Elastic Boundary Projection for 3D Medical Image Segmentation

Tianwei Ni[1], Lingxi Xie[2,3](✉), Huangjie Zheng[4], Elliot K. Fishman[5], Alan L. Yuille[2]
[1]Peking University [2]Johns Hopkins University [3]Noah's Ark Lab, Huawei Inc.
[4]Shanghai Jiao Tong University [5]Johns Hopkins Medical Institute
{twni2016, 198808xc, alan.l.yuille}@gmail.com zhj865265@sjtu.edu.cn efishman@jhmi.edu

Abstract

We focus on an important yet challenging problem: using a 2D deep network to deal with 3D segmentation for medical image analysis. Existing approaches either applied multi-view planar (2D) networks or directly used volumetric (3D) networks for this purpose, but both of them are not ideal: 2D networks cannot capture 3D contexts effectively, and 3D networks are both memory-consuming and less stable arguably due to the lack of pre-trained models.

In this paper, we bridge the gap between 2D and 3D using a novel approach named Elastic Boundary Projection (EBP). The key observation is that, although the object is a 3D volume, what we really need in segmentation is to find its boundary which is a 2D surface. Therefore, we place a number of pivot points in the 3D space, and for each pivot, we determine its distance to the object boundary along a dense set of directions. This creates an elastic shell around each pivot which is initialized as a perfect sphere. We train a 2D deep network to determine whether each ending point falls within the object, and gradually adjust the shell so that it gradually converges to the actual shape of the boundary and thus achieves the goal of segmentation. EBP allows boundary-based segmentation without cutting a 3D volume into slices or patches, which stands out from conventional 2D and 3D approaches. EBP achieves promising accuracy in abdominal organ segmentation. Our code will be released on https://github.com/twni2016/EBP.

1. Introduction

Medical image analysis (MedIA), in particular 3D organ segmentation, is an important prerequisite of computer-assisted diagnosis (CAD), which implies a broad range of applications. Recent years, with the blooming development of deep learning, convolutional neural networks have been widely applied to this area [23, 22], which largely boosts the performance of conventional segmentation approaches based on handcrafted features [17, 18], and even surpasses human-level accuracy in many organs and soft tissues.

	2D-Net [23, 31]	3D-Net [22, 33]	AH-Net [20]	**EBP** (ours)
Pure 2D network?	✓			✓
Pure 3D network?		✓		
Working on 3D data?		✓	✓	✓
3D data not cropped?				✓
3D data not rescaled?				✓
Can be pre-trained?	✓		✓	✓
Segmentation method	**R**	**R**	**R**	**B**

Table 1. A comparison between EBP and previous approaches in network dimensionality, data dimensionality, the ways of preprocessing data, network weights, and segmentation methodology. Due to space limit, we do not cite all related work here – see Section 2 for details. **R** and **B** in the last row stand for *region-based* and *boundary-based* approaches, respectively.

Existing deep neural networks for medical image segmentation can be categorized into two types, differing from each other in the dimensionality of the processed object. The first type cuts the 3D volume into 2D slices, and trains a 2D network to deal with each slice either individually [31] or sequentially [6]. The second one instead trains a 3D network to deal with volumetric data directly [22, 20]. Although the latter was believed to have potentially a stronger ability to consider 3D contextual information, it suffers from two weaknesses: (1) the lack of pre-trained models makes the training process unstable and the parameters tuned in one organ less transferrable to others, and (2) the large memory consumption makes it difficult to receive the entire volume as input, yet fusing patch-wise prediction into the final volume remains non-trivial yet tricky.

In this paper, we present a novel approach to bridge the gap between 2D networks and 3D segmentation. Our idea comes from the observation that an organ is often single-connected and locally smooth, so, instead of performing voxel-wise prediction, segmentation can be done by finding its boundary which is actually a 2D surface. Our approach is named Elastic Boundary Projection (**EBP**), which uses the spherical coordinate system to project the irregular boundary into a rectangle, on which 2D networks can be applied. EBP starts with a pivot point within or without the target organ and a elastic shell around it. This shell, parameterized by the radius along different directions, is initialized as a perfect sphere (all radii are the same). The goal is to adjust the shell so that it eventually converges to the boundary of the target organ, for which we train a 2D network to predict whether each ending point lies inside or outside the organ, and correspondingly increase or decrease the radius at that direction. This is an iterative process, which terminates when the change of the shell is sufficiently small. In practice, we place a number of pivots in the 3D space, and summarize all the converged shells for outlier removal and 3D reconstruction.

Table 1 shows a comparison between EBP and previous 2D and 3D approaches. EBP enjoys three-fold advantages. First, EBP allows using a 2D network to perform volumetric segmentation, which absorbs both training stability and contextual information. Second, with small memory usage, EBP processes a 3D object entirely without cutting it into slices or patches, and thus prevents the trouble in fusing predictions. Third, EBP can sample abundant training cases by placing a number of pivots, which is especially useful in the scenarios of limited data annotations. We evaluate EBP in segmenting several organs in abdominal CT scans, and demonstrate its promising performance.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work, and Section 3 describes the proposed EBP algorithm. After experiments are shown in Section 4, we draw our conclusions in Section 5.

2. Related Work

Computer aided diagnosis (CAD) is a research area which aims at helping human doctors in clinics. Currently, a lot of CAD approaches start from medical image analysis to obtain accurate descriptions of the scanned organs, soft tissues, *etc.*. One of the most popular topics in this area is object segmentation, *i.e.*, determining which voxels belong to the target in 3D data, such as abdominal CT scans studied in this paper. Recently, the success of deep convolutional neural networks for image classification [15, 28, 11, 13] has been transferred to object segmentation in both natural images [27, 7] and medical images [23, 22].

One of the most significant differences between natural and medical images lies in data dimensionality: natural images are planar (2D) while medical data such as CT and MRI scans are volumetric (3D). To deal with it, researchers proposed two major pipelines. The first one cut each 3D volume into 2D slices, and trained 2D networks to process each of them individually [23]. Such methods often suffer from missing 3D contextual information, for which various techniques were adopted, such as using 2.5D data (stacking a few 2D images as different input channels) [24, 25], training deep networks from different viewpoints and fusing multi-view information at the final stage [32, 30, 31], and applying a recurrent network to process sequential data [6, 4]. The second one instead trained a 3D network to deal with volumetric data [8, 22]. These approaches, while being able to see more information, often require much larger memory consumption, and so most existing methods worked on small patches [10, 33], which left a final stage to fuse the output of all patches. In addition, unlike 2D networks that can borrow pre-trained models from natural image datasets [9], 3D networks were often trained from scratch, which often led to unstable convergence properties [29]. One possible solution is to decompose each 3D convolution into a 2D-followed-by-1D convolution [20]. A discussion on 2D vs. 3D models for medical image segmentation is available in [16].

