Deep Implicit Statistical Shape Models for 3D Medical Image Delineation

Ashwin Raju, Shun Miao, Dakai Jin, Le Lu, Junzhou Huang, Adam P. Harrison

1 PAII Inc, Bethesda, MD, USA
2 University of Texas at Arlington, Arlington, TX, USA

{ashwin.raju93, jzhuang}@uta.edu
{shwinmiao, dakai.jin, tiger.lelu, adam.p.harrison}@gmail.com

Abstract

3D delineation of anatomical structures is a cardinal goal in medical imaging analysis. Prior to deep learning, statistical shape models (SSMs) that imposed anatomical constraints and produced high quality surfaces were a core technology. Today’s fully-convolutional networks (FCNs), while dominant, do not offer these capabilities. We present deep implicit statistical shape models (DISSMs), a new approach that marries the representation power of deep networks with the benefits of SSMs. DISSMs use an implicit representation to produce compact and descriptive deep surface embeddings that permit statistical models of anatomical variance. To reliably fit anatomically plausible shapes to an image, we introduce a novel rigid and non-rigid pose estimation pipeline that is modelled as a Markov decision process (MDP). Intra-dataset experiments on the task of pathological liver segmentation demonstrate that DISSMs can perform more robustly than four leading FCN models, including nnU-Net + an adversarial prior: reducing the mean Hausdorff distance (HD) by 7.5-14.3 mm and improving the worst case Dice-Sørensen coefficient (DSC) by 1.2-2.3%. More critically, cross-dataset experiments on an external and highly challenging clinical dataset demonstrate that DISSMs improve the mean DSC and HD by 2.1-5.9% and 9.9-24.5 mm, respectively, and the worst-case DSC by 5.4-7.3%. Supplemental validation on a highly challenging and low-contrast larynx dataset further demonstrate DISSM’s improvements. These improvements are over and above any benefits from representing delineations with high-quality surfaces.

Introduction

3D delineation is a fundamental task in medical imaging analysis. Currently, medical segmentation is dominated by fully-convolutional networks (FCNs) (Long, Shelhamer, and Darrell 2015), which segment each pixel or voxel in a bottom-up fashion. FCNs are well-suited to the underlying convolutional neural network (CNN) technology and are straightforward to implement using modern deep learning software. The current state has been made particularly plain by the dominance of nnU-Net (Isensee et al. 2021), i.e., nothing-new U-Net, in the medical segmentation decathlon (MSD) challenge (Simpson et al. 2019). Yet, despite their undisputed abilities, FCNs do lack important features compared to prior technology. For one, a surface-based representation is usually the desired end product, but FCNs output masks, which suffer from discretization effects (see top panel of Fig. 1). These are particularly severe when large inter-slice distances come into play. Conversion to a smoothed mesh is possible, but it introduces its own artifacts. While this is an important drawback, an arguably more critical limitation is that current FCN pipelines typically operate with no shape constraints.

Shape priors are critical for ensuring anatomically plausible delineations, but the techniques and concepts of statistical shape models (SSMs), so integral prior to deep learning (Heimann and Meinzer 2009), have fallen out of favor. Despite the incontrovertible power of FCNs, they can produce egregious mistakes, especially when presented with...
morbidity, scanners, or other scenarios not seen in training (again see top panel of Fig. 1). Because it is impossible to represent all clinical scenarios well enough in training datasets, priors can act as valuable regularizing forces. FCNs may also struggle with anatomical structures with low-contrast boundaries. Efforts have been made to incorporate anatomical priors with CNNs, but these either do not directly model shape and/or do not estimate rigid poses. Thus, they do not construct a true SSM. Even should their interoperability with CNNs not be an issue, classic SSMs also have their own disadvantages, as they mostly rely on the point distribution model (PDM) (Cootes et al. 1992), which requires determining a correspondence across shapes. Ideally, correspondences would not be required.

