<Improving the Random Walker algorithm for interactive 3D medical image segmentation using AI predictions>

<Modify the weight function to rely on an ensemble of segmentation predictions>

< **Bram Stellinga**[1]>

Supervisor(s): <Klaus Hildebrandt[1]**>, <Nicolas Chaves-de-Plaza**[1]**>**

[1]EEMCS, Delft University of Technology, The Netherlands

No Institute Given

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: <Bram Stellinga>
Final project course: CSE3000 Research Project
Thesis committee: <Klaus Hildebrandt>, <Nicolas Chaves-de-Plaza>, <Thomas Abeel>

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract. The segmentation of anatomical structures in 3D medical images is crucial for various applications in the field of medical imaging. Fully automated methods often lack accuracy, while manual segmenting requires much time and effort from a user. Due to this, Active Learning approaches are being proposed to solve this by making the method interactive. This is done by iteratively segmenting the image based on user provided input, which can then be added to correct uncertain regions of intermediate segmentation results. We propose a method to improve a Random Walker (RW) algorithm that is used for interactive 3D medical image segmentation, by integrating an ensemble of predicted segmentations obtained by a previously trained Bayesian Deep Neural Network (BDNN). The predictions are used to improve the weight function used by the RW to indicate the similarity of neighbouring voxels. We evaluate our results by combining the weight function originally used by the RW, with two custom approaches, Mean Predictions and Unanimous Votes. Both are combined with the original RW's weights in the form of a weighted sum. Mean Predictions is based on only the mean of all predictions for each voxel, while Unanimous Votes only considers averages of either exactly 1 or 0. Lastly, we will propose a method to combine the original weight function with the mean predictions by summing together the image intensities with the mean predictions while keeping the variance normalized. All results are evaluated using both synthetic data and empirical data from a MICCAI dataset. Lastly, our method shows that the Adaptive Alpha Approach outperforms all other methods including the original RW in terms of average DICE Coeffients for the first two iterations.

1 Introduction

In the field of medical imaging, the segmentation of anatomical structures in a 3D medical image plays a crucial role, e.g. by providing information about treatment planning by locating organs or identifying tumor boundaries in radiation therapy etc. Besides the various algorithms designed for 3D image segmentation, state-of-the-art algorithms often still produce errors that need to be checked by humans. Recent sources, such as [5], therefore propose methods that combine Active Learning with 3D image segmentation.

Active learning is a type of machine learning where samples from unlabeled data are selected based on the model's uncertainty and iteratively queried to a human to provide additional labels. More specifically, the method proposed in [5] starts with some user input in the form of contours drawn around the region of interest (ROI), for one or more 2D slices of the 3D input image. A segmentation algorithm then segments the image based on these annotations. The resulting segmentation is then evaluated by creating an uncertainty field indicating the uncertainty of the classification per each voxel in the image, depending on boundary energy, regional energy, smoothness and entropy energy. Subsequently, a 2D slice with the most uncertainty is queried to the user, who then aims to improve the previous segmentation result by drawing additional

contours on the queried slice. After the segmentation is executed again with the updated user input, the previous steps are repeated until the user is satisfied. See figure 1 for an example.

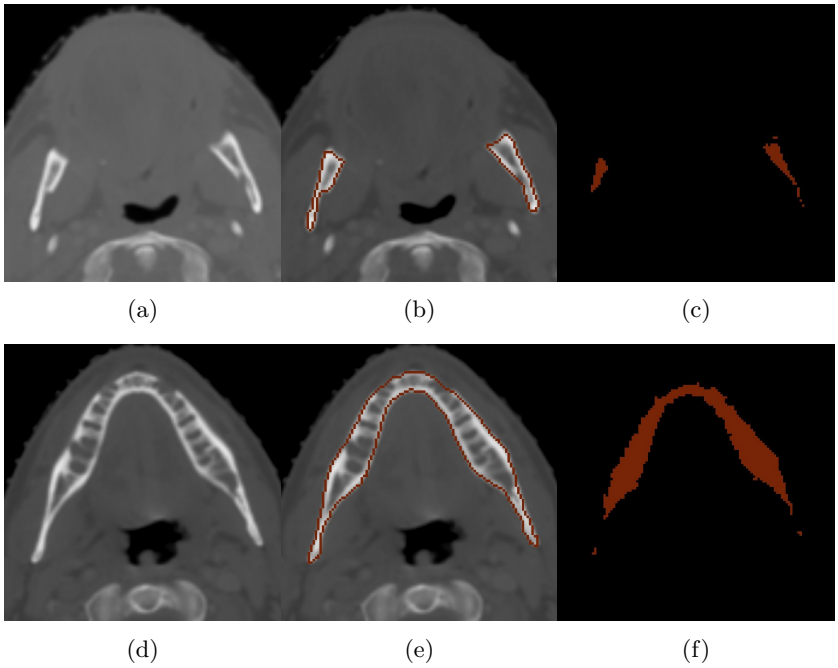


Fig. 1: An example of the general pipeline, the first row shows the first iteration and the second row the second. (a) The input image. (b) The contour(s) around the ROI added by the user. (c) The resulting segmentation. (d) The query slice. (e) The additional contours added by the user. (f) The new segmentation result.

