IntellEditS: Intelligent Learning-Based Editor of Segmentations

Adam P. Harrison[1,2], Neil Birkbeck[1], and Michal Sofka[1]

[1] Siemens Corporation, Corporate Technology, Princeton, NJ, USA
[2] University of Alberta, Edmonton, Canada

Abstract. Automatic segmentation techniques, despite demonstrating excellent overall accuracy, can often produce inaccuracies in local regions. As a result, correcting segmentations remains an important task that is often laborious, especially when done manually for 3D datasets. This work presents a powerful tool called Intelligent Learning-Based Editor of Segmentations (IntellEditS) that minimizes user effort and further improves segmentation accuracy. The tool partners interactive learning with an energy-minimization approach to editing. Based on interactive user input, a discriminative classifier is trained and applied to the edited 3D region to produce soft voxel labeling. The labels are integrated into a novel energy functional along with the existing segmentation and image data. Unlike the state of the art, IntellEditS is designed to correct segmentation results represented not only as masks but also as meshes. In addition, IntellEditS accepts intuitive boundary-based user interactions. The versatility and performance of IntellEditS are demonstrated on both MRI and CT datasets consisting of varied anatomical structures and resolutions.

1 Introduction

Interactive approaches to segmentation have a proven track record [9]. However, the important task of interactively editing a pre-existing but imperfect segmentation, or *presegmentation*, has not enjoyed much attention [6, 11, 8]. This is unfortunate, as manually correcting segmentations in 3D medical imaging applications can be a time-consuming but necessary task. Local corrections are often needed for fully-automatic segmentation techniques [6, 8], which are powerful and time-saving but still have not matched the performance of interactive tools in key applications [9]. To use all available information, an editor should consider: (1) user interactions, (2) the presegmentation, and (3) the underlying volume or data. These sources should guide an editing algorithm that strives to (1) minimize number of edits to achieve desired result, (2) minimize segmentation error after each edit, and (3) respect user guidance.

Apart from 2D-focused tools [10, 15], many 3D editors function by propagating user corrections from an interaction plane to the larger 3D volume [6, 3, 8]. Propagation can be achieved by minimizing an energy term constrained by user input, the presegmentation, and the volumetric data [6]. Such approaches face several key challenges. First, volumetric data is incredibly rich and varied, making it non-trivial in how to best employ it. Ideally, an editor would use volume data in a way that can generalize to

(a) (b) (c) (d) (e)

Fig. 1: Removing the vena cava from a liver mesh (red) using IntellEditS. (a) User chooses an interaction plane and draws a corrected boundary (green); (b) Mesh is updated in the plane; (c) As shown in a perpendicular plane, mesh boundary is accurately propagated in 3D; (d)-(e) The 3D propagation can also be viewed by comparing before and after surfaces in (d) and (e), respectively. As IntellEditS is executed within an MPR viewer, pre- and post-edit segmentation accuracy can be quickly assessed.

different modalities, imaging qualities, and anatomical structures. Second, many presegmentations are mesh-based, meaning they must be reconciled with the voxel-based volume data. Third, user input should be as intuitive and user-friendly as possible.

This work presents an interactive editor of 3D presegmentations that simultaneously addresses all the above challenges. First, the system allows users to precisely draw new boundaries using lines within the interaction plane. This is called *splice-based interactions*. Second, a discriminative classifier uses foreground and background regions defined by the splice to model voxels inside and outside the object being edited. Classification results of unknown voxels are then incorporated within an energy functional that locally propagates user interactions to the 3D volume. Finally, when editing mesh presegmentations, a soft voxel-based representation is used, reconciling it with the volume data while still retaining a highly accurate boundary. These features culminate in a system entitled Intelligent Learning-Based Editor of Segmentations (IntellEditS) that provides a sophisticated and flexible means to rapidly edit presegmentations. Fig. 1 visually depicts the steps involved in editing.