To fill these gaps, we introduce deep implicit statistical shape models (DISSMs), a new and deep approach to SSMs. Using the recently introduced deep implicit shape concept (Park et al. 2019; Chen and Zhang 2019; Mescheder et al. 2019), DISSMs learn a compact and rich latent space that can accurately and densely generate the signed distance functions (SDFs) of a representative sample of shapes. Importantly, correspondence between shapes is unnecessary, eliminating a major challenge with traditional SSMs. Statistics, e.g., mean shape, principle component analysis (PCA), and interpolation, can be performed directly on the latent space (see bottom panel of Fig. 1). To fit an anatomically plausible shape to a given image, DISSMs use a CNN to determine rigid and non-rigid poses, thus marrying the representation power of deep networks with anatomically-constrained surface representations. Pose estimation is modelled as a Markov decision process (MDP) to determine a trajectory of rigid and non-rigid poses, where the latter are defined as PCA loadings of the DISSM latent space. To handle the intractably large search space of poses, DISSMs make use of marginal space learning (MSL) (Zheng et al. 2008; Zheng and Comaniciu 2014) and inverted episodic training, the latter a concept we introduce. A final constrained deep level set refinement (Michalkiewicz et al. 2019) captures any fine details not represented by DISSM latent space shape. At a high level, DISSMs share many philosophies with traditional SSMs, but they modernize these concepts in a powerful deep-learning framework.

As proof of concept, we evaluate DISSM primarily on the problem of 3D pathological liver delineation from computed tomography (CT) scans, with supplemental validation on a challenging larynx segmentation task from low-contrast CT. For the former, we compare our approach to leading 2D (Harrison et al. 2017), hybrid (Li et al. 2018), 3D cascaded (Isensee et al. 2021), and adversarial learning (Yang et al. 2017) FCNs. When trained and tested on the MSD liver dataset (Simpson et al. 2019), DISSMs provide more robust delineations, improving the mean Hausdorff distance (HD) by 7.5-14.3mm. This is over and above any benefits of directly outputting a high resolution surface. More convincingly, we perform cross-dataset evaluation on an external dataset (97 volumes) that directly reflect clinical conditions (Raju et al. 2020a). DISSMs improve the mean Dice-Sørensen coefficient (DSC) and HD from 92.4% to 95.9% and from 34.1mm to 21.8mm, respectively, over the best fully FCN alternative (nnU-Net). In terms of robustness, the worst-case DSC is boosted from 88.1% to 93.4%. Commensurate improvements are also observed on the larynx dataset. These results confirm the value, once taken for granted, of incorporating anatomical priors in 3D delineation. Our contributions are thus: 1) we are the first to introduce a true deep SSM model that outputs high resolution surfaces; 2) we build a new correspondence-free, compact, and descriptive anatomical prior; 3) we present a novel pose estimation scheme that incorporates inverted episodic training and MSL; and 4) we provide a more robust solution than leading FCNs in both intra- and cross-dataset evaluations. Our code is publicly shared³.

Related Work

Anatomical Shape Priors: Because anatomical structures are highly constrained, delineations should match shape priors. SSMs were a workhorse for 3D medical segmentation and were most popularly realized as PDMs (Cootes et al. 1992), which requires determining a set of surface point correspondences across all shapes. Correspondence permitted statistical techniques, such as PCA, to model shape variability (Heimann and Meinzer 2009). However, dense 3D surface landmarks are not typically possible to define, and even when so, they are not always reliably present (Heimann and Meinzer 2009). The alternative of automatically generating correspondences is still a dense research topic (Heimann and Meinzer 2009), and all existing solutions are imperfect. While PDMs have been used with CNNs (Milletari et al. 2017; Bhododia et al. 2018), these works make no attempt at determining rigid poses, thus they are only applicable in constrained and limited setups. In general the reliance on explicit surface representations, e.g., meshes, makes it difficult to integrate traditional SSMs with CNNs. SSMs based on implicit level set representations have been explored (Cremers, Rousson, and Deriche 2007), but the statistics must be collected across a dense 3D regular grid. The most popular approach today is to impose an implicit prior (different from implicit representation) using auto-encoders (Ok-tay et al. 2017; Ravishankar et al. 2017) or generative adversarial networks (Yang et al. 2017; Raju et al. 2020b; Cai et al. 2019), but these do not permit a controllable and interpretable prior and are not invariant to rigid similarity transforms, so a true shape prior is not constructed. Recent work has also used CNNs to deform an initial mesh (Yao et al. 2019; Wickramasinghe et al. 2020) or point-cloud (Cai et al. 2019) sphere, but the process is used to augment an existing voxel-based output, otherwise the deformation cannot capture all details (Wickramasinghe et al. 2020). Besides, these works offer no statistical description of anatomical priors. Like traditional SSMs, DISSMs explicitly enforce shape constraints via a model of anatomical variability, but using a deep implicit SDF representation that is 1) highly descriptive, 2) highly compact, and 3) requires no correspondences. DISSM is the first to offer a complete “deep” SSM solution that includes rigid and non-rigid pose estimation.