Prior to the deep learning era, planar image segmentation algorithms were often designed to detect the boundary of a 2D object [12, 1, 3, 26, 19]. Although these approaches have been significantly outperformed by deep neural networks in the area of medical image analysis [17, 18], we borrow the idea of finding the 2D boundary instead of the 3D volume and design our approach.

3. Elastic Boundary Projection

3.1. Problem, Existing Methods and Drawbacks

The problem we are interested in is to segment an organ from abdominal CT scans. Let an input image be $\mathbf{U}$, a 3D volume with $H_x \times H_y \times H_z$ voxels, and each voxel $U(x,y,z)$ indicates the intensity at the specified position measured by the Haunsfield unit (HU). The label $\mathbf{V}$ shares the same dimension with $\mathbf{U}$, and $V(x,y,z)$ indicates the class annotation of $U(x,y,z)$. Without loss of generality, we assume that $V(x,y,z) \in \{0,1\}$ where 1 indicates the target organ and 0 the background. Suppose our model predicts a volume $\mathbf{W}$, and $\mathcal{V} = \{(x,y,z) \mid V(x,y,z) = 1\}$ and $\mathcal{W} = \{(x,y,z) \mid W(x,y,z) = 1\}$ are the *foreground* voxels in ground-truth and prediction, respectively, we can compute segmentation accuracy using the Dice-Sørensen coefficient (DSC): $\mathrm{DSC}(\mathcal{V},\mathcal{W}) = \frac{2\times|\mathcal{V}\cap\mathcal{W}|}{|\mathcal{V}|+|\mathcal{W}|}$, which has a range of $[0,1]$ with 1 implying a perfect prediction.

Let us denote the goal as $\mathbf{W} = \mathbf{f}(\mathbf{U};\boldsymbol{\theta})$. Thus, there are two typical ways of designing $\mathbf{f}(\cdot;\boldsymbol{\theta})$. The first one trains a 3D model to deal with volumetric data directly [8, 22], while the second one works by cutting the 3D volume into slices and using 2D networks for segmentation [24, 31]. Both 2D and 3D approaches have their advantages and disadvantages. We appreciate the ability of 3D networks to take volumetric cues into consideration (radiologists also exploit 3D information to make decisions), however, 3D networks are sometimes less stable, arguably because we need to train all weights from scratch, while the 2D networks can be initialized using pre-trained models from natural images (*e.g.*, RSTN [31] borrowed FCN [21] as ini-

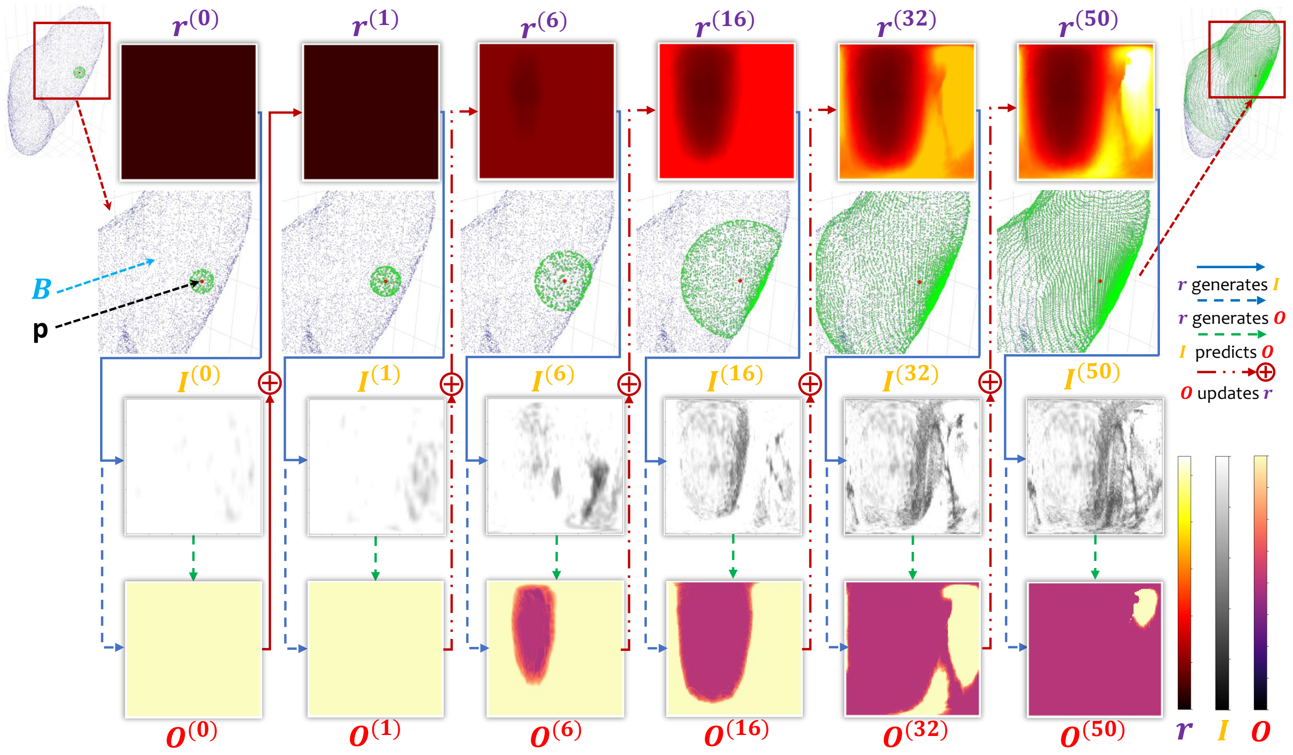

Figure 1. The overall flowchart of EBP (best viewed in color). We show the elastic shell after some specific numbers of iterations (green voxels in the second row) generated by a pivot $\mathbf{p}$ (the red center voxel in the second row) within a boundary $\mathcal{B}$ of the organ (blue voxels in 2nd row). The data generation process starts from a perfect sphere initialized by $r^{(0)}$, and then we obtain the $\left(\mathbf{I}^{(t)}, \mathbf{O}^{(t)}\right)$ pairs (the third and fourth row) by $r^{(t)}$ in the training stage. In the testing stage, $\mathbf{O}^{(t)}$ is predicted by our model $\mathbb{M}$ given $\mathbf{I}^{(t)}$. After that, one iteration is completed by the adjustment of $r^{(t)}$ to $r^{(t+1)}$ by the addition of $\mathbf{O}^{(t)}$. Finally, the elastic shell converges to $\mathcal{B}$.

tialization). On the other hand, processing volumetric data (*e.g.*, 3D convolution) often requires heavier computation in both training and testing. We aim at designing an algorithm which takes both benefits of 2D and 3D approaches.

