    slnii.nii(slnii.img <= 0.25) = 0;
    slnii.nii(slnii.img > 0) = 1;

    pproc_cluster = 200;
    [~, slnii.nii] = distance_cluster(slnii.img, 26); %ET: clustering algorithm
    slnii.nii(slnii.img < pproc_cluster) = 0;
    slnii.nii(slnii.img > 0) = 1;
    save_nii(slnii, [slnii.fileprefix '_clustered.nii']);

    asx = toc(full);
    times = [times asx];

    % ---------------- DSC ------------------------------------------
    matlabbatch=[];
    matlabbatch{1}.spm.util.defs.comp{1}.def = {['iy_' fname fext]}; % file path for inverse deformation field
    matlabbatch{1}.spm.util.defs.out{1}.pull.fnames = {['s' num2str(fwhm) fname '_labels_clustered.nii']}; % file path for reverse normalization
    matlabbatch{1}.spm.util.defs.out{1}.pull.savedir.savepwd = 1; % save in current directory
    matlabbatch{1}.spm.util.defs.out{1}.pull.interp = 7; %7th degree b-spline interpolation (values 4-7 can be used for 4th to 7th degree b-spline)
    matlabbatch{1}.spm.util.defs.out{1}.pull.mask = 0; % no implicit masking
    matlabbatch{1}.spm.util.defs.out{1}.pull.fwhm = [0 0 0]; % Gaussian smoothing kernel FWHM (can be applied to mask e.g. 8 8 8)
    spm_jobman('run',matlabbatch); % run job

    lbl = load_nii(fullfile(labelsDir, ['l' fname fext]));
    l = lbl.img;
    wf1 = load_nii(['ws' num2str(fwhm) fname '_labels_clustered.nii']);
    f = wf1.img;

    cutoff = max(max(max(f)))*.325;
    f(f>=cutoff) = 1;
    f(f<cutoff) = 0;
    f = uint8(f);

    sliceCnt = size(f,3);
    ratio = reshape((2*summ(f&l))./(summ(f)+summ(l)), 1, sliceCnt);
    areal = reshape(sum(sum(l)), 1, sliceCnt); %note: might find better whay than reshaping
    areaf = reshape(sum(sum(f)), 1, sliceCnt);
    mx = max([areal areaf])+100;
    areal = areal./mx;
    areaf = areaf./mx;

    plot(1:sliceCnt, ratio, 'LineWidth', 2); hold on
    plot(1:sliceCnt, areal, 'r--', 1:sliceCnt, areaf, 'g--'); hold off
    xlabel('Slice␣number');
    legend('DSC','Manual','Automatic')
    title(fname(1:end-2));
    %     title([fname '; average DSC = ' num2str(sum(ratio))])
    ylim([0 1]);
    grid on

    ratios{i} = ratio;

    printPlot(fname,pngDir,0,0);

    %     if i > 6; break; end
    %     break;
end

fullTime = toc(fullTime)
cd('D:\Games\MATLAB␣R2015a\_et␣files\Masters');
save etrun.mat times segTimes

cd(origdir)
disp('␣␣Done.')



function run_sectioning(filepath)
utilDir = fileparts(which('spm'));

matlabbatch=[];
matlabbatch{1}.spm.spatial.preproc.channel.vols = {filepath};
matlabbatch{1}.spm.spatial.preproc.channel.biasreg = 0.001;
matlabbatch{1}.spm.spatial.preproc.channel.biasfwhm = 60;
matlabbatch{1}.spm.spatial.preproc.channel.write = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(1).tpm = {[utilDir, '/tpm/TPM.nii,1']};
matlabbatch{1}.spm.spatial.preproc.tissue(1).ngaus = 1;
matlabbatch{1}.spm.spatial.preproc.tissue(1).native = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(1).warped = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(2).tpm = {[utilDir, '/tpm/TPM.nii,2']};
matlabbatch{1}.spm.spatial.preproc.tissue(2).ngaus = 1;
matlabbatch{1}.spm.spatial.preproc.tissue(2).native = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(2).warped = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(3).tpm = {[utilDir, '/tpm/TPM.nii,3']};
matlabbatch{1}.spm.spatial.preproc.tissue(3).ngaus = 2;
matlabbatch{1}.spm.spatial.preproc.tissue(3).native = [0 0];
```