IntellEditS advances the state of the art of *data-driven* editing through several means. For instance, many editing techniques use interactions, such as brush-based tools [6, 3], that are nonintuitive for clinicians [8] as they do not allow precise correction of boundaries. In addition, state of the art data-driven editors only consider mask-based presegmentations [6, 3, 8], meaning that IntellEditS is the first to simultaneously perform mesh-based and data-driven editing. Finally, the above works only use local differences in volume intensity and do not attempt to model foreground and background voxels. This state of data-driven editing is in contrast to many interactive segmentation techniques, which successfully leverage learning algorithms as they build up a segmentation from scratch [16, 17, 13, 18].

With these advances, IntellEditS fills an important gap in the state of the art of data-driven editing. The algorithmic details of IntellEditS are expounded further in the methodology section (§2). As demonstrated in the results section (§3), using identical parameters IntellEditS can perform effectively on volumetric data coming from different anatomical structures, modalities, imaging conditions, resolutions, and anisotropic characteristics. This work is concluded by a discussion of results (§4).

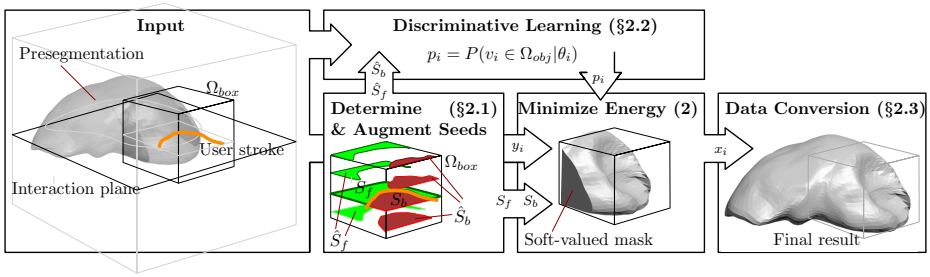


Fig. 2: Algorithm steps of IntellEditS.

2 Interactive Learning-based Editing

IntellEditS partners interactive discriminative classification with energy-based minimization in order to edit mesh- or mask-based representations of anatomical structures. Fig. 2 provides a high-level view of IntellEditS' algorithm steps.

First, a user corrects a presegmentation on a 2D interaction plane using splices. Foreground and background seed points, S_f and S_b respectively, are calculated based on user input and the presegmentation. If the presegmentation is a mesh, it is converted to floating-point distance map, y_i, where i indexes individual voxel locations within the volume. IntellEditS only operates on a cropped region of the volume, Ω_{box}, which is specified using a bounding box around the splice line and the region of the presegmentation contained within said splice.

An augmented set of seed points, $\hat{S}_f$ and $\hat{S}_b$, serve as inputs to IntellEditS' classifier, which learns how to discriminate between foreground and background regions based on a pool of features for each voxel. Features are denoted using an ordered vector of values $\boldsymbol{\theta}_i$. In formal terms, define a variable $p_i \in [0, 1]$, which describes the probability that voxel v_i is in the foreground. Based on the trained model, the classifier calculates the following posterior probability for each non-seed voxel:

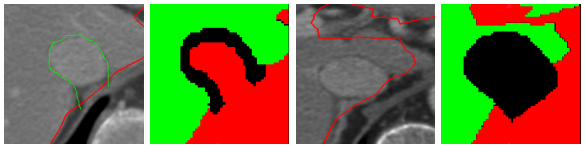
$$p_i = P(v_i \in \Omega_{obj}|\boldsymbol{\theta}_i), \tag{1}$$

where Ω_{obj} denotes the true boundary of the anatomical object being edited.