In the previously mentioned paper, a Random Walker (RW) algorithm [2] is used for image segmentation. One of its main advantages is that besides the segmented image, it also provides the probabilities of each voxel belonging to a label, which can directly be interpreted as the certainty of the classification and thus be used to create the uncertainty field. The overall performance is mainly evaluated as the amount of human interaction needed for the uncertainty to drop below a preset threshold. Since the uncertainty values are directly based on the result of the RW, improving the segmentation algorithm will result in a decreasing amount of effort required from the user.

To both improve the accuracy of the segmentation algorithm and reduce the required user effort, we came up with a method to integrate predefined AI-based segmentation results with the Random Walker. For this, we used a network

that has been trained beforehand, on a subset of a MICCAI[1] dataset, and has been used to generate an ensemble of predictions for the remaining test split of this data. As the RW provides uncertainty values that are needed for finding the query slice, we decided to not replace the algorithm but try to make it benefit from these predictions. This is because the uncertainties that originate from the predictions are not necessarily based on the image itself, but rather on the standard deviation of the ensemble. The aim of this report is therefore to answer the following question: how can the performance of the Random Walker for interactive 3D image segmentation be improved by integrating an ensemble of AI-based segmentation predictions?

To answer this research question, first [5] will be replicated and the results will be evaluated similarly. Next, the pre-generated predictions will be integrated by modifying the weight function used in the RW and make this depend on these predictions. As this modification can be done in several ways, we will start by defining two different methods. One where the weight function is directly depending on the average of all predictions, and one where the average is scaled to minimize the influence of non-unanimous votes. Both methods will then be combined with the original weight function that is based on the image intensities. This combination will be in the form of a weighted sum, for which different weights will be evaluated. Finally, we will get to our main goal, which is to make the ratio of this weighted sum adaptive, by making it depend on the similarity of all predictions for each voxel.

The report will be presented in the following structure. First the methodology used in this research will be further described and explained in section 2. After this, the experimental setup is explained together with the results in 3. Next, a reflection of the responsible research will be discussed in section 4. Finally, the results are discussed in 5 followed by a conclusion and a discussion about possibilities for future work in 6.

2 Methodology

Before explaining the RW algorithm and how we will use a set of pre-generated segmentation predictions by a Deep Neural Network (DNN) to improve it, we will define some notations. The spatial image domain will be represented by $\Omega \subset \mathbb{R}^3$ and therefore each voxel location will be represented by $x \in \Omega$. It is also important to note that we will only focus on grayscale images, and the intensity function is defined as $I(x) : \Omega \rightarrow \mathbb{R}$. Furthermore, we define our own segmentation algorithm as a classifier represented by $y(x) : \Omega \rightarrow \{0, 1\}$, which classifies each $x \in \Omega$ to either 1 or 0, which correspond to the ROI and the background, respectively. Besides our own classifier, we will have an ensemble of N predefined predictions $P = \{P_1, P_2, ..P_N\}$, where each element is also a segmentation classifier, that is $\forall P_k \in P, P_k : \Omega \rightarrow \{0, 1\}$. Lastly, since we are working with an Active Learning approach and because the RW requires some

[1] http://www.miccai.org/

initial labeling, a user will provide some labeled data. This data is represented by a set T for which each element is represented by $(X, Y) \in T$, where $X \in \Omega$, $Y \in \{0, 1\}$ and each X is unique.