3.2. EBP: the Overall Framework

Our algorithm is named Elastic Boundary Projection (EBP). As the name shows, our core idea is to predict the boundary of an organ instead of every pixel in it.

Consider a binary volume $\mathbf{V}$ with $\mathcal{V}$ indicating the foreground voxel set. We define its *boundary* $\mathcal{B} = \partial\mathbf{V}$ as a set of (continuous) coordinates that are located between the foreground and background voxels[1]. Since $\mathcal{B}$ is a 2D surface, we can parameterize it using a rectangle, and then apply a 2D deep network to solve it. We first define a set of *pivots* $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ which are randomly sampled from the region-of-interest (ROI), *e.g.*, the 3D bounding-box of the object. Then, in the spherical coordinate system, we define a fixed set of *directions* $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_M\}$, in which each $\mathbf{d}_m$ is a unit vector $(\hat{x}_m, \hat{y}_m, \hat{z}_m)$, *i.e.*, $\hat{x}_m^2 + \hat{y}_m^2 + \hat{z}_m^2 = 1$, for $m = 1, 2, \ldots, M$. For each pair of $\mathbf{p}_n$ and $\mathbf{d}_m$, there is a *radius* $r_{n,m}$ indicating how far the boundary is along this direction, *i.e.*, $\mathbf{e}_{n,m} = \mathbf{p}_n + r_{n,m} \cdot \mathbf{d}_m \in \mathcal{B}$[2]. When $\mathcal{B}$ is not convex, it is possible that a single pivot cannot see the entire boundary, so we need multiple pivots to provide complementary information. Provided a sufficiently large number of pivots as well as a densely distributed direction set $\mathcal{D}$, we can approximate the boundary $\mathcal{B}$ and thus recover the volume $\mathcal{V}$ which achieves the goal of segmentation.

Therefore, volumetric segmentation reduces to the following problem: given a pivot $\mathbf{p}_n$ and a set of directions $\mathcal{D}$, determine all $r_{n,m}$ so that $\mathbf{e}_{n,m} = \mathbf{p}_n + r_{n,m} \cdot \mathbf{d}_m \in \mathcal{B}$. This task is difficult to solve directly, which motivates us to consider the following counter problem: given $\mathbf{p}_n$, $\mathcal{D}$ and a group of $r_{n,m}$ values, determine whether these values correctly describe the boundary, *i.e.*, whether each $\mathbf{e}_{n,m}$

[1]The actual definition used in implementation is slightly different – see Section 3.4 for details.

[2]If $\mathbf{p}$ is located outside the boundary, there may exist some directions that the ray $\mathbf{e}_{n,m}(r) = \mathbf{p}_n + r \cdot \mathbf{d}_m$ does not intersect with $\mathcal{B}$. In this case, we define $r_{n,m} = 0$, *i.e.*, along these directions, the boundary collapses to the pivot itself. In all other situations (including $\mathbf{p}$ is located within the boundary), there may be more than one r_m's that satisfy this condition, in which cases we take the maximal r_m. When there is a sufficient number of pivots, the algorithm often reconstructs the entire boundary as expected. See Sections 3.4 and 3.5 for implementation details.

falls on the boundary. We train a model $\mathbb{M} : \mathbf{O} = \mathbf{f}(\mathbf{I}; \boldsymbol{\theta})$ to achieve this goal. Here, the input is a generated image $\mathbf{I}_n \equiv \{U(\mathbf{p}_n + r_{n,m} \cdot \mathbf{d}_m)\}_{m=1}^{M} = \{U(\mathbf{e}_{n,m})\}_{m=1}^{M}$, where $U(\mathbf{e}_{n,m})$ is the intensity value of $\mathbf{U}$ at position $\mathbf{e}_{n,m}$, interpolated by neighboring voxels if necessary. Note that $\mathbf{I}$ appears in a 2D rectangle. The output is a map $\mathbf{O}$ of the same size, with each value o_m indicating whether $\mathbf{e}_{n,m}$ is located within, and how far it is from the boundary.

The overall flowchart of EBP is illustrated in Figure 1. In the training stage, we sample $\mathcal{P}$ and generate $(\mathbf{I}, \mathbf{O})$ pairs to optimize $\boldsymbol{\theta}$. In the testing stage, we randomly sample $\mathcal{P}'$ and initialize all $r'_{n,m}$'s with a constant value, and use the trained model to iterate on each $\mathbf{p}'_n$ until convergence, *i.e.*, all entries in $\mathbf{O}'$ are close to 0 (as we shall see later, convergence is required because one-time prediction can be inaccurate). Finally, we perform 3D reconstruction using all $\mathbf{e}'_{n,m}$'s to recover the volume $\mathbf{V}$. We will elaborate the details in the following subsections.