```
matlabbatch{1}.spm.spatial.preproc.tissue(3).warped = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(4).tpm = {[utilDir, '/tpm/TPM.nii,4']};
matlabbatch{1}.spm.spatial.preproc.tissue(4).ngaus = 3;
matlabbatch{1}.spm.spatial.preproc.tissue(4).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(4).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(5).tpm = {[utilDir, '/tpm/TPM.nii,5']};
matlabbatch{1}.spm.spatial.preproc.tissue(5).ngaus = 4;
matlabbatch{1}.spm.spatial.preproc.tissue(5).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(5).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(6).tpm = {[utilDir, '/tpm/TPM.nii,6']};
matlabbatch{1}.spm.spatial.preproc.tissue(6).ngaus = 2;
matlabbatch{1}.spm.spatial.preproc.tissue(6).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(6).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.warp.mrf = 1;
matlabbatch{1}.spm.spatial.preproc.warp.cleanup = 1;
matlabbatch{1}.spm.spatial.preproc.warp.reg = [0 0.001 0.5 0.05 0.2];
matlabbatch{1}.spm.spatial.preproc.warp.affreg = 'mni';
matlabbatch{1}.spm.spatial.preproc.warp.fwhm = 0;
matlabbatch{1}.spm.spatial.preproc.warp.samp = 3;
matlabbatch{1}.spm.spatial.preproc.warp.write = [1 0];
spm_jobman('run',matlabbatch);

function run_sectioningFull(filepath)
utilDir = fileparts(which('spm'));

matlabbatch=[];
matlabbatch{1}.spm.spatial.preproc.channel.vols = {filepath}; % specifies file
    insted of selection
matlabbatch{1}.spm.spatial.preproc.channel.biasreg = 0.001; % default
matlabbatch{1}.spm.spatial.preproc.channel.biasfwhm = 60; % default
matlabbatch{1}.spm.spatial.preproc.channel.write = [1 1]; % save bias corrected -
    saves field and corrected; default is [0 0] - save nothing
matlabbatch{1}.spm.spatial.preproc.tissue(1).tpm = {[utilDir, '/tpm/TPM.nii,1']}; %
     default; tissue probability map of gray matter
matlabbatch{1}.spm.spatial.preproc.tissue(1).ngaus = 1; % default
matlabbatch{1}.spm.spatial.preproc.tissue(1).native = [1 0]; % default; save native
    , dont save "Dartel Imported"
matlabbatch{1}.spm.spatial.preproc.tissue(1).warped = [1 1]; % save modulated and
    unmodulated warped tissue - later code needs unmodulated warped tissues 1-3;
    default [0 0] - no save
matlabbatch{1}.spm.spatial.preproc.tissue(2).tpm = {[utilDir, '/tpm/TPM.nii,2']}; %
     default; tissue prob. map of white matter
matlabbatch{1}.spm.spatial.preproc.tissue(2).ngaus = 1; % default
matlabbatch{1}.spm.spatial.preproc.tissue(2).native = [1 0]; % default; save native
    , dont save "Dartel Imported"
matlabbatch{1}.spm.spatial.preproc.tissue(2).warped = [1 1]; % save modulated and
    unmodulated warped tissue - later code needs unmodulated warped tissues 1-3;
```