2.1 Random Walker algorithm

The classifier we will use for the image segmentation is the Random Walker algorithm [2]. This is an algorithm that, using a graph datastructure, assigns probabilities for each pair of neighbouring voxels. Together with a small set of labeled data T, it calculates for each unlabeled $x \in \Omega$ the total probability of arriving at X for each $(X, Y) \in T$. The classification $y(x)$ will then equal the label Y for which the probability of x reaching X was the greatest, for $(X, Y) \in T$. Note that the algorithm also works for 2D images, but as the focus of this research is on 3D images only, the term voxels is used rather than pixels.
Initial labeling The RW requires a set T of initial training data, which a user should provide. The method proposed by [5] uses a functionality called 2D Livewire [1], which helps a user to interactively draw accurate contours at 2D slices around the ROI. However, the RW requires labeled voxels, rather than one or more drawn contours, so this user input has to be converted to fit the purpose. As the paper does not explicitly state how they did this, a decision has to be made. One thing we can do is, for each contour drawn, label all pixels outside the contour as background and the rest as foreground. However, this would assume the contours are drawn perfectly which would require alot of effort from the user. Therefore, we applied a dilation method to thicken the contour before labeling the outside and inside of the contour as background and foreground respectively. This results in having unlabeled pixels at the location of the drawn (dilated) contour, thus keeping some degree of freedom for inaccuracies from the user. The unlabeled pixels are then naturally labeled according to the RW's probability assignment.

(a) (b) (c)

Fig. 2: Example of our workflow with a) being the contour(s) a user provides and b) the labels obtained from the dilated contour, with brown and blue being the ROI and the background respectively. Lastly, c) is the resulting segmentation from the basic RW for this specific 2D slice.

Classification based on weighted graph An important preprocessing step in the algorithm is the creation of a weighted graph. Based on the image to be classified, a graph $G = (V, E)$ is created, where each vertex $v \in V$ corresponds to a unique $x \in \Omega$ and each edge $e \in E$ corresponds to a connection between two neighbouring voxels $v_i, v_j \in V$, denoted by $e_{i,j}$. The weight of an edge $e_{i,j}$ is then defined by the following formula:

$$w_{i,j} = exp[\frac{-\beta}{10 * \sigma_{intensities}}(I_i - I_j)^2)] \tag{1}$$

where for any k, I_k is the voxel intensity of the x corresponding to v_k, and β is the only parameter of the algorithm. Furthermore, $\sigma_{intensities}$ refers to the Standard Deviation of all the voxel intensities in the input image such that $\sigma_{intensities} \in \mathbb{R}$.

It is important to notice that there exist different implementations of such a Random Walker's weight function in image segmentation. However, as our goal is to modify an existing library method (for details see section 3.1), we decided to stick to their definition, and therefore we use Eq.1. Lastly, as described in [2], by structuring the problem in this way using a weighted graph, it can be rewritten to a combinatorial Dirichlet problem. Finding the solution then equals to solving a linear equation with as many unknowns as there are unlabeled voxels.

2.2 Integrating AI predictions

To minimize the labeling effort of the user while improving the intermediate segmentation results, we will extend the RW algorithm by integrating existing segmentation predictions generated by a Bayesian Deep Neural Network. To have more certainty about the correctness of these results, we will use an ensemble of predictions for the same input image and ROI.

In the original algorithm, the classification is based on the weights assigned to neighbouring voxels, which depend on the similarity of local intensity, and the input parameter β (see Eq.1). Therefore, we propose a method to modify this weight function by making it depend on the predictions in the ensemble P. We will focus on different approaches of how to process P and combine it as a weighted sum together with the original function. Lastly, we will propose a method to make the weights used in this weighted sum adaptive.

Mean Prediction First, for each voxel in the image, we will calculate the average prediction and again use 1 to create the weighted graph, but now on these averages instead of on the image intensities. The new weight function is then defined by:

$$w_{i,j} = exp[\frac{-\beta}{10 * \sigma_{mean\_pred}}(\frac{1}{N}\sum_{n}^{N} P_{n,i} - P_{n,j})^2] \tag{2}$$

where $P_n \in P$, $N = |P|$ and for every k, $P_{n,k} \in \{0, 1\}$ is the predicted classification of P_n at the x corresponding to v_k. Furthermore, σ_{mean_pred} denotes the Standard Deviation of the mean of the mean of all predictions, and again

$\sigma_{mean\_pred} \in \mathbb{R}$. With this, we can interpret voxels corresponding to a high average prediction value, to a higher chance of belonging to the ROI, and vice versa. As Eq.1 ensures by definition that regions of similar intensity are labeled similarly, we aim to make the RW less likely to cross the predicted segmentation boundary with this new formula.