³https://github.com/AshStuff/dissm
Deep Implicit Shapes: The concept of deep implicit shapes (Park et al. 2019; Chen and Zhang 2019; Mescheder et al. 2019) were introduced to represent shapes using a compact latent space and an SDF decoder. DISSMs adopt Park et al. (2019)’s auto-decoder formulation. Unlike these seminal works, DISSMs build a model of anatomical variability and propose a pose estimation strategy for 3D images.

Pose Estimation: Fitting an SSM to an image requires determining the rigid and non-rigid poses of the shape. Standard learning strategies, such as one shot regression (Stern, Ebner, and Urschler 2016; Gauriau et al. 2015) and exhaustive scanning (Zheng et al. 2015), yield sub-optimal results as the model relies on a prohibitively large number of samples (Ghesu et al. 2017) and only has one opportunity to produce pose estimations. A recent work directly regresses poses using deep implicit shapes, but these are from simulated 2D images on white backgrounds (Xu et al. 2019). DISSMs must operate in the challenging 3D medical setting, where datasets may be small and the contrast low. We formulate pose estimation as an agent-based MDP, where the agent navigates to a pose matching the present anatomical structure. This has connections to some recent registration approaches (Liao et al. 2017; Krebs et al. 2017; Ma et al. 2017). However, estimating both non-rigid and rigid poses presents an extremely large search space. To deal with this, 1) we introduce inverted episodic training and 2) we employ a deep realization of MSL (Zheng and Comaniciu 2014; Zheng et al. 2008) to incrementally learn pose parameters. The use of MSL links DISSMs to a variety of traditional SSM works pre-dating modern deep networks.

Method

Fig. 2 illustrates the DISSM framework. As the bottom panel demonstrates, a CNN encoder predicts rigid and non-rigid poses, which, along with desired coordinates, are fed into a deep implicit multilayer perceptron (MLP) shape decoder to output corresponding SDF values. The encoder searches for the best pose using an MDP combined with MSL (top panel). We first outline the deep implicit shape model, discuss pose estimation, then describe the final local surface refinement.

Implicit Shape Representation

Deep implicit shapes (Park et al. 2019; Chen and Zhang 2019; Mescheder et al. 2019) are a recent and powerful implicit shape representation. We use Park et al. (2019)’s formulation to model organ shapes using an SDF, which, given a coordinate, outputs the distance to a shape’s surface:

$$SDF(x) = s : x \in \mathbb{R}^3, s \in \mathbb{R},$$

where $s$ is negative inside the shape and positive outside of it. The iso-surface of (1), i.e., coordinates where it equals 0, corresponds to the shape surface. The insight of deep implicit shapes is that given a set of coordinate/SDF pairs in some canonical or normalized space, $\tilde{X} = \{\tilde{x}_i, s_i\}$, a deep MLP can be trained to approximate a shape’s SDF:

$$f_{\theta_s}(\tilde{x}) \approx SDF(\tilde{x}), \forall \tilde{x} \in \Omega,$$

where the tilde denotes canonical coordinates. Because it outputs smoothly varying values (2) has no resolution limitations, and, unlike meshes, it does not rely on an explicit discretization scheme. In practice the resolution of any captured details are governed by the model capacity and the set of training samples within $X$. While (2) may describe a single shape, it would not describe the anatomical variation across a set of shapes. To do this, we follow Park et al. (2019)’s elegant approach of auto-decoding latent variables. We set of K SDF samples from the same organ, $S = \{x_k\}_{k=1}^K$, and we create a corresponding set of latent vectors, $Z = \{z_k\}_{k=1}^K$. The deep MLP is modified to accept two inputs, $f_{\theta_s}(\tilde{x}, z_k)$, conditioning the output by the latent vector to specify which particular SDF is being modeled. We jointly optimize the network weights $\theta_s$ and latent vectors $Z$ to produce the best SDF approximations:

$$\arg \min_{\theta_s, Z} \sum_{k=1}^K \left( \sum_{i=1}^{|x_k|} \mathcal{L}(f_{\theta_s}(\tilde{x}_i, z_k), s_i) + \frac{1}{\sigma^2} \|z_k\|^2_2 \right), \quad (3)$$

where $\mathcal{L}$ is the L1 loss and the second term is a zero-mean Gaussian prior whose compactness is controlled by $\sigma^2$.