```
default [0 0] - no save
matlabbatch{1}.spm.spatial.preproc.tissue(3).tpm = {[utilDir, '/tpm/TPM.nii,3']}; % default; tissue prob. map of CSF
matlabbatch{1}.spm.spatial.preproc.tissue(3).ngaus = 2; % default
matlabbatch{1}.spm.spatial.preproc.tissue(3).native = [1 0]; % default; save native, dont save "Dartel Imported"
matlabbatch{1}.spm.spatial.preproc.tissue(3).warped = [1 1]; % save modulated and unmodulated warped tissue - later code needs unmodulated warped tissues 1-3; default [0 0] - no save
matlabbatch{1}.spm.spatial.preproc.tissue(4).tpm = {[utilDir, '/tpm/TPM.nii,4']}; % default; tissue prob. map of bone
matlabbatch{1}.spm.spatial.preproc.tissue(4).ngaus = 3; % default
matlabbatch{1}.spm.spatial.preproc.tissue(4).native = [1 0]; % default; save native, dont save "Dartel Imported"
matlabbatch{1}.spm.spatial.preproc.tissue(4).warped = [1 1]; % save modulated and unmodulated warped tissue - later code needs unmodulated warped tissues 1-3 (not this one); default [0 0] - no save
matlabbatch{1}.spm.spatial.preproc.tissue(5).tpm = {[utilDir, '/tpm/TPM.nii,5']}; % default; -//- soft tissue
matlabbatch{1}.spm.spatial.preproc.tissue(5).ngaus = 4; % default
matlabbatch{1}.spm.spatial.preproc.tissue(5).native = [1 0]; % default; save native, dont save "Dartel Imported"
matlabbatch{1}.spm.spatial.preproc.tissue(5).warped = [0 0]; % default; dont save warped tissues
matlabbatch{1}.spm.spatial.preproc.tissue(6).tpm = {[utilDir, '/tpm/TPM.nii,6']}; % default; -//- air/background
matlabbatch{1}.spm.spatial.preproc.tissue(6).ngaus = 2; % default
matlabbatch{1}.spm.spatial.preproc.tissue(6).native = [1 0]; % save native, dont save "Dartel Imported"; deafult is [0 0] - no saving
matlabbatch{1}.spm.spatial.preproc.tissue(6).warped = [0 0]; % default; dont save warped tissues
matlabbatch{1}.spm.spatial.preproc.warp.mrf = 1; % default
matlabbatch{1}.spm.spatial.preproc.warp.cleanup = 1; % default; "light cleanup"
matlabbatch{1}.spm.spatial.preproc.warp.reg = [0 0.001 0.5 0.05 0.2]; % default
matlabbatch{1}.spm.spatial.preproc.warp.affreg = 'mni'; % default
matlabbatch{1}.spm.spatial.preproc.warp.fwhm = 0; % default
matlabbatch{1}.spm.spatial.preproc.warp.samp = 3; % default
matlabbatch{1}.spm.spatial.preproc.warp.write = [1 1]; % save forward and inverse deformation fields; default is [0 0] - dont save
spm_jobman('run',matlabbatch);

function y = summ(x)
y = sum(sum(x));


% When sectioning 47/48 and processing all:
% Execution times summary
% Minumum segmentation time: 83.054 seconds
% Average segmentation time: 130.3229 seconds
% Maximum segmentation time: 204.4439 seconds
% Full script runtime: 6705.0625 seconds



% [src, fname, ext] = fileparts(filepath);
% movefile([src '\wc1' fname ext]);
% movefile([src '\wc2' fname ext]);
% movefile([src '\wc3' fname ext]);
% movefile([src '\' fname '_seg8.mat']);