If the p_i values were completely accurate, the editing task would be finished. However, as complete accuracy cannot be guaranteed, p_i values are employed as part of an energy minimization formulation that incorporates the presegmentation, user seeds, and a regularizer. Formally, the energy formulation can be expressed as:

$$E(\mathbf{x}) = \sum_{e_{ij}} w_{ij}(x_i - x_j)^2 + \gamma_i \sum_{i \in \Omega_{box}} (y_i - x_i)^2 + \alpha \sum_{i \in \Omega_{box}} (p_i - x_i)^2 \tag{2}$$
$$\text{s.t.,} \begin{cases} x_i = 1, v_i \in S_f \\ x_i = 0, v_i \in S_b \end{cases},$$

where x_i are soft output potential values and w_{ij} denotes the weights assigned to the graph edges e_{ij} connecting each vertex v_i or voxel. The first summation acts as a regularizer, ensuring coherent output potentials. The second summation incorporates the



(a) (b) (c) (d)

Fig. 3: Calculating Seed Points. (a) an edit and the presegmentation; (b) corresponding foreground (S_f) and background (S_b) seed points in green and red, respectively; (c) volume and presegmentation above the editing plane; (d) corresponding foreground ($\hat{S}_f$) and background ($\hat{S}_b$) *augmented* seed points in green and red, respectively.

presegmentation, where γ_i is a local parameter controlling its influence. IntellEditS uses the same γ_i and w_{ij} values as Grady and Funka-Lea [6]. However, floating-point presegmentation values are used instead of binary values. Finally, the third summation incorporates the per-voxel probabilities of being in the foreground. The influence of the p_i values are controlled by α, which has a context sensitive value explained in Sect. 2.2. IntellEditS minimizes (2) using the random walker algorithm [5].

Apart from using splice-based interactions and accommodating mesh-based presegmentations, the third summation in (2) represents one of the most important departures from the state of the art. While combining learning with energy minimization has proven successful in interactive segmentation, it has not been used to locally *edit* presegmentations. Other energy-minimization-based editors only employ volume data to calculate w_{ij} [6, 3, 10, 15]. As a result, these methods neglect the highly informative ensemble of *local* foreground and background voxel features that users implicitly specify during edits. Thus, the strength of (2) rests on its use of all available sources of information—foreground/background features, presegmentation, and user seeds.

In order to apply (2) in a concrete implementation, the generation of seed points must be specified (§2.1). The seed points define the fixed regions of the segmentation and are also used to train the discriminative classifier (§2.2). After solving (2), the voxel-based output potentials are converted to a mesh-based representation (§2.3) or thresholded for mask-based presegmentations.

2.1 Determining and Augmenting Seeds

When a user draws a splice intersecting a presegmentation, she is drawing a new 2D boundary. As Fig. 3(a) and (b) demonstrate, corresponding seed points, S_f and S_b, can be inferred by flood filling the new boundary. These are only extracted from the 2D interaction plane. A buffer ensures any voxels on or close to the new boundary are not chosen as seed points, allowing IntellEditS to settle on a precise iso-contour instead.

S_f and S_b provide reliable training samples. However, since IntellEditS must propagate an edit away from the interaction plane, training samples should be extracted from the larger volume. As well, any learning algorithm's performance partly hinges on the number of training samples used. For this reason, seed points are augmented into two larger sets, $\hat{S}_f$ and $\hat{S}_b$. These are calculated by projecting locations of out-of-editing-plane voxels onto the editing plane itself. If their 2D projected location is far enough

from the user's splice then they are included into $\hat{S}_f$ ($\hat{S}_b$) if they are in the presegmentation foreground (background). One such example is provided by Fig. 3(c) and (d). While there is a possibility of incorrectly labelled samples entering the training set, experience indicates the impact, if any, to be minimal.

2.2 Discriminative Learning

Both generative [16, 17] and discriminative [13, 18] models have been used to interactively segment visual data. While powerful, generative-model performance hinges on selecting an appropriate model and the correct feature(s) to examine. This is a challenge when faced with different modalities, anatomical structures, image qualities, and editing contexts. Since IntellEditS' goal is to operate effectively even on dataset types not encountered during development, it employs discriminative classification to directly model posterior probability. Classification features are drawn from a large pool of 3D Haar wavelets, where their relative influence varies based on the dataset.