3.3. Data Preparation: Distance to Boundary

In the preparation stage, based on a binary annotation $\mathbf{V}$, we aim at defining a relabeled matrix $\mathbf{C}$, with its each entry $C(x, y, z)$ storing the *signed distance* between each integer coordinate (x, y, z) and $\partial\mathbf{V}$. The sign of $C(x, y, z)$ indicates whether (x, y, z) is located within the boundary (positive: inside; negative: outside; 0: on), and the absolute value indicates the distance between this point and the boundary (a point set). We follow the convention to define

$$|C(x, y, z)| = \min_{(x',y',z') \in \partial\mathbf{V}} \mathrm{Dist}[(x, y, z), (x', y', z')], \tag{1}$$

where we use the ℓ_2-distance $\mathrm{Dist}[(x, y, z), (x', y', z')] = \left(|x - x'|^2 + |y - y'|^2 + |z - z'|^2\right)^{1/2}$ (the Euclidean distance) while a generalized ℓ_p-distance can also be used. We apply the KD-tree algorithm for fast search. If other distances are used, *e.g.*, ℓ_1-distance, we can apply other efficient algorithms, *e.g.*, floodfill, for constructing matrix $\mathbf{C}$. The overall computational cost is $O(N_0 \log N_0^{\circ})$, where $N_0 = H_x H_y H_z$ is the number of voxels and $N_0^{\circ} = |\partial\mathbf{V}|$ is the size of the boundary set[3].

After $\mathbf{C}$ is computed, we multiply $C(x, y, z)$ by -1 for all background voxels, so that the sign of $C(x, y, z)$ distinguishes inner voxels from outer voxels. In the following parts, (x, y, z) can be a floating point coordinate, in which case we use trilinear interpolation to obtain $C(x, y, z)$.

3.4. Training: Data Generation and Optimization

To optimize the model $\mathbb{M}$, we need a set of training pairs $\{(\mathbf{I}, \mathbf{O})\}$. To maximally reduce the gap between training and testing data distributions, we simulate the iteration process in the training stage and sample data on the way.

We first define the direction set $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_M\}$. We use the spherical coordinate system, which means that each direction has an azimuth angle $\alpha_{m_1} \in [0, 2\pi)$ and a polar angle $\varphi_{m_2} \in [-\pi/2, \pi/2]$. To organize these M directions into a rectangle, we represent $\mathcal{D}$ as the Cartesian product of an azimuth angle set of M^{a} elements and a polar angle set of M^{p} elements where $M^{\mathrm{a}} \times M^{\mathrm{p}} = M$. The M^{a} azimuth angles are uniformly distributed, *i.e.*, $\alpha_{m_1} = 2m_1\pi/M^{\mathrm{a}}$, but the M^{p} polar angles have a denser distribution near the equator, *i.e.*, $\varphi_{m_2} = \cos^{-1}(2m_2/(M^{\mathrm{p}} + 1) - 1) - \pi/2$, so that the M unit vectors are approximately uniformly distributed over the sphere. Thus, for each m, we can find the corresponing m_1 and m_2, and the unit direction vector $(\hat{x}_m, \hat{y}_m, \hat{z}_m)$ satisfies $\hat{x}_m = \cos\alpha_{m_1} \cos\varphi_{m_2}$, $\hat{y}_m = \sin\alpha_{m_1} \cos\varphi_{m_2}$ and $\hat{z}_m = \sin\varphi_{m_2}$, respectively. $\mathbf{d}_m = (\hat{x}_m, \hat{y}_m, \hat{z}_m)$. In practice, we fix $M^{\mathrm{a}} = M^{\mathrm{p}} = 120$ which is a tradeoff between sampling density (closely related to accuracy) and computational costs.

We then sample a set of pivots $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$. At each $\mathbf{p}_n$, we construct a unit sphere with a radius of R_0, *i.e.*, $r_{n,m}^{(0)} = R_0$ for all m, where the superscript 0 indicates the number of undergone iterations. After the t-th iteration, the coordinate of each ending point is computed by:

$$\mathbf{e}_{n,m}^{(t)} = \mathbf{p}_n + r_{n,m}^{(t)} \cdot \mathbf{d}_m. \tag{2}$$

Using the coordinates of all m, we look up the ground-truth to obtain an input-output data pair:

$$\mathbf{I}_{n,m}^{(t)} = U\left(\mathbf{e}_{n,m}^{(t)}\right), \quad \mathbf{O}_{n,m}^{(t)} = C\left(\mathbf{e}_{n,m}^{(t)}\right), \tag{3}$$

and then adjust $r_{n,m}^{(t)}$ accordingly[6]:

$$r_{n,m}^{(t+1)} = \max\left\{r_{n,m}^{(t)} + C\left(\mathbf{e}_{n,m}^{(t)}\right), 0\right\}, \tag{4}$$

until convergence is achieved and thus all ending points fall on the boundary or collapse to $\mathbf{p}_n$ itself[7].

[3]Here are some technical details. The KD-tree is built on the set of *boundary voxels*, *i.e.*, the integer coordinates with at least one (out of six) neighborhood voxels having a different label (foreground vs. background) from itself. There are in average $N_0^{\circ} = 50{,}000$ such voxels for each case, and performing N_0 individual searches on this KD-tree takes around 20 minutes. To accelerate, we limit $|C(x, y, z)| \leqslant \tau$ which implies that all coordinates with a sufficiently large distance are truncated (this is actually more reasonable for training – see the next subsection). We filter all pixels with an $\ell_{-\infty}$-distance not smaller than τ[4], which runs very fast[5] and typically reduces the number of searches to less than 1% of N_0. Thus, data preparation takes less than 1 minute for each case.

[6]Eqn 4 is not strictly correct, because $\mathbf{d}_m$ is not guaranteed to be the fastest direction along which $\mathbf{e}_{n,m}^{(t)}$ goes to the nearest boundary. However, since $C\left(\mathbf{e}_{n,m}^{(t)}\right)$ is the shortest distance to the boundary, Eqn (4) does not change the inner-outer property of $\mathbf{e}_{n,m}^{(t)}$.

[7]If $\mathbf{p}_n$ is located within the boundary, then all ending points will eventually converge onto the boundary. Otherwise, along all directions with $\mathbf{e}_{n,m}^{(0)}$ being outside the boundary, $r_{n,m}^{(t)}$ will be gradually reduced to 0 and thus the ending point collapses to $\mathbf{p}_n$ itself. These collapsed ending points will not be considered in 3D reconstruction (see Section 3.6).

When the 3D target is non-convex, there is a possibility that a ray $\mathbf{e}_{n,m}(r) = \mathbf{p}_n + r \cdot \mathbf{d}_m$ has more than one intersections with the boundary. In this case, the algorithm will converge to the one that is closest to the initial sphere. We do not treat this issue specially in both training and testing, because we assume a good boundary can be recovered if (i) most ending points are close to the boundary and (ii) pivots are sufficiently dense.