Unanimous Vote Secondly, it is important to remember that the predicted segmentations can be erroneous. Therefore, besides the previously mentioned method, we will also try to improve the segmentation algorithm by only looking at unanimous votes, and ensure an inconclusive value assignment for non-unanimous predictions. For this, instead of directly taking the difference between two average predictions, we do this only for cases where they are both unanimously voted, which means for voxels where the average predicted label is either exactly 0.0 or 1.0. For all the other cases, this value will manually be set to 0.5 to remain indecisive. To be more specific, the new weight function will be defined by:

$$w_{i,j} = exp[\frac{-\beta}{10 * \sigma_{mean\_pred}}(\frac{1}{N}\sum_{n}^{N} d(P_{n,i}, P_{n,j}))] \tag{3}$$

where:

$$d(m,n) = \begin{cases} 0.5 & \text{if } 0 < m < 1 \text{ or } 0 < n < 1 \\ m - n & \text{else} \end{cases} \tag{4}$$

Combine with the original weight function Furthermore, for both of the new defined functions (Eq. 2 and Eq. 3), we will combine it with the original weight function. We will do this by seperately calculating both the orignal weights and the chosen extended weights, and adding them together using some value for α. This will be defined by the following formula:

$$w_{i,j} = \alpha * w_{original\text{-}i,j} + (1 - \alpha) * w_{pred\text{-}i,j} \tag{5}$$

Here, $w_{original}$ represents the original weight function (Eq.1) and w_{pred} is the extended weight function based on only the predictions, and both Eq.2 and Eq.3 will be used seperately to evaluate this. The value for α indicates how much we will depend on the original weight function based on the image intensities only. With this we aim to improve the RW by integrating the predicted results, and at the same time being able to account for inaccuracies in the predictions by also using the original weight function.

Adaptive weighted sum Up untill now, we have used a predefined value for α that is the same throughout all iterations and for every voxel in the image. This approach limits the algorithm from dynamically choosing which method to rely more on, the one based on the image intensities or one based on the predictions. Where are convinced to believe that in this way the algorithm does not uses both data in its best interest, therefore we propose the Adaptive Alpha method.

Instead of calculating both weight functions and constructing a weighted sum, we will now sum the data itsself and use the original weight function (Eq.1) on this data. The new weight function then becomes:

$$w_{i,j} = exp[\frac{-\beta}{10 * \sigma_{intensities}}(S_i - S_j)^2)] \tag{6}$$

where S is the sum of the average predictions and the image intensities:

$$S_i = \frac{1}{\sigma_{mean\_pred}} \frac{1}{N} (\sum_{n}^{N} P_{n,i}) + \frac{I_i}{\sigma_{intensities}} \tag{7}$$

Here, we have divided both the mean predictions aswell as the image intensities by their corresponding Standard Deviation. By doing this we aim to ensure a better comparability between the two by normalizing the variation.

3 Experimental Setup and Results

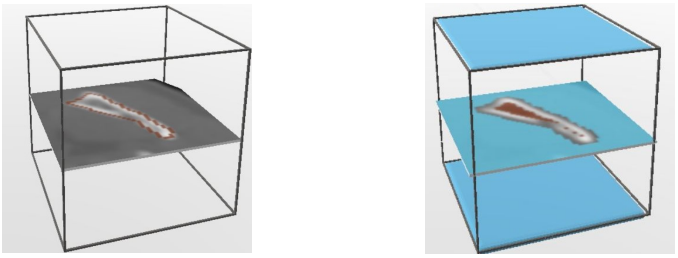
3.1 Experimental setup

In this subsection we will explain how we setup our experiment and which assumptions have been made.

Simulating initial labeling We simulated the user input by assuming the user behaves perfectly. This means that instead of manually drawing contours around the ROIs, we directly used the ground truths and generated the contours around this region using a library method findContours. To keep in mind the goal of minimizing the labeling effort, we choose to always label only one arbitrary 2D slice initially. We assume that for the initial labeling, a user also provides the first and last index on the z-axis where the ROI is present. To simulate this input we calculate a boundary box for the z-axis using the ground truth, with a pixel offset of 1. With respect to these boundaries, we have chosen to initially always label the middle slice aligned to the x,y-plane. Note that for cases where the ROI is not present in the middle positioned slice at all, we choose the x,y-plane aligned slice, containing the ROI, that is closest to this middle position.

Converting initial contours to labels For the conversion of the initial contours around the ROI to labels, we made use of a scikit-image's Morphology method to apply a dilation. Initially a kernel of size 3x3 is used. However, a problem arises when the area inside a contour is close to the kernel size, which result in a morphological closing. That means in this case that the area inside the contour is now fully filled and no foreground labels will be added there. This often results in the RW labeling this 2D slice fully as background, because the unlabeled voxels are surrounded by background labels in this slice. To fix this problem, we check for each contour whether this is the case and iteratively reduce the size of the kernel if needed, until some space is left for the foreground labels. After this, the outside of the contour is filled with value 2 using scikit-image's floodfill method. Subsequently, the pixel values of the dilated contours

and the region inside of it are flipped, making label 1 correspond to the ROI, and the neighbourhood around the drawn contours remain unlabeled. Lastly, as the initial labeling always happens on 2D slices that are x,y-plane aligned, the RW initially has no information about boundaries on the z-axis. This can result in having false positives on x,y aligned slices where no ROI is present, and since we similate user input only in the form of contours around the ROI, this is not possible to fix on x,y aligned slices. We are aware that these errors should easily be corrected by user input on rotated slices, but we consider this sometime to account for in future work. Therefore to fix this, we initially always label the first and the very last slice as fully background.