This work uses a Probabilistic Boosting Tree (PBT) [14], whose nodes are composed of AdaBoost classifiers. Unlike much of previous work using PBTs, classification must execute at interactive speeds. A recent work by Birkbeck *et al.* documented a GPU PBT implementation [4], which also included fast calculation of 3D Haar features. However, since classifiers were still trained offline using multiple volumes, the algorithm was not meant to be interactive and did not address speeding up training. In contrast, IntellEditS requires *both* fast training and detection using a *single* volume. As a result, IntellEditS extends Birkbeck *et al.*'s work by implementing an interactive-speed PBT (I-PBT) for single-volume classification. Training is GPU-implemented using CUDA.

Based on feature values of $\hat{S}_f$ and $\hat{S}_b$, IntellEditS trains the I-PBT to discriminate between foreground and background voxels. After training, the detection accuracy, $\rho \in [0, 1]$, of the I-PBT in classifying the training samples is calculated. The weight parameter, α, in (2) is then set to 0.5ρ, providing an automatic and performance-based tuning of the editor.

2.3 Data Conversion

Unlike previous works, IntellEditS' goal is to edit meshes in addition to masks. Nonetheless, outside of pre- and post-processing steps to convert data, IntellEditS works within a voxel-based domain. This enables parallel execution of many of the per-voxel tasks in training and detection. As well, resampling of the volume and its features, commonly needed in simplex-based representations of voxel data, is avoided.

To retain the high resolution of mesh-based presegmentations, IntellEditS converts them to soft-valued voxel representations based on distance-map calculations. For the mesh-to-distance-map direction, IntellEditS uses the fast pseudo-normal method [2]. Converting the distance map back to a mesh is accomplished using the marching cubes algorithm [12]. The authors' experience indicates that this conversion does not affect visual quality of the boundary. Since (2) works within a $[0, 1]$ range, distance-map values must be mapped to this range using a scale and offset. The reverse mapping must also be executed before converting output potentials to a mesh.

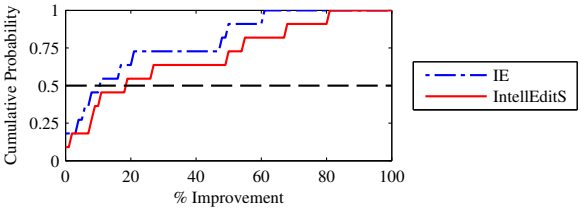


Fig. 4: Cumulative probability distribution of improvements in SPS error after IE and IntellEditS corrections. The median improvements of the IE and IntellEdits were 11 and 19% respectively and third-quartile improvements were 48 and 55%, respectively.

3 Results

Experiments tested IntellEditS' performance on 11 CT and MRI datasets of different anatomical structures, resolutions, and anisotropies. Using the *same* single splice, experiments compared IntellEditS' performance against an intensity-based editor (IE) that only uses the first two summations in (2). As such, IE does not model foreground and background voxels and is similar to Grady and Funka-Lea's approach [6], except it has been significantly modified to use splice interactions and edit meshes. Each tool's performance was gauged using symmetrical point-to-surface (SPS) error against a manually-annotated ground truth in the Ω_{box} region. These results were then compared against the original SPS error of the presegmentation.

All experiments used the same parameter values, detailed in §2. For all datasets, the I-PBT was configured to have a depth of 3 with 10 weak classifiers at each node. On average, editing time consumed 3 seconds on an 8 core machine with an NVIDIA GeForce 9800 GT video card. As Fig. 4 illustrates, IntellEditS is able to more effectively reduce segmentation errors after the same user interaction. Fig. 5 visually demonstrates the effectiveness of IntellEditS by depicting representative qualitative results drawn from the quantitative experiment.