The implicit shape model assumes shapes are specified in a canonical space. However, segmentation labels are usually given as 3D masks. To create canonical coordinate/SDF training pairs, we first perform within-slice interpolation (Albu, Beugeling, and Laurendeau 2008) on masks to remove the most egregious of discretization artifacts. We then convert the masks to meshes using marching cubes (Lewiner et al. 2003), followed by a simplification algorithm (Forstmann 2020). Each mesh is then rigidly aligned to an arbitrarily chosen anchor mesh using coherent point drift (Myronenko and Song 2010). Similar to Park et al. (2019), SDF and coordinate values are randomly sampled from the mesh, with regions near the surface much more densely sampled. SDF values are also scaled to fit within $[-1, 1]$. Based on the anchor mesh, an affine matrix that maps between canonical and pixel coordinates can be constructed, $x = Ax$. More details can be found in the Supplementary.

Once the shape decoder is trained, any latent vector can be inputted into $f_{\theta_s}(\ldots)$ along with a set of coordinates to rasterize an SDF, which can then be rendered by extracting the iso-boundary. As the top row of Fig. 3 and bottom panel of Fig. 1 demonstrate, the latent space provides a rich description of shape variations. The mean latent vector, $\mu$, produces an anatomically valid shape. A PCA can capture meaningful variation, e.g., the first basis corresponds to stretching and flattening while the second controls the prominence of lobe protuberances. Interpolating between latent vectors produces reasonable shapes (bottom row of Fig. 3).

Pose Estimation

The next major step is use the compact and rich DISSM shape space to delineate an object boundary given an image, $I$. We assume a dataset of labelled images is available, allowing for the generation of coordinate/SDF pairs: $D = \{d_k\}_{k=1}^N$, where $d_k = \{x_k, s_k\}$ is specified using pixel coordinates. Note, we only assume a mask/SDF is
While (4) and (5) could work in principle, directly predicting global pose parameters in one shot is highly sensitive to any errors and we were unable to ever reach convergence. We instead interpret the encoder \( g_{\theta_E}(\cdot) \) as an “agent” that, given an initial pose, \( \omega^0 \), generates samples along a trajectory by predicting corrections to the previous pose:

\[
\Delta^\tau \omega^\tau = g_{\theta_E}(I_k, \omega^{\tau-1}) \quad \text{,} \\
\omega^\tau = \Delta^\tau \circ \omega^{\tau-1}, \quad \text{if } \tau > 0,
\]

where \( \circ \) denotes the composition of two rigid-body transformations and \( \tau \) indicates the current step in the trajectory. An observation of \( \omega^{\tau-1} \) is injected into the input of the encoder so that it is aware the previous pose to correct. To do this we rasterize the SDF corresponding to the mean shape, \( SDF_{\mu} \), once. After every step it is rigidly transformed using \( \omega^{\tau-1} \) and fed as a second input channel into \( g_{\theta_E}(\cdot, \cdot) \). The agent-based formulation turns the challenging one-step pose estimation task into a simpler multi-step correction task. Note in (6) we do not predict a trajectory for the PCA loadings. Unlike rigid pose estimation, which can use the transformed \( SDF_{\mu} \), it not clear how to best inject a concept of PCA state into the input of the encoder without rasterizing a new SDF after every step. Since this is prohibitively expensive, the PCA loadings are directly estimated at each step. We break the search space down even further by first predicting rigid poses then predicting the PCA loadings, as detailed below.

**Rigid Pose Estimation** We first train the encoder \( g_{\theta_E}(\cdot) \) to predict rigid poses. In training we generate samples along a trajectory of \( T \) steps, which is referred to as an episode. The encoder is trained by minimizing a loss calculated over the episodes generated on the training data:

\[
\arg\min_{\theta_E} \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{|X_k|} \sum_{\tau=1}^{T} L(f_{\theta}(A^{-1}T(\omega^\tau)x_i, \mu, s_i)),
\]

where back-propagation is only executed on the encoder weights \( \theta_E \) within each step \( \tau \), and the dependence on \( g_{\theta_E}(\cdot) \) is implied through \( \omega^\tau \). Note that (8) uses the mean latent vector, \( \mu \), to generate SDF predictions and the \( A \) output in (6) is ignored for now. This training process shares similarities to deep reinforcement learning (DRL), particularly in its formulation of the prediction step as an MDP. Unlike DRL, and similar to MDP registration tasks (Liao et al. 2019).
Stage | $\Delta$ | $\eta$ | $\omega^0$
---|---|---|---
Trans. | (t, s) | $\eta_t \circ \eta_s$ | $\omega^0_{T_D}$
Scale | {b, s, t} | $\eta_t \circ \eta_s \circ \eta_b$ | $\omega^0_{b,k}$
Rot. | {b, s, t} | $\eta_t' \circ \eta_s \circ \eta_b'$ | $\omega^0_{b,k}$
Non Rig. | {b, s, t} | $\eta_t' \circ \eta_s \circ \eta_b'$ | $\omega^0_{b,k}$

| Table 1: MSL schedule used in DISSM. |

\[
\Delta^\tau, \lambda^\tau = g_{\theta_E} \left(I_k, \eta^\tau \circ \omega^{\tau - 1}\right),
\]
\[
\omega^\tau = \Delta^\tau \circ \eta^\tau \circ \omega^{\tau - 1}, \quad \text{if } \tau > 0.
\]