(a) (b)

Fig. 3: Simplified example of the initial labeling

Segment the image The next step is to run the RW with a desired value for β. Similarly to [5], we choose the value for β by qualitatively observing which produced the best results. For the original weight function we choose $\beta = 20$. Furthermore, to ensure reliability in the comparisons against the extended weight functions, we decided to be consistent with this value for all methods. Lastly, for the random walker algorithm we used a library method from scikit-image[2] which is an open source image processing library. We replicated the function used by the RW to assign weights to edges in a graph and modified it to fit our purposes.

Simulate additional labeling After the segmentation is done, and a plane of most uncertainty is provided to the user, we again need to simulate labeling. This will be done exactly the same as for the initial labeling described in 3.1, namely by generating a contour around the ground truth. Note that this time this is been done only on the proposed slice instead of on the middle slice as for the initial labeling.

Converting additional contours to labels The next step is to convert the new user input again to labels, and union it with the labels obtained by all previous user input. For this conversion we use the same method as explained in 3.1, except that we don't need to label the first and last slice of the 3D image

[2] https://scikit-image.org

anymore, as we only have a 2D slice to label this time. The new labeled slice is then inserted into the previous labels for all non-zero values of the new labeled slice. The purpose of ignoring zero labeled voxels here, is to make sure we avoid replacing labeled voxels by value zero, which would represent unlabeling the voxel.

Empirical data As test data we used a subset of a MICCAI dataset consisting of CT-scans from 10 different patients and the corresponding ground truth labeling 9 different parts of the human brain [4], the same parts for each patient. For the data used as AI generated predictions, we used ensembles of segmentations generated by a Bayesian DNN [3]. The DNN was trained on a subset of the patients in the MICCAI dataset and tested on the rest of the dataset, resulting in 6 predicted segmentated images.

Synthetic data Besides real-world data, we created synthetic data to highlight the main differences between the proposed methods. For this we created a 2D image containing a square region centered in the middle that has been blurred to simulate changes in intensity around the edges of the square. We also create an arbirary ground truth, indicating a square region centered such that the desired segmentation boundary is surrounded by inconsistent intensities of the input image. Then, we simulated erroneous predictions by replicating the ground truth 6 times, and randomly shifting each in a random direction. Lastly, we simulated the initial contours by creating a 1-dimensional line-based contour, to fit the purpose for this 2D image example.

Technical details Regarding the coding environment, all code is written in Python (3.10), and in order to easily work with a large set of available libraries we use Anaconda as a software distribution for Python. Furtermore, OpenCVs[3] is also a library we used, most importantly to create contours of images. Lastly, the repository consisting of our reproduction of [5] is made public and can be accessed in a Github[4] repository[5].

3.2 Results

In this section we will present and describe the results obtained. We will first show the results of our methods on randomly generated synthetic data. For this we will compare the results of the original method, the Mean Predictions method and the Adaptive Alpha method. Futhermore, we will show the DICE Coefficients for the different organs in our MICCAI dataset for our different methods.

[3] https://opencv.org/
[4] https://github.com/
[5]