4 Discussion and Conclusion

This work presented an interactive editing tool called IntellEditS that represents a novel and powerful way to correct presegmentations in a 3D context. Unlike previous data-driven editors, IntellEditS can edit meshes. Users employ intuitive splice-based interactions to correct presegmentations on a 2D interaction plane, which IntellEditS propagates in 3D through a combination of discriminative learning coupled with energy minimization. The practical benefits and versatility of IntellEditS were demonstrated in quantitative and qualitative experiments composed of challenging datasets of various modalities, anatomical structures, resolutions, and anisotropy. Comparing against an intensity-based editor and using the same user input, these experiments demonstrated that IntellEditS can output a mesh significantly closer to ground truth, saving valuable end-user time and effort. Interesting directions of future work include combining IntellEditS with tools that can help choose suspicious interaction planes [1], providing the

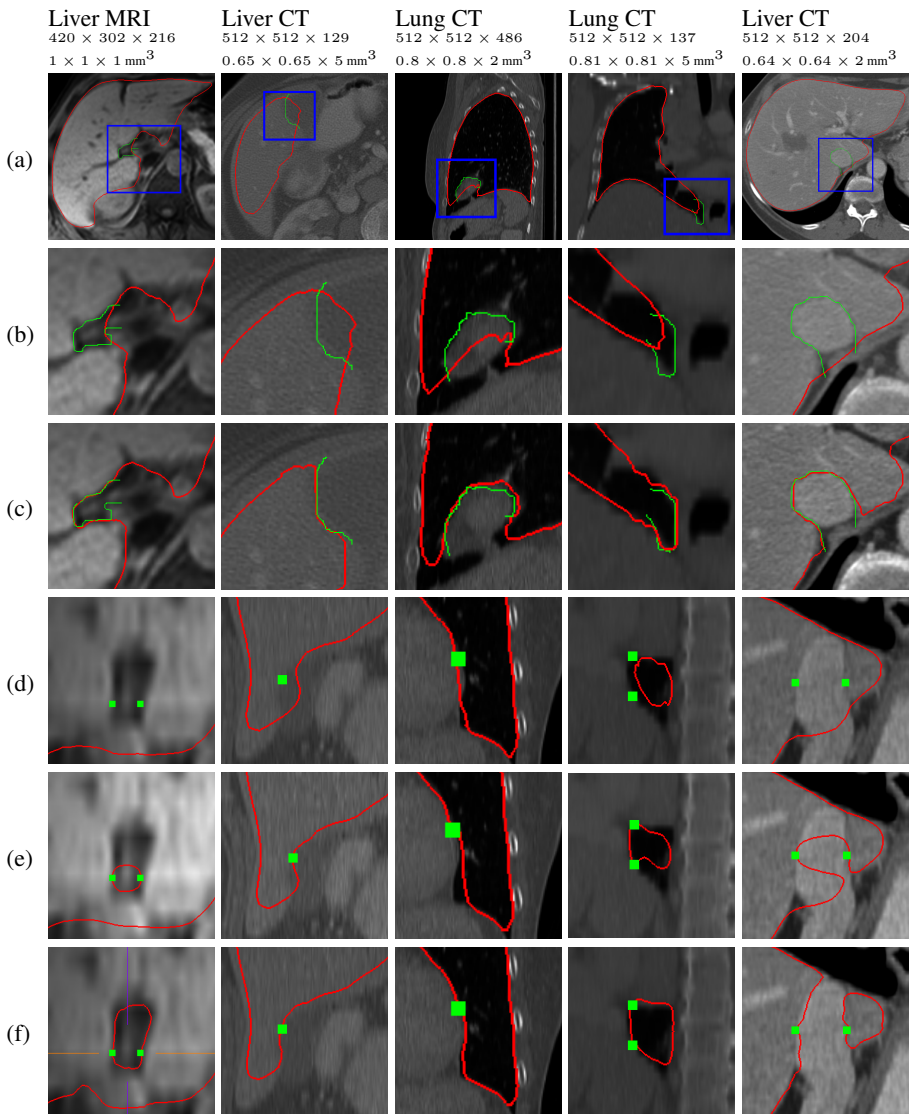


Fig. 5: Qualitative comparison of IntellEditS vs IE using the same *single* splice. (a) overall view of presegmentation and editing splice; (b) zoomed-in view of (a); (c) corrected mesh; (d) cross-section view of presegmentation; (e) and (f) same view as (d) but after the correction produced by IE and IntellEditS, respectively. Dataset dimensions and resolutions are also provided.

option to use live-wire techniques [7], and incorporating online learning into the editing process to incrementally learn a more robust model from consecutive interactions.