Here we make an assumption: by looking at the projected image at the boundary, it is not accurate to predict that the radius along any direction should be increased or decreased by a distance larger than τ (we use $\tau = 2$ in experiments). So, we constrain $C(x, y, z) \in [-\tau, \tau]$. This brings three-fold benefits. First, the data generation process becomes much faster (see the previous subsection); second, iteration allows to generate more training data; third and the most important, this makes prediction easier and more reasonable, as we can only expect accurate prediction within a small neighborhood of the boundary.

After the training set is constructed, we optimize $\mathbb{M} : \mathbf{O} = \mathbf{f}(\mathbf{I}; \boldsymbol{\theta})$ with regular methods, *e.g.*, stochastic gradient descent is used in this paper. Please see section 4.1 for the details of $\mathbb{M}$. As a side comment, our approach can generate abundant training data by increasing N and thus the sampling density of pivots, which is especially useful when the labeled training set is very small.

3.5. Testing: Iteration and Inference

The testing stage is mostly similar to the training stage, which starts with a set of randomly placed pivots and a unit sphere around each of them. We fix the parameters $\boldsymbol{\theta}$ and iterate until convergence or the maximal number of rounds T is reached (unlike training in which ground-truth is provided, iteration may not converge in testing). After that, all ending points of all pivots, except for those collapsed to the corresponding pivot, are collected and fed into the next stage, *i.e.*, 3D reconstruction. The following techniques are applied to improve testing accuracy.

First, the input image $\mathbf{I}_n^{(t)}$ at each training/testing round only contains intensity values at the current shell defined by $\left\{\mathbf{e}_{n,m}^{(t)}\right\}$. However, such information is often insufficient to accurately predict $\mathbf{O}_n^{(t)}$, so we complement it by adding more *channels* to $\mathbf{I}_n^{(t)}$. The l-th channel is defined by M radius values $\left\{s_{n,l,m}^{(t)}\right\}$. There are two types of channels, with L^{A} of them being used to sample the boundary and L^{B} of them to sample the inner volume:

$$\begin{aligned} s_{n,l^{\mathrm{A}},m}^{(t)} &= r_{n,m}^{(t)} + l^{\mathrm{A}} - \left(L^{\mathrm{A}} + 1\right)/2, \\ s_{n,l^{\mathrm{B}},m}^{(t)\prime} &= \frac{l^{\mathrm{B}}}{L^{\mathrm{B}} + 1}\left[r_{n,m}^{(t)} - \left(L^{\mathrm{A}} + 1\right)/2\right]. \end{aligned} \tag{5}$$

When $L^{\mathrm{A}} = 1$ and $L^{\mathrm{B}} = 0$, it degenerates to using one single slice at the boundary. With relatively large L^{A} and L^{B} (*e.g.*, $L^{\mathrm{A}} = L^{\mathrm{B}} = 5$ in our experiments), we benefit from seeing more contexts which is similar to volumetric segmentation but the network is still 2D. The number of channels in $\mathbf{O}$ remains to be 1 regardless of L^{A} and L^{B}.

Second, we make use of the spatial consistency of distance prediction to improve accuracy. When the radius values at the current iteration $\left\{r_{n,m}^{(t)}\right\}$ are provided, we can randomly sample M numbers $\varepsilon_m \sim \mathcal{N}\left(0, \sigma^2\right)$ where σ is small, add them to $\left\{r_{n,m}^{(t)}\right\}$, and feed the noisy input to $\mathbb{M}$. By spatial consistency we mean the following approximation is always satisfied for each direction m:

$$C\left(\mathbf{e}_{n,m}^{(t)} + \varepsilon_m \cdot \mathbf{d}_m\right) = C\left(\mathbf{e}_{n,m}^{(t)}\right) + \varepsilon_m \cdot \cos\beta\left(\mathbf{d}_m, \mathbf{e}_{n,m}^{(t)}\right), \tag{6}$$

where $\beta\left(\mathbf{d}_m, \mathbf{e}_{n,m}^{(t)}\right)$ is the angle between $\mathbf{d}_m$ and the normal direction at $\mathbf{e}_{n,m}^{(t)}$. Although this angle is often difficult to compute, we can take the left-hand side of Eqn (6) as a linear function of ε_m and estimate its value at 0 using multiple samples of ε_m. This technique behaves like data augmentation and improves the stability of testing.

Third, we shrink the gap between training and testing data distributions. Note that in the training stage, all $r_{n,m}^{(t)}$ values are generated using ground-truth, while in the testing stage, they are accumulated by network predictions. Therefore, inaccuracy may accumulate with iteration if the model is never trained on such "real" data. To alleviate this issue, in the training stage, we gradually replace the added term in Eqn (4) with prediction, following the idea of curriculum learning [2]. In Figure 1, we show that this strategy indeed improves accuracy in validation.

Last but not least, we note that most false positives in the testing stage are generated by outer pivots, especially those pivots located within another organ with similar physical properties. In this case, the shell may converge to an unexpected boundary which harms segmentation. To alleviate this issue, we introduce an extra stage to determine which pivots are located within the target organ. This is achieved by constructing a graph with all pivots being nodes and edges being connected between neighboring pivots. The weight of each edge is the the intersection-over-union (IOU) rate between the two volumes defined by the elastic shells. To this end, so we randomly sample several thousand points in the region-of-interest (ROI) and compute whether they fall within each of the N shells, based on which we can estimate the IOU of any pivot pairs. Then, we find the minimum cut which partitions the entire graph to two parts, and the inner part is considered the set of inner pivots. Only the ending points produced by inner pivots are considered in 3D reconstruction.