% function checkNii(template, saveImg, name)
% template.img = saveImg;
% save_nii(template, [name '.nii']);
% disp(['Saved .nii to: ' cd '\' name '.nii'])
```

References

[1] Image Analyst. Image segmentation tutorial. *MATLAB*, 2015.

[2] Joseph C Griffis, Jane B Allendorfer, and Jerzy P Szaflarski. Voxel-based Gaussian naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans. *Journal of neuroscience methods*, 257:97–108, 2016.

[3] Chandra Sekhar Ravuri. Automatic segmentation of brain tumor in mr images. *MATLAB*, 2015.

[4] Shazid Mahmood. Brain tumor detection from mri images using anisotropic filter and segmentation image processing. *MATLAB*, 2017.

[5] John Ashburner and Karl J Friston. Unified segmentation. *Neuroimage*, 26(3):839–851, 2005.

[6] Edward A Ashton, Chihiro Takahashi, Michel J Berg, Andrew Goodman, Saara Totterman, and Sven Ekholm. Accuracy and reproducibility of manual and semiautomated quantification of MS lesions by MRI. *Journal of Magnetic Resonance Imaging*, 17(3):300–308, 2003.

[7] Elizabeth Bates, Stephen M Wilson, Ayse Pinar Saygin, Frederic Dick, Martin I Sereno, Robert T Knight, and Nina F Dronkers. Voxel-based lesion–symptom mapping. *Nature neuroscience*, 6(5):448–450, 2003.

[8] Rong Chen and Edward H Herskovits. Voxel-based Bayesian lesion–symptom mapping. *Neuroimage*, 49(1):597–602, 2010.

[9] Jenny Crinion, John Ashburner, Alex Leff, Matthew Brett, Cathy Price, and Karl Friston. Spatial normalization of lesioned brains: performance evaluation and impact on fMRI analyses. *Neuroimage*, 37(3):866–875, 2007.

[10] Jenny Crinion, Audrey L Holland, David A Copland, Cynthia K Thompson, and Argye E Hillis. Neuroimaging in aphasia treatment research: quantifying brain lesions after stroke. *Neuroimage*, 73:208–214, 2013.

[11] Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[12] Julie A Fiez, Hanna Damasio, and Thomas J Grabowski. Lesion segmentation and manual warping to a reference brain: Intra-and interobserver reliability. *Human brain mapping*, 9(4):192–211, 2000.

[13] Sharon Geva, Jean-Claude Baron, P Simon Jones, Cathy J Price, and Elizabeth A Warburton. A comparison of VLSM and VBM in a cohort of patients with post-stroke aphasia. *NeuroImage: Clinical*, 1(1):37–47, 2012.

[14] Sonya Mehta, Thomas J Grabowski, Yogi Trivedi, and Hanna Damasio. Evaluation of voxel-based morphometry for focal lesion detection in individuals. *Neuroimage*, 20(3):1438–1454, 2003.

[15] Jhimli Mitra, Pierrick Bourgeat, Jurgen Fripp, Soumya Ghose, Stephen Rose, Olivier Salvado, Alan Connelly, Bruce Campbell, Susan Palmer, Gagan Sharma, et al. Lesion segmentation from multimodal MRI using random forest following ischemic stroke. *NeuroImage*, 98:324–335, 2014.

[16] Rajeev DS Raizada and Yune-Sang Lee. Smoothness without smoothing: why Gaussian naive Bayes is not naive for multi-subject searchlight studies. *PloS one*, 8(7):e69566, 2013.

[17] Francisco Pereira, Tom Mitchell, and Matthew Botvinick. Machine learning classifiers and fMRI: a tutorial overview. *Neuroimage*, 45(1):S199–S209, 2009.

[18] P Ripolles, Josep Marco-Pallarés, Ruth de Diego-Balaguer, J Miro, M Falip, M Juncadella, F Rubio, and Antoni Rodriguez-Fornells. Analysis of automated methods for spatial normalization of lesioned brains. *Neuroimage*, 60(2):1296–1306, 2012.

[19] Irina Rish, Joseph Hellerstein, and Jayram Thathachar. An analysis of data characteristics that affect naive bayes performance. *IBM TJ Watson Research Center*, 30, 2001.