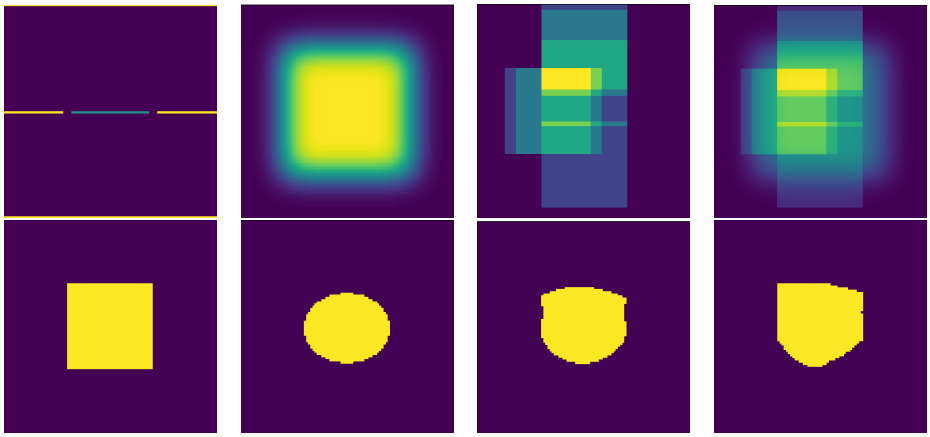
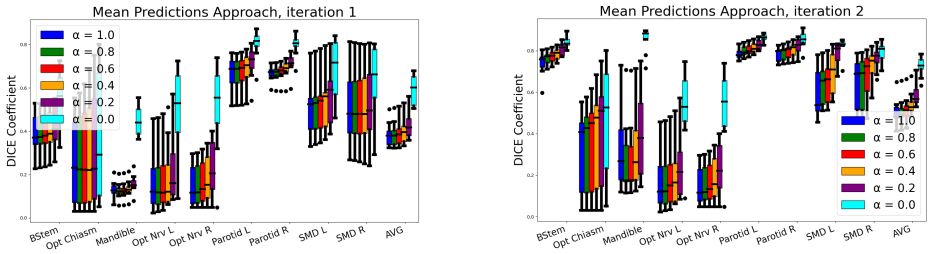
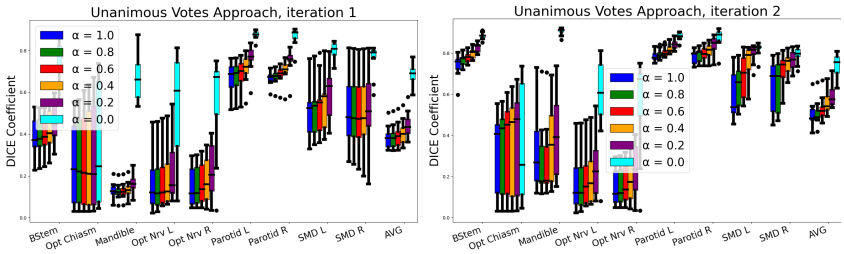


Fig. 4: 2D example synthetic data. First row from left to right: Contour, Input image, Mean predictions, Summed data as Eq.7. Second row from left to right: Ground Truth, Original segmentation result, Mean Predictions based segmentation result, Adaptive Alpha based segmentation result. The resulting DICE Coefficients of the three segmentations are 0.7823, 0.8624, 0.9093 for Mean Prediction, Unanimous Vote and Adaptive Alpha respectively.



(a) (b)

Fig. 5: Boxplot showing the DICE scores for the MICCAI dataset. This figure shows the results of combining the original weight function with the Mean Prediction based, for different values of α. The left (a) shows the results after the first iteration, and the right (b) after the second. The x-axis indicates the organ of interest with the last position being the average over all organs.

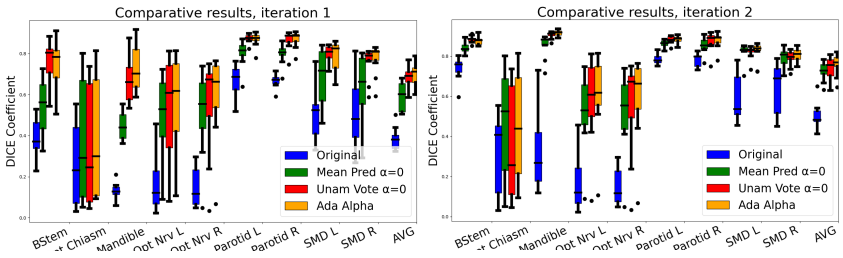


(a) (b)

Fig. 6: Boxplot showing the DICE scores for the MICCAI dataset. This figure shows the results of combining the original weight function with the Unanimous Votes based, for different values of α. The left (a) shows the results after the first iteration, and the right (b) after the second. The x-axis indicates the organ of interest with the last position being the average over all organs.

Figure 5 shows the results for combining the original weight function with the Mean Prediction based. It shows the DICE Coefficient of 9 patients in our MICCAI dataset, for each ROI. It also shows the average DICE Coefficient over all different ROIs. It can be seen that for $\alpha = 0.0$ the method always outperforms all the others. More specifically, as α increases, the overall performance has a corresponding improvement. Although the method performs the best for $\alpha = 0.0$, it is also worth to notice that this seems to have a high variance, especially for the first iteration. For the parotids (Parotid L and Parotid R) the algorithm works the best, especially for the first iteration. For the second iteration, using $\alpha = 0.0$, the results have a small variance for every organ except for the optics (Optic Chiasm, Optic Nrv L and Optic Nrv R).