5 Acknowledgments

We thank S.Kevin Zhou, Noha El-Zehiry, and Enrico Kuhn for valuable discussions, feedback, and code that contributed to this paper.

References

1. Andrew Top, G.H., Abugharbieh, R.: Active Learning for Interactive 3D Image Segmentation. In: MICCAI 2011. LNCS, vol. 6893, pp. 603–610. Springer, Heidelberg (2011)
2. Baerentzen, J.A., Aanaes, H.: Generating Signed Distance Fields From Triangle Meshes. Tech. rep., Informat. and Math. Mod., Tech. Univ. of Denmark (2002)
3. Beichel, R., Bauer, C., Bornik, A., Sorantin, E., Bischof, H.: Liver segmentation in CT Data: A Segmentation Refinement Approach. In: Proceedings of 3D Segmentation in the Clinic: A Grand Challenge. pp. 235–245 (2007)
4. Birkbeck, N., Sofka, M., Zhou, S.: Fast Boosting Trees for Classification, Pose Detection, and Boundary Detection on a GPU. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on. pp. 36 –41 (2011)
5. Grady, L.: Random Walks for Image Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 28(11), 1768–1783 (Nov 2006)
6. Grady, L., Funka-Lea, G.: An Energy Minimization Approach to the Data Driven Editing of Presegmented Images/Volumes. In: MICCAI (2). pp. 888–895 (2006)
7. Hamarneh, G., Yang, J., Mcintosh, C., Langille, M.: 3D live-wire-based semi-automatic segmentation of medical images. In: Proceedings of SPIE Medical Imaging: Image Processing 5747. pp. 1597–1603 (2005)
8. Heckel, F., Moltz, J.H., Bornemann, L., Dicken, V., Bauknecht, H.C., Fabel, M., Hittinger, M., Kieling, A., Meier, S., Psken, M., Peitgen, H.O.: 3D contour based local manual correction of tumor segmentations in CT scans. In: SPIE Med. Img. vol. 7259, pp. 1–9
9. Heimann, T., van Ginneken, B., et al.: Comparison and Evaluation of Methods for Liver Segmentation From CT Datasets. IEEE Trans. Med. Imaging 28(8), 1251–1265 (2009)
10. Jagadeesh, V., Manjunath, B.: Interactive graph cut segmentation of touching neuronal structures from electron micrographs. In: Image Processing (ICIP), 2010 17th IEEE International Conference on. pp. 3625 –3628 (2010)
11. Kang, Y., Engelke, K., Kalender, W.A.: Interactive 3D editing tools for image segmentation. Medical Image Analysis 8(1), 35 – 46 (2004)
12. Lorensen, W.E., Cline, H.E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics 21(4), 163–169 (1987)
13. Santner, J., Unger, M., Pock, T., Leistner, C., Saffari, A., Bischof, H.: Interactive Texture Segmentation using Random Forests and Total Variation. In: British Machine Vision Conference (BMVC) (2009)
14. Tu, Z.: Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. vol. 2, pp. 1589 –1596 Vol. 2 (2005)
15. Yang, H.F., Choe, Y.: An Interactive Editing Framework for Electron Microscopy Image Segmentation. In: Advances in Visual Computing, LNCS, vol. 6938, pp. 400–409 (2011)
16. Yang, Q., Tang, X., Wang, C., Ye, Z., Chen, M.: Progressive Cut: An Image Cutout Algorithm that Models User Intentions. MultiMedia, IEEE 14(3), 56–66 (2007)
17. Yang, W., Cai, J., Zheng, J., Luo, J.: User-friendly Interactive Image Segmentation Through Unified Combinatorial User Inputs. IEEE Trans. Image Process 19(9), 2470 –2479 (2010)
18. Zhao, Y., Zhu, S.C., Luo, S.: CO3 for Ultra-fast and Accurate Interactive Segmentation. In: Proceedings of the International Conference on Multimedia. pp. 93–102. ACM (2010)