A downside to episodic training is that each image is sampled consecutively $T$ times, which can introduce instability and overfitting to the learning process. To avoid this we introduce inverted episodic training, altering the loop order to make each episodic step play out as an outer loop:

\[
\arg \min_{\theta_E} \sum_{\tau=1}^T \sum_{k=1}^{K_D} \sum_{i=1}^{|x_i|} L(f_\theta \left(A^{-1}_{1,t} \omega(x_i)^\tau, \mu + W \lambda^\tau\right), s_i),
\]

where $\omega^\tau$ is saved for each sample after each iteration.

Marginal Space Learning: The MDP of (11) provides an effective sampling strategy, but it requires searching amongst all possible translation, scale, and rotation configurations, which is too large a search space. Indeed we were unable to ever reliably produce trajectories that converged. To solve this, DISSMs use a deep realization of marginal space learning (MSL) (Zheng and Comaniciu 2014). MSL decomposes the search process into a chain of dependant estimates, focusing on one set while marginalizing out the others. In practice (Tab. 1) this means that we first limit the search space by training the encoder to only predict a translation trajectory, $t$, with the random perturbations also limited to only translation, i.e., $\eta_t$. The initial pose is the mean location in the training set, denoted $\omega^0$. Once trained, the translation encoder weights and final poses, $\{\omega^0_{t,k}\}_{k=1}^{K_D}$, are used to initialize a scale encoder of identical architecture, but one that predicts scale corrections, $s$, in addition to fine-tuning the translation. Importantly, to focus the search space on scale, the random translation perturbations are configured to be much smaller than before, which is represented by the prime modifier on $\eta_t$. Finally, a rotation model is trained (while finetuning translation + scale with smaller perturbations). In inference, the rigid pose is estimated by successively applying the models of each stage, using the final pose of the previous step to initialize the next.

Non-Rigid Pose Estimation Once a rigid pose is determined, anatomically plausible deformations can then be estimated. We initialize the weights and poses of the non-rigid encoder using the translation + scale + rotation rigid model, modifying (11) to now incorporate the PCA basis:

\[
\arg \min_{\theta_E} \sum_{\tau=1}^T \sum_{k=1}^{K_D} \sum_{i=1}^{|x_i|} L(f_\theta \left(A^{-1}_{1,t} T(\omega^\tau), \mu + W \lambda^\tau\right), s_i)
\]
\[+ \frac{1}{\sigma^2} \|\mu + W \lambda^\tau\|_2^2.\]

As Table 1 indicates, the random rigid perturbations are configured to be small in magnitude to confine the search space to primarily the PCA loadings.

Surface Refinement

Like classic SSMs (Heimann and Meziner 2009), non-rigid pose estimation provides a robust and anatomically plausible prediction, but it may fail to capture very fine details. We execute local refinements using an FCN model, $r = h_{\theta_R}(I, SDF_\lambda)$, that accepts a two-channel input comprising the 3D image and the rasterized SDF after the non-rigid shape estimation. Its goal is to refine the $SDF_\lambda$ to better match the ground truth SDF. To retain an implicit surface representation, we adapt portions of a recent deep level set loss (Michalkiewicz et al. 2019):

\[
L_r = \sum_{\mathbf{x} \in \Omega_h} (SDF(\mathbf{x})^2 \cdot \delta_r (SDF_\lambda(\mathbf{x}) + r(\mathbf{x})))^{1/2}
\]
\[+ \lambda_1 \sum_{\mathbf{x} \in \Omega_h} (\|\nabla (SDF_\lambda(\mathbf{x}) + r(\mathbf{x}))\| - 1)^2
\]
\[+ \lambda_2 \sum_{\mathbf{x} \in \Omega_h} \max(0, r(\mathbf{x}) - \rho),\]

where $SDF$ is the ground truth SDF and $\delta_r$ is a differentiable approximation of the Dirac-delta function. The first term penalizes mismatch between the iso-boundaries of the refined SDF ground-truth. The second term ensures a unit gradient everywhere, guaranteeing that it remains a proper SDF. See Michalkiewicz et al. (2019) for more details. The third term ensures the refinement does not deviate too much from $SDF_\lambda$ beyond a margin, $\rho$, otherwise it is free to deviate without penalty. Following standard level set practices, we only produce refinements within a narrow band, $\Omega_h$, around the $SDF_\lambda$ iso-boundary, which is also represented in the loss of (13). Finally, in addition to standard data augmentations to $I$, we also independently augment $SDF_\lambda$ with random rigid and non-rigid pose variations, enriching the model’s training set.