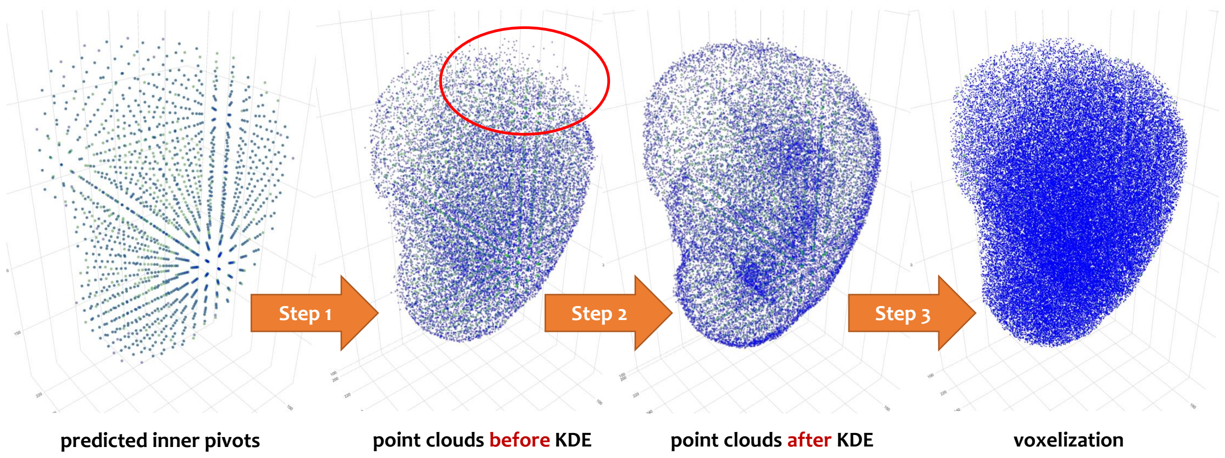

Figure 2. An example of 3D reconstruction (best viewed in color). We start with all pivots (green and blue points indicate ground-truth and predicted inner pivots, respectively) predicted to be located inside the target. In Step 1, all converged ending points generated by these pivots form the point clouds. In Step 2, a kernel density estimator (KDE) is applied to remove outliers (marked in a red oval in the second figure). In Step 3, we adopt a graphics algorithm for 3D reconstruction and finally we voxelize the point cloud.

3.6. 3D Reconstruction

The final step is to reconstruct the surface of the 3D volume based on all ending points. Note that there always exist many false positives (*i.e.*, predicted ending points that do not fall on the actual boundary), so we adopt kernel density estimation (KDE) to remove them, based on the assumption that with a sufficient number of pivots, the density of ending points around the boundary is much larger than that in other regions. We use the Epanechnikov kernel with a bandwidth of 1, and preserve all integer coordinates with a log-likelihood not smaller than -14.

Finally, we apply a basic graphics framework to accomplish this goal, which works as follows. We first use the Delaunay triangulation to build the mesh structure upon the survived ending points, and then remove improper tetrahedrons with a circumradius larger than α. After we obtain the alpha shape, we use the subdivide algorithm to voxelize it into volumes with hole filling. Finally, we apply surface thinning to the volumes by 3 slices. This guarantees a closed boundary, filling which obtains final segmentation. We illustrate an example of 3D reconstruction in Figure 2.

3.7. Discussions and Relationship to Prior Work

The core contribution of EBP is to provide a 2D-based approach for 3D segmentation. To the best of our knowledge, this idea was not studied in the deep learning literature. Conventional segmentation approaches such as GraphCut [3] and GrabCut [26] converted 2D segmentation to find the minimal cut, a 1D contour that minimizes an objective function, which shared a similar idea with us. Instead of manually defining the loss function by using voxel-wise or patch-wise difference, EBP directly measures the loss with a guess and iteratively approaches the correct boundary. This is related to the *active contour* methods [14, 5].

In the perspective of dimension reduction, EBP adds a different solution to a large corpus of 2D segmentation approaches [23, 24, 25, 32, 31] which cut 3D volumes into 2D slices without considering image semantics. Our solution enjoys the ability of extracting abundant training data, *i.e.*, we can sample from an infinite number of pivots (no need to have integer coordinates). This makes EBP stand out especially in the scenarios of fewer training data (see experiments). Also, compared to pure 3D approaches [8, 22], we provide a more efficient way of sampling voxels which reduces computational overheads as well as the number of parameters, and thus the risk of over-fitting.

4. Experiments

4.1. Datasets, Evaluation and Details

We evaluate EBP in a dataset with 48 high-resolution CT scans. The width and height of each volume are both 512, and the number of slices along the *axial* axis varies from 400 to 1,100. These data were collected from some potential renal donors, and annotated by four expert radiologists in our team. Four abdominal organs were labeled, including *left kidney*, *right kidney* and *spleen*. Around 1 hour is required for each scan. All annotations were later verified by an experienced board certified Abdominal Radiologist. We randomly choose half of these volumes for training, and use the remaining half for testing. The data split is identical for different organs. We compute DSC for each case individually, *i.e.*, $\mathrm{DSC}(\mathcal{V},\mathcal{W}) = \frac{2\times|\mathcal{V}\cap\mathcal{W}|}{|\mathcal{V}|+|\mathcal{W}|}$ where $\mathcal{V}$ and $\mathcal{W}$ are ground-truth and prediction, respectively.

For the second dataset, we refer to the *spleen* subset in the Medical Segmentation Decathlon (MSD) dataset (website: http://medicaldecathlon.com/). This is a public dataset with 41 cases, in which we randomly choose 21 for training and the remaining 20 are used for testing. This dataset has quite a different property from ours, as the spatial resolution varies a lot. Although the width and height are still both 512, the length can vary from 31 to 168. DSC is also used for accuracy computation.

Two recenty published baselines named RSTN [31] and VNet [22] are used for comparison. RSTN is a 2D-based network, which uses a coarse-to-fine pipeline with a saliency transformation module. We directly follow the implementation by the authors. VNet is a 3D-based network, which randomly crops into $128 \times 128 \times 64$ patches from the original patch for training, and uses a 3D sliding window followed by score average in the testing stage. Though RSTN does not require a 3D bounding-box (ROI) while EBP and VNet do, this is considered fair because a 3D bounding-box is relatively easy to obtain. In addition, we also evaluate RSTN with 3D bounding-box, and found very little improvement compared to the original RSTN.

The model $\mathbb{M} : \mathbf{O} = \mathbf{f}(\mathbf{I};\boldsymbol{\theta})$ of EBP is instantiated as a 2D neural network based on UNet [23]. The input image $\mathbf{I}$ has a resolution of $M = M^{\mathrm{a}} \times M^{\mathrm{p}} = 120 \times 120$. We set $L^{\mathrm{A}} = L^{\mathrm{B}} = 5$, and append 3 channels of $\mathbf{d}$ for both parts (thus each part has 8 channels, and group convolution is applied). Our network has 3 down-sampling and 3 up-sampling blocks, each of which has three consecutive 2-group dilated (rate is 2) convolutions. There are also short (intra-block) and long (inter-block) residual connections. The output $\mathbf{O}$ is a one-channel signed distance matrix.