[20] Chris Rorden, Julius Fridriksson, and Hans-Otto Karnath. An evaluation of traditional and novel tools for lesion behavior mapping. *Neuroimage*, 44(4):1355–1362, 2009.

[21] Chris Rorden, Hans-Otto Karnath, and Leonardo Bonilha. Improving lesion-symptom mapping. *Journal of cognitive neuroscience*, 19(7):1081–1088, 2007.

[22] Mohamed L Seghier, Anil Ramlackhansingh, Jenny Crinion, Alexander P Leff, and Cathy J Price. Lesion identification using unified segmentation-normalisation models and fuzzy clustering. *Neuroimage*, 41(4):1253–1266, 2008.

[23] Marko Wilke, Bianca de Haan, Hendrik Juenger, and Hans-Otto Karnath. Manual, semi-automated, and automated delineation of chronic brain lesions: a comparison of methods. *Neuroimage*, 56(4):2038–2046, 2011.

[24] Alex P Zijdenbos and Benoit M Dawant. Brain segmentation and white matter lesion detection in MR images. *Critical reviews in biomedical engineering*, 22(5-6):401–465, 1993.

[25] Bruno Alfano, Arturo Brunetti, Michele Larobina, Mario Quarantelli, Enrico Tedeschi, Andrea Ciarmiello, Eugenio M Covelli, and Marco Salvatore. Automated segmentation and measurement of global white matter lesion volume in patients with multiple sclerosis. *Journal of Magnetic Resonance Imaging*, 12(6):799–807, 2000.

[26] Miguel Angel Gonzalez Ballester, Andrew Zisserman, and Michael Brady. Segmentation and measurement of brain structures in MRI including confidence bounds. *Medical Image Analysis*, 4(3):189–200, 2000.

[27] Koen Van Leemput, Frederik Maes, Dirk Vandermeulen, Alan Colchester, and Paul Suetens. Automated segmentation of multiple sclerosis lesions by model outlier detection. *IEEE transactions on medical imaging*, 20(8):677–688, 2001.

[28] RK-S Kwan, Alan C Evans, and G Bruce Pike. MRI simulation-based evaluation of image-processing and classification methods. *IEEE transactions on medical imaging*, 18(11):1085–1097, 1999.

[29] Remi K-S Kwan, Alan C Evans, and G Bruce Pike. An extensible MRI simulator for post-processing evaluation. In *Visualization in biomedical computing*, pages 135–140. Springer, 1996.

[30] Edward Ashton, Saara Totterman, Chihiro Takahashi, Jose Tamez-Pena, and Kevin J Parker. Automated measurement of structures on CT and MR imagery: A validation study. In *Proceedings of the Fourteenth IEEE Symposium on Computer-Based Medical Systems*, page 300. IEEE Computer Society, 2001.

[31] Edward A Ashton, Kevin J Parker, Michel J Berg, and Chang Wen Chen. A novel volumetric feature extraction technique with applications to MR images. *IEEE transactions on medical imaging*, 16(4):365–371, 1997.

[32] David C Taylor and William A Barrett. Image segmentation using globally optimal growth in three dimensions with an adaptive feature set. In *Visualization in Biomedical Computing 1994*, pages 98–107. International Society for Optics and Photonics, 1994.

[33] Dorian Pustina, H Coslett, Peter E Turkeltaub, Nicholas Tustison, Myrna F Schwartz, and Brian Avants. Automated segmentation of chronic stroke lesions using LINDA: lesion identification with neighborhood data analysis. *Human brain mapping*, 2016.

[34] Bianca de Haan, Philipp Clas, Hendrik Juenger, Marko Wilke, and Hans-Otto Karnath. Fast semi-automated lesion demarcation in stroke. *NeuroImage: Clinical*, 9:69–74, 2015.