Experiments

Liver Dataset: We focus on delineating pathological livers from venous-phase CTs. We use the size-131 training set of the MSD liver dataset (Simpson et al. 2019), splitting it randomly into training, testing, and validation using proportions of 70%, 20%, and 10%, respectively. Tumor masks were merged into the liver to create a complete pathological liver mask. Intra-dataset results are important, but we take a further step to evaluate robustness and generalizability on unseen data from true and challenging clinical scenarios. To do this, we also evaluate on the external test set of
Figure 4: The DISSM pose estimation process. Red and green contours represent the prediction and ground truth, respectively.

Table 2: Quantitative results on the MSD liver dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>DSC (%)</th>
<th>ASSD (mm)</th>
<th>HD (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D-PHNN</td>
<td>95.9 ± 2.7</td>
<td>2.6 ± 1.2</td>
<td>35.7 ± 16.1</td>
</tr>
<tr>
<td>nnU-Net</td>
<td>96.4 ± 1.9</td>
<td>1.7 ± 0.9</td>
<td>29.1 ± 12.4</td>
</tr>
<tr>
<td>H-DenseUNet</td>
<td>96.3 ± 2.1</td>
<td>1.9 ± 1.1</td>
<td>30.4 ± 13.9</td>
</tr>
<tr>
<td>Adv. Shape Prior</td>
<td>96.0 ± 2.4</td>
<td>2.1 ± 1.3</td>
<td>28.9 ± 12.8</td>
</tr>
<tr>
<td>DISSM w/o refine</td>
<td>96.1 ± 0.9</td>
<td>1.5 ± 0.7</td>
<td>23.4 ± 12.2</td>
</tr>
<tr>
<td>DISSM w refine</td>
<td>96.5 ± 0.7</td>
<td>1.1 ± 0.7</td>
<td>21.4 ± 11.8</td>
</tr>
</tbody>
</table>

We compare DISSM against very strong FCN alternatives. 1) 2D P-HNN (Harrison et al. 2017), used as an FCN backbone for a recent semi-supervised liver segmentation method (Raju et al. 2020a). 2) HDenseUNet (Li et al. 2018), a leader of the LiTS liver segmentation challenge (Bilic et al. 2019) that uses a cascade of 2D and 3D FCNs. 3) nnU-Net (Isensee et al. 2021), the winner of the MSD (Simpson et al. 2019) and KiTS (Heller et al. 2020) challenges. We use its dual model 3D cascade setting, which performed best on the liver task of the MSD challenge. 4) nnU-Net augmented with an adversarial anatomical prior that follows Yang et al. (2017). For its discriminator, we use Raju et al. (2020a)’s more modern version, which has already proven effective on the clinical liver dataset. For all, we use their published implementations, including recommended resolutions, pre- and post-processing, and data augmentations. For quantitative comparisons we threshold the DISSM SDF surface to produce a mask and measure the DSC, average symmetric surface distance (ASSD), and HD scores against the original masks. Note, this is a disadvantageous setup for DISSM, as the original masks suffer from stair-like discretization effects, which DISSM aims to rectify.

Larynx Dataset: We also perform supplementary validation on larynx segmentation from CT, a critical organ-at-risk (OAR) for head and neck cancer radiotherapy (Brouwer et al. 2015). This task is highly challenging due to very low contrast boundaries in CT. We compare against the SOARS method and dataset of Guo et al. (2020) (142 CTs), who reported a recent and computationally intensive approach that relies on stratified learning and neural architecture search. However, SOARS is also designed to segment other OARs at the same time, so results are not apples-to-apples. For this reason, we additionally compare against nnU-Net (Isensee et al. 2021), trained only to segment the larynx. More details on this dataset can be found in the supplementary.