4.2. Quantitative Results

Results are summarized in Table 2. In all these organs, EBP achieves comparable segmentation accuracy with RSTN, and usually significantly outperforms VNet.

On our own data, EBP works slightly worse than RSTN, but on the *spleen* set, the worst case reported by RSTN has a much lower DSC (78.75%) than that of EBP (89.67%). After diagnosis, we find that RSTN fails to detect a part this organ in a few continuous 2D slices, but EBP, by detecting the boundary, successfully recovers this case. This suggests that in many cases, EBP and RSTN can provide supplementary information to organ segmentation. Moreover, on the MSD spleen dataset, a challenging public dataset, EBP outperforms RSTN by more than 2%. In addition, (i) the worst case in MSD *spleen* reported by EBP is 77.07%, much higher than 48.45% reported by RSTN; (ii) all standard deviations reported by RSTN are significantly larger. Both the above suggest that EBP enjoys higher stability.

We can observe that VNet often suffers even lower stability in terms of both standard deviation and worst accuracy. In addition, the training process is not guaranteed to converge to a good model, *e.g.*, in *right kidney* of our own dataset, we trained two VNet models – one of them, as shown in Table 2, is slightly worse than both RSTN and EBP; while the other, reports even worse results: $86.30 \pm 6.50\%$ average, 95.32% max and 73.66% min DSCs, respectively. Similar phenomena, which are mainly due to the difficulty of optimizing a 3D-based network, were also observed in [8, 20, 30].

We also observe the impact of spatial resolution in the MSD *spleen* dataset. This dataset has a relatively low spatial resolution (*i.e.*, 31–168 voxels along the long axis), which raises extra difficulties to VNet (it requires $128 \times 128 \times 64$ patches to be sampled). To deal with this issue, we normalize all volumes so as to increase the number of slices along the long axis. The results of VNet shown in Table 2 are computed in this normalized dataset (in DSC computation, all volumes are normalized back to the original resolution for fair comparison), while both RSTN and EBP directly work on the original non-normalized dataset. In addition, VNet reports an $71.07 \pm 36.27\%$ average DSC on the non-normalized dataset, with a few cases suffering severe false negatives. This implies that VNet heavily depends on data homogeneity, while EBP does not.

A complicated organ with irregular geometric shape and blurring boundary, *pancreas*, is also investigated by these approaches. We find that voxel-wise/region-based methods such as RSTN perform well on it, with over 80% accuracy on our own dataset. However, EBP predicts many false positives and thus only has about 60% accuracy on the same setting. After careful diagnosis, we figure out that for pancreas segmentation, EBP tends to mistake some other organs around pancreas within 3D bounding-box, such as *small bowel*, *inferior vena cava* and *duodenum*, for pancreas. There are several reasons accounting for the weakness in pancreas segmentation. **First**, those surrounding organs are intertwined with pancreas and some parts of their boundaries coincide within several voxels. **Second**, their boundaries look similar to that of pancreas from the perspective of intensity distribution, which adds difficulty to EBP. **Third**, 3D reconstruction of EBP is inaccurate for organs with irregular shape, for the difficulty in choosing the hyperparameter α to trim the convex hull of the predicted point cloud to the ground-truth irregular shape.

4.3. How Does EBP Find the Boundary?

Now, we discuss on how EBP finds the boundary. We study two aspects, *i.e.*, convergence and consistency, on one case of medium difficulty in the subset of *right kidney*.

We start with investigating **convergence**, by which we refer to whether each pivot $\mathbf{p}_n$, after a sufficient number of iterations, can converge to a boundary. We use the ℓ_1-norm of $\mathbf{O}'_{n,m}$ to measure convergence, the output of which indicates the amount of revision along the radius. With its value reaching at a low level (positive but smaller than

Approach	*left kidney*			*right kidney*			*spleen*			**MSD** *spleen*		
	Average	Max	Min	Average	Max	Min	Average	Max	Min	Average	Max	Min
RSTN [31]	94.50 ± 2.66	97.69	93.64	96.09 ± 2.21	98.18	87.35	94.63 ± 4.21	97.38	78.75	89.70 ± 12.60	97.25	48.45
VNet [22]	91.95 ± 4.63	95.23	71.40	92.97 ± 3.67	97.48	80.51	92.68 ± 3.25	96.75	83.18	92.94 ± 3.58	97.35	81.96
EBP (ours)	93.45 ± 1.62	97.28	90.88	95.26 ± 1.59	97.45	90.19	94.50 ± 2.64	96.76	89.67	92.01 ± 4.50	96.48	77.07

Table 2. Comparison of segmentation accuracy (DSC, %) on our multi-organ dataset and the *spleen* class in the MSD benchmark. Within each group, average (with standard deviation), max and min accuracies are reported.

0.5), perfect convergence is achieved. Results are shown in Figure 3. We can see that, starting from most inner pivots, the elastic shell can eventually converge to the boundary. In the figure, we show 200 iterations, but in practice, for acceleration, we only perform 10 iterations before sending all “ending points” to 3D reconstruction. This is to say, although convergence is not achieved and many ending points are not indeed located at the boundary, it is possible for 3D reconstruction algorithm to filter these outliers. This is because we have sampled a large number of pivots. Therefore, an ending point located near the boundary will be accompanied by a lot of others, while one located inside or even outside the target will be isolated. By applying kernel density estimation (KDE), we can filter out those isolated points so that 3D reconstruction is not impacted.

Next, we investigate **consistency**, for which we take some pivot pairs and compute the DSC between the converged shells centered at them. This criterion was introduced in Section 3.5 to distinguish inner pivots from outer pivots. The assumption is that the shells generated by a pair of inner pivots should have a large similarity, while those generated by an inner pivot and an outer pivot should not. To maximally make fair comparison, we take all the *boundary pivots*, defined as the inner pivots with at least one neighbor being outside. Then, we sample all pivot pairs in which at least one of them is a boundary pivot, and make statistics. For those inner-inner pivot pairs, the average DSC (73.46%) is much larger than that (51.39%) of inner-outer pivot pairs. This experiment suggests that, two neighboring pivots are more likely to agree with each other if both of them are located within the target, otherwise the chance of getting a low DSC becomes large[8].