[35] Xin-Wei Li, Qiong-Ling Li, Shu-Yu Li, and De-Yu Li. Local manifold learning for multiatlas segmentation: application to hippocampal segmentation in healthy population and alzheimer's disease. *CNS Neuroscience & Therapeutics*, 21(10):826–836, 2015.

[36] Rola Harmouche, Nagesh K Subbanna, D Louis Collins, Douglas L Arnold, and Tal Arbel. Probabilistic multiple sclerosis lesion classification based on modeling regional intensity variability and local neighborhood information. *IEEE Transactions on Biomedical Engineering*, 62(5):1281–1292, 2015.

[37] Matthew A Petoe, Winston D Byblow, Esther JM de Vries, Venkatesh Krishnamurthy, Cathy S Zhong, P Alan Barber, and Cathy M Stinear. A template-based procedure for determining white matter integrity in the internal capsule early after stroke. *NeuroImage: Clinical*, 4:695–700, 2014.

[38] Celine R Gillebert, Glyn W Humphreys, and Dante Mantini. Automated delineation of stroke lesions using brain CT images. *NeuroImage: Clinical*, 4:540–548, 2014.

[39] Colm Elliott, Douglas L Arnold, D Louis Collins, and Tal Arbel. Temporally consistent probabilistic detection of new multiple sclerosis lesions in brain mri. *IEEE Transactions On medical imaging*, 32(8):1490–1503, 2013.

[40] Xavier Lladó, Onur Ganiler, Arnau Oliver, Robert Martí, Jordi Freixenet, Laia Valls, Joan C Vilanova, Lluís Ramió-Torrentà, and Àlex Rovira. Automated detection of multiple sclerosis lesions in serial brain MRI. *Neuroradiology*, 54(8):787–807, 2012.

[41] Diane M Renz, Horst K Hahn, Peter Schmidt, Jan Rexilius, Markus Lentschig, Alexander Pfeil, Dieter Sauner, Clemens Fitzek, Hans-Joachim Mentzel, Werner A Kaiser, et al. Accuracy and reproducibility of a novel semi-automatic segmentation technique for MR volumetry of the pituitary gland. *Neuroradiology*, 53(4):233–244, 2011.

[42] Rong Chen, Argye E Hillis, Mikolaj Pawlak, and Edward H Herskovits. Voxelwise Bayesian lesion-deficit analysis. *NeuroImage*, 40(4):1633–1642, 2008.

[43] Oskar Maier, Matthias Wilms, Janina von der Gablentz, Ulrike M Krämer, Thomas F Münte, and Heinz Handels. Extra tree forests for sub-acute ischemic stroke lesion segmentation in MR sequences. *Journal of neuroscience methods*, 240:89–100, 2015.

[44] DL Collins, AC Evans, C Holmes, and TM Peters. Automatic 3D segmentation of neuro-anatomical structures from MRI. In *Information processing in medical imaging*, volume 3, pages 139–152. Kluwer Dordrecht, 1995.

[45] Joseph C. Griffis. lesion gnb toolbox for SPM12. Online, March 2016. Accessed November 2016.

[46] By members & collaborators of the Wellcome Trust Centre for Neuroimaging. Statistical Parametric Mapping - SPM. Online. Accessed November 2016 at: www.fil.ion.ucl.ac.uk/spm/software/.

[47] David MacDonald, Noor Kabani, David Avis, and Alan C Evans. Automated 3-D extraction of inner and outer surfaces of cerebral cortex from MRI. *NeuroImage*, 12(3):340–356, 2000.

[48] Alain Pitiot, Hervé Delingette, Paul M Thompson, and Nicholas Ayache. Expert knowledge-guided segmentation system for brain MRI. *NeuroImage*, 23:S85–S96, 2004.

[49] John Ashburner, Karl J Friston, et al. Nonlinear spatial normalization using basis functions. *Human brain mapping*, 7(4):254–266, 1999.

[50] David W Shattuck, Stephanie R Sandor-Leahy, Kirt A Schaper, David A Rottenberg, and Richard M Leahy. Magnetic resonance image tissue classification using a partial volume model. *NeuroImage*, 13(5):856–876, 2001.

[51] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.