Implementation Details: We kept settings as similar as possible for the two datasets. The shape decoder structure and hyperparameters follow that of Park et al. (2019) and we use a size 256 latent variable. For the pose encoders, \( q_{\theta_p}(\cdot) \), use the 3D encoder of a 3D U-Net (Çiçek et al. 2016), with 4 downsampling layers and global averaging pooling to produce \( \Delta^r \) and \( \lambda \). For the liver we estimate the first 28 PCA components (72% of the variance). Larynx shape variations are more constrained, so we estimate the first 12 components (95% variance). The number of inverted episodic steps, \( T \), for training the translation, scale, rotation, and non-rigid encoders was 7, 15, 15, and 15, respectively. The translation encoder was trained on a coarsely sampled volume. After it converged, we cropped volumes to encompass the maximum organ size and trained the remaining pose encoders on higher resolution volumes. Finally, we use a patch-based 3D U-Net (Çiçek et al. 2016) as the local surface refinement model, \( h_{\theta_h}(\cdot) \). Full training details, including the schedule of random pose perturbations and a complete listing of all hyperparameters, can be found in the supplementary.

Liver Results: DISSM consumes 12–13s to fully delineate a 3D volume. Fig. 4 overlays the DISSM results on top of a CT scan after each stage. As can be seen, each stage progressively improves the result. After the non-rigid PCA loading, the delineation quality is already quite high, capturing a lobe curve not represented by the mean shape. The local refinement improves results even further by capturing fine-grained boundary curvatures. An ablation study can be found in the supplementary.

Table 2 outlines the performance on the MSD data. As can be seen, all models perform quite well, but DISSM exhibits...
Given that clinical deployments are the ultimate end goal of delineation solutions, these results offer convincing demonstration of the benefits of imposing strong and descriptive anatomical priors for segmentation.

**Larynx Results:** Table 4 outlines mean DSC and HD scores on the larynx dataset, which is all that SOARS reported (Guo et al. 2020). As can be seen, DISSM outperforms the competitors, posting a mean DSC of 60.9%, which is roughly 2% better than the next best result (nnU-Net). More notably, the standard deviation is significantly reduced (from 14.7 to 5.9), indicating that DISSM is much more reliable. These robustness benefits on a delineation task with challenges distinct from pathological livers, i.e., low contrast boundaries, provide further evidence of the value of the DISSM SSM. Qualitative examples can be found in the supplemental.

### Conclusion

Deep implicit statistical shape models (DISSMs) use a deep implicit model to construct statistical and correspondence-free anatomical priors and directly outputs high-quality surfaces, rather than voxelized masks. For pose estimation, DISSM proposes a robust MDP that incorporates MSL and inverted episodic training. DISSM is the first to integrate a true SSM with deep learning technology. Cross dataset evaluations on pathological liver segmentation demonstrate that DISSM outperforms leading FCNs, e.g., nnU-Net (Isensee et al. 2021), improving the mean and worst-case DSC by 3.5% and 5.1%, respectively. Supplemental validation on a challenging larynx dataset further confirmed the value of DISSM. While continued maturation is necessary, DISSM represents a new and promising approach to 3D medical imaging delineation.
References


Deep Implicit Statistical Shape Models for 3D Medical Image Delineation

**Supplementary Material**

## SDF Decoder Implementation

### Sampling SDF Coordinate/Value Pairs

As mentioned in the main body, to train the shape decoder we require coordinate/signed distance function (SDF) value pairs in a canonical space. Since segmentation labels are almost always provided as voxelized masks, we must first remove the worst of the discretization effects. Because high inter-slice distances are a common issue, we first perform within-slice interpolation (Albu, Beugeling, and Laurendeau 2008) to a common 1mm inter-slice resolution. We then apply Gaussian smoothing using the PyMCubes (pmneila 2020) package to the masks, followed by marching cubes (Lewiner et al. 2003) to produce a good quality mesh. Since this produces an unnecessarily dense set of vertices, we reduce their number using Quadric smoothing (Forstmann 2020). We do this two times to produce two versions: one reduced by a factor of 10 and one by a factor of 100, constructing a simplified and extremely simplified version of each mesh, respectively. The next step is to rigidly align each mesh to an arbitrarily chosen anchor mesh (the first mesh in the training list) using coherent point drift (Myronenko and Song 2010; Khallaghi 2020). We use the extremely simplified mesh versions since it is much more computationally efficient. The computed rigid alignment parameters from the “extremely simplified” meshes are then use to align the “simplified” meshes.