Last, we perform an interesting experiments to further reveal how inter-pivot DSC changes with the relative position of a pivot to the boundary. Starting from an inner pivot, we keep going along a fixed direction until being outside, and on the way, we record the DSC between the elastic shells generated by every neighboring pivot pairs. Some statistics are provided in Figure 3. On all these curves, we

[8]There is a side note here. Theoretically, for an inner pivot and an outer pivot, if both elastic shells are perfectly generated, they should have a DSC of 0. However, it is not often the case, because the elastic shell of the outer pivot is also initialized as a sphere, which may intersect with the boundary. In this case, all ending points that are initially located within the target will start growing until they reach the other border of the target. Consequently, it has a non-zero DSC with some of the inner pivots.

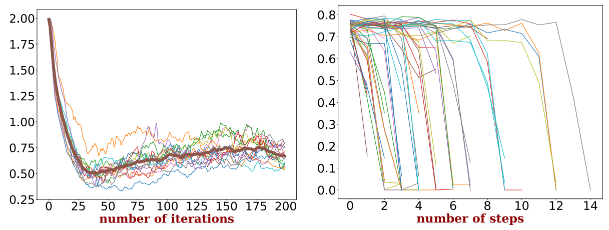

Figure 3. Left: the ℓ_1-norm of $\mathbf{O}'_{n,m}$ during the first 200 iterations. The thick curve is averaged over 15 pivots, each of which appears as a thin curve. Right: The inter-pivot DSC recorded when a pivot keeps going along a fixed direction until it goes out of the target (we do not plot the curve beyond this point).

observe a sudden drop at some place, which often indicates the moment that the pivot goes from inside to outside.

5. Conclusions

This paper presents EBP, a novel approach that trains 2D deep networks for 3D object segmentation. The core idea is to build up an elastic shell and adjust it until it converges to the actual boundary of the target. Since the shell is parameterized in the spherical coordinate system, we can apply 2D networks (low computational overhead, fewer parameters, pre-trained models, *etc.*) to deal with volumetric data (richer contextual information). Experiments are performed on several organs in abdominal CT scans, and EBP achieves comparable performance to both 2D and 3D competitors. In addition, EBP can sample sufficient training data from few annotated examples, which claims its advantage in medical image analysis.

We learn from this work that high-dimensional data often suffer redundancy (*e.g.*, not every voxel is useful in a 3D volume), and mining the discriminative part, though being challenging, often leads to a more efficient model. In the future, we will continue investigating this topic and try to cope with the weaknesses of EBP, so that it can be applied to a wider range of 3D vision problems, in particular when the object has a peculiar shape.

Acknowledgments This paper was supported by the Lustgarten Foundation for Pancreatic Cancer Research. We thank Prof. Zhouchen Lin for supporting our research. We thank Prof. Wei Shen, Dr. Yan Wang, Weichao Qiu, Zhuotun Zhu, Yuyin Zhou, Yingda Xia, Qihang Yu, Runtao Liu and Angtian Wang for instructive discussions.

References

[1] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981. 2

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48. ACM, 2009. 5

[3] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *IEEE International Conference on Computer Vision*, volume 1, pages 105–112, 2001. 2, 6

[4] Jinzheng Cai, Le Lu, Yuanpu Xie, Fuyong Xing, and Lin Yang. Pancreas segmentation in mri using graph-based decision fusion on convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 674–682. Springer, 2017. 2

[5] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001. 6

[6] Jianxu Chen, Lin Yang, Yizhe Zhang, Mark Alber, and Danny Z Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. In *Advances in Neural Information Processing Systems*, pages 3036–3044, 2016. 1, 2

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 2

[8] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016. 2, 6, 7

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[10] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, 2017. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[12] Paul VC Hough. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654. 2

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 2

[14] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 6

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 2

[16] Matthew Lai. Deep learning for medical image segmentation. *arXiv preprint arXiv:1505.02000*, 2015. 2

[17] Daw-Tung Lin, Chung-Chih Lei, and Siu-Wan Hung. Computer-aided kidney segmentation on abdominal ct images. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):59–65, 2006. 1, 2

[18] Haibin Ling, S Kevin Zhou, Yefeng Zheng, Bogdan Georgescu, Michael Suehling, and Dorin Comaniciu. Hierarchical, learning-based automatic liver segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 1, 2

[19] Jiangyu Liu, Jian Sun, and Heung-Yeung Shum. Paint selection. *ACM Transactions on Graphics*, 28(3):69, 2009. 2

[20] Siqi Liu, Daguang Xu, S Kevin Zhou, Olivier Pauly, Sasa Grbic, Thomas Mertelmeier, Julia Wicklein, Anna Jerebko, Weidong Cai, and Dorin Comaniciu. 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 851–858. Springer, 2018. 1, 2, 7

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2

[22] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision*, pages 565–571. IEEE, 2016. 1, 2, 6, 7, 8

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 1, 2, 6, 7

[24] Holger R Roth, Le Lu, Amal Farag, Hoo-Chang Shin, Jiamin Liu, Evrim B Turkbey, and Ronald M Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 556–564. Springer, 2015. 2, 6

[25] Holger R Roth, Le Lu, Amal Farag, Andrew Sohn, and Ronald M Summers. Spatial aggregation of holistically-nested networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 451–459. Springer, 2016. 2, 6

[26] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics*, volume 23, pages 309–314, 2004. 2, 6

[27] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, (4):640–651, 2017. 2

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015. 2

[29] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016. 2

[30] Yingda Xia, Lingxi Xie, Fengze Liu, Zhuotun Zhu, Elliot K Fishman, and Alan L Yuille. Bridging the gap between 2d and 3d organ segmentation with volumetric fusion net. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 445–453. Springer, 2018. 2, 7

[31] Qihang Yu, Lingxi Xie, Yan Wang, Yuyin Zhou, Elliot K Fishman, and Alan L Yuille. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8280–8289, 2018. 1, 2, 6, 7, 8

[32] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K Fishman, and Alan L Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 693–701. Springer, 2017. 2, 6

[33] Zhuotun Zhu, Yingda Xia, Wei Shen, Elliot K Fishman, and Alan L Yuille. A 3d coarse-to-fine framework for automatic pancreas segmentation. In *International Conference on 3D Vision*, 2018. 1, 2