Furthermore, figure 6 shows the results for combining the original weight function with the Unanimous Vote based. It shows the DICE Coefficient of 9 patients in our MICCAI dataset, for each ROI. It also shows the average DICE Coefficient over all different ROIs. It can be seen that for $\alpha = 0.0$ the method again outperforms all the others and has an overall corresponding improvement in performance as α increases. For all values of α, there is more consitency in the performance for the brainstem (BStem) and the parotids (Parotid L and Parotid R). The overall performance is the worst for the optic chiasm (Opt Chiasm), the mandible (Mandible) and the optic nerves (Opt Nrv L and Opt Nrv R). However, if we only look at $\alpha = 0.0$, out of all organs it performs the best on the mandible after the second iteration.



(a) (b)

Fig. 7: Boxplot showing the DICE scores for the MICCAI dataset. This figure shows the comparative results between the Mean Predictions Approach and the Unanimous Votes Approach (both using $\alpha = 0.0$), the Original RW and the Adaptive Alpha Approach. The left (a) shows the results after the first iteration, and the right (b) after the second. The x-axis indicates the organ of interest with the last position being the average over all organs.

Lastly, figure 7 shows the results of the Adaptive Alpha Approach. For comparativity, we plotted the boxplots against the Original RW, the Mean Predictions Approach and the Unanimous Votes Approach. For the latter two, we used both $\alpha = 0.0$ as this optimized their results. It can be seen that the Adaptive Alpha Approach performs the worst on the optics, while having an overall performance that is slightly better than the other three methods.

4 Responsible Research

It is of great importance that a research is done responsible, meaning it should be reproducible and ethical aspects and possible risks should be taken into consideration.

First of all, we made sure our research is reproducible by providing the link to a GitHub repository in section 3. We also made sure to provide our code with comments explaining parts as clear as possible. Lastly, we tried to explain every decision that deviates from the replicated proposal [5] as precise as possible in this report.

Another important aspect of responsible research, is to consider ethical aspects. One of the most important things to consider for this research is the data used. Most importantly with respect to privacy rules, there are limitations in the data we can use. Fortunately, we used a database which has been made available to the public in the hope it can be helpful to others.

Furthermore, it is very important to consider possible risks in a responsible research. In our research the first and most straightforward risk, is being biased to our test data. Because of the need of AI generated segementation results for this research and because of the limited time we had, it was not possible to test our method on the whole MICCAI dataset, but only a subset for which

the predictions where generated. As this subset is a rather small dataset (6 predictions of 9 ROIs for 10 different patients), the chances of overfitting occur. This is unfortunately an undesired risk that could not be solved in the given time for this research.

5 Discussion

In this section we will discuss the results presented in 3.2.

First of all, it can be seen that for all methods, it performs the worst for the optic organs, which are the Optic Nerve (L and R) and the Optic Chiasm. This is not surprising, as the optic organs have one of the smallest, if not the smallest, ROI out of all the other organs in our dataset. For which some dimensions are only a few pixels in length. This makes them more sensitive to artifacts in the image and most importantly there is simply less information to obtain.

Another interesting result is that $\alpha = 0$ always outperformed the other possibilities for α, by far. Again, this is not surprising, as we have only focused on two iterations. And as we decided to only annotate one new slice per iteration, the original RW has still little information to utilize after two iterations. Meanwhile the predictions provide more information about the ROI for these first two iterations, admidittely erroneous possibly, but for the first two iterations these errors do not compare to the errors of the orginal RW for such a small number of iterations. The fact that the first two iterations do not ensure a high performance for the original RW can also be verified by observing the results in [5]. It can be seen that the most rapid increase of DICE Coefficient there, is present during the few initial iterations. This confirms why using $\alpha = 0$ always outperformed the other chosen values.

Another observation is that the Unanimous Votes Approach performs slightly better than Mean Prediction Approach. This is not what we expected, as the decision to discard the influence of all non-unanimous votes was arbitrarily chosen. While actually we could have chosen any other threshold for this, for example only taking the two most similar voted values etc. We expected that since the value for this threshold was not based on any actual data, this would perform worse than the Mean Prediction Based Approach. However, it actually makes sense as unanimous votes provide the most certainty in its prediction, making this approach less sensitive to ensembles of predictions where a high variance is present.

Another remarkible observation is the results for the Mandible. It seems to have one of the best performances with minimum variance, only for the methods using $\alpha = 0$ and the Adaptive Alpha Approach. Meanwhile, for all the other methods, the Mandible produces one of the worst results. We believe that this is due to the fact that the automated slice annotation, after the first segmentation does not add much information. It could be that for example the chosen slice of most uncertainty has a small ROI. However, this results in a minimum increase of performance between the two iterations, which can also be observed in the boxplots. It then makes sense that for $\alpha = 0$ the methods are not affected by

this, as they focus only on the predictions. Also, the Mandible is a relatively big organ, which also means that it could take longer for the original RW to obtain a high performance, which on the other hand does not apply for the prediction based methods.

Lastly, we observe that the Adaptive Alpha Approach slightly outperforms all other methods for both iterations. This confirms the hypothesis that by summing the data beforehand, the value of each voxel is depending on both the predicted mean and the intensity, which results in a more adaptive ratio compared to combining the weight functions afterwards.

For future work, a recommendation would be to test our methods on different ensembles which vary in accuracy. As the ensembles we used are relatively accurate which can result in our methods to be biased to relying heavily on the ensembles. This is also shown to be the case as methods that focus only on the predictions outperformed almost all the other methods that take into account the image intensities. For another future work, we believe the algorithm can be improved even more by taking into account local regions for each voxel. As the weight functions we propose either handle the 3D image as a whole, or voxel-wise, we believe that by depending more on a local 3D window for each voxel, we can make the methods less sensitive for outliers in the image. This is a general approach that can apply to different parts of the algorithm. An example could be to base the difference in intensities on the average intensities in a 3D neighbourhood around a voxel, etc. Additionally, this local information can be used to handle areas where the image intensities are fuzzy, resulting in an indication of how much the weight function should rely on the predictions for a voxel in this area. Lastly, for the future it can be valuable to focus on a way to define an indication of certainties, as for example by looking at local neighbourhood, for the voxels. These certainties can then be used to make the method of Adaptive Alpha even more adaptive, by taking into account the reliability of both the predictions and the image intensities. We have tried multiple methods that make use of this by taking the standard deviation, however, this was either time consuming or it did not perform as expected. For the sake of time we decided that this approach unfortunately requires more effort and thus we excluded these ideas from this research.

6 Conclusion and future work

In this paper we have proposed a method to integrate an ensemble of segmentation predictions, obtained by a BDNN (ALL GENOEMD?), into a Random Walker's weight function, to improve interactive 3D medical image segmentation. We evaluated three methods, Prediction Means Approach, Unanimous Votes Approach and Adaptive Alpha Approach, while comparing to the orginal RW. The Prediction Means Approach focuses on the mean of all predictions in the ensemble, while the Unanimous Votes Approach only takes into account predictions that where all the same throuhout the whole ensemble for a voxel. Furhtermore, the Adaptive Alpha Approach sums the mean predictions with the image

intensities after normalizing their variances, with the aim to have a more adaptive approach for each voxel. All methods are evaluated by showing the average DICE Coefficients of all patients in our dataset, for different ROIs, for the first two iterations. We simulated the intermediate user input by automatically annotating one slice per iteration. With this, our work shows that the Adaptive Alpha Approach outperforms all other methods.

For future work, we recommend to test our methods on data that contains less accurate ensembles of predictions. Lastly, an approach to indicate uncertainties in local regions for both the predictions and the image intensities, and use this to optimise the weighted sum is a desired future work.

References

1. Barrett, W.A., Mortensen, E.N.: Interactive live-wire boundary extraction. Medical Image Analysis **1**(4), 331–341 (1997). https://doi.org/https://doi.org/10.1016/S1361-8415(97)85005-0, https://www.sciencedirect.com/science/article/pii/S1361841597850050
2. Grady, L.: Random walks for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(11), 1768–1783 (2006). https://doi.org/10.1109/TPAMI.2006.233
3. Mody, P.P., de Plaza, N.C., Hildebrandt, K., van Egmond, R., de Ridder, H., Staring, M.: Comparing Bayesian models for organ contouring in head and neck radiotherapy. In: Colliot, O., Išgum, I. (eds.) Medical Imaging 2022: Image Processing. vol. 12032, p. 120320F. International Society for Optics and Photonics, SPIE (2022). https://doi.org/10.1117/12.2611083, https://doi.org/10.1117/12.2611083
4. Raudaschl, P.F., Zaffino, P., Sharp, G.C., Spadea, M.F., Chen, A., Dawant, B.M., Jung, F.: Evaluation of segmentation methods on head and neck ct: Auto-segmentation challenge 2015. In: AutoSeg Workshop and Challenge at MICCAI. pp. 2020–2036 (2017)
5. Top, A., Hamarneh, G., Abugharbieh, R.: Active learning for interactive 3d image segmentation. In: Fichtinger, G., Martel, A., Peters, T. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011. pp. 603–610. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)