We now have high-quality and aligned meshes, which are all scaled to fit in the unit sphere to create the canonical meshes. To sample SDF values from the meshes, we follow Park et al.’s (Park et al. 2019) approach of randomly sampling within the unit sphere, with regions near the surface much more densely sampled. We use the mesh-to-sdf package (Kleineberg 2020), which recreates the sampling scheme of Park et al. (Park et al. 2019), but we modify its settings. First, we sample a greater proportion of uniformly distributed points (20% of coordinate/SDF value pairs are sampled uniformly across the unit sphere). The other 80% are sampled closer to the boundary by randomly selecting boundary points and jittering them randomly. Unlike Park et al. (Park et al. 2019), we use higher magnitude jitters: either 0.1 or 0.01 (with equal probability) in normalized space. This sampling scheme allowed our shape decoder to converge much better than when using Park et al.’s settings. We sample 1 million coordinate/SDF value pairs per shape to train the SDF decoder.

### Training

For the most part, we follow Park et al.’s settings and hyper-parameters to train the shape decoder multilayer perceptron (MLP). Namely, the MLP structure is a simple 8 layer MLP with 512 channels in each layer and dropout with 20% probability after each layer. The latent code is injected at the fourth layer. Weight normalization (Salimans and Kingma 2016) is used after every layer as well. At every iteration and for every shape, 30 000 coordinate/SDF value pairs are randomly sampled from the list of 1 million to compute the loss. Like Park et al., an equal proportion of positive and negative SDF values are sampled. Unlike Park et al. (Park et al. 2019), we do not use a clamped loss, since the specific SDF values far away from the boundary are still useful for fitting pose. Table S1 lists the hyper-parameters, most of which follow Park et al. (Park et al. 2019). Note, after 1000 epochs, the learning rates were reduced by a factor of 10.

### Pose Encoder Implementation

### Sampling SDF Coordinate/Value Pairs

To train the pose encoders, coordinate/SDF value pairs need to be sampled in the image space, i.e., not the canonical space. To do this, we first generate an SDF from the original

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>100</td>
</tr>
<tr>
<td>Batch Size</td>
<td>6</td>
</tr>
<tr>
<td>SDF Samples/Iteration</td>
<td>30000</td>
</tr>
<tr>
<td>$\mathcal{Z}$ Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>$\theta_S$ Learning Rate</td>
<td>0.0005</td>
</tr>
<tr>
<td>Epochs</td>
<td>2000</td>
</tr>
<tr>
<td>Latent Size</td>
<td>256</td>
</tr>
<tr>
<td>Dropout Probability</td>
<td>0.2</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam (Kingma and Ba 2015)</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Table S1: Hyper-parameters for training the shape decoder
Table S2: Hyper-parameters for training DISSM

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size (Translation)</td>
<td>8</td>
</tr>
<tr>
<td>Batch Size (Scale, Rotation, Non-Rigid)</td>
<td>12</td>
</tr>
<tr>
<td>Epochs (Translation)</td>
<td>65</td>
</tr>
<tr>
<td>Epochs (Scale)</td>
<td>80</td>
</tr>
<tr>
<td>Epochs (Rotation)</td>
<td>80</td>
</tr>
<tr>
<td>Epochs (non-rigid)</td>
<td>120</td>
</tr>
<tr>
<td>SDF Samples/Step</td>
<td>150000</td>
</tr>
<tr>
<td>$\sigma$ (Non-Rigid)</td>
<td>100</td>
</tr>
<tr>
<td>Model Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.0001</td>
</tr>
<tr>
<td>T (Translation)</td>
<td>7</td>
</tr>
<tr>
<td>T (Scale, Rotation, Non-Rigid)</td>
<td>15</td>
</tr>
</tbody>
</table>

Table S3: Perturbation ranges for training the scale encoder.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard Range</th>
<th>Fine Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation (mm)</td>
<td>[−16, 16]</td>
<td>[−8, 8]</td>
</tr>
<tr>
<td>Scale</td>
<td>[0.7, 1.3]</td>
<td>[0.9, 1.1]</td>
</tr>
</tbody>
</table>

Table S4: Perturbation ranges for training the rotation encoder.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard Range</th>
<th>Fine Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation (deg)</td>
<td>[−7.5, 7.5]</td>
<td>[−4.5, 4.5]</td>
</tr>
<tr>
<td>Scale</td>
<td>[0.97, 1.03]</td>
<td>[0.99, 1.01]</td>
</tr>
</tbody>
</table>

Training

We use the encoder structure of Lee et al. ’s (Lee et al. 2017) residual symmetric U-Net architecture to estimate the rigid and non-rigid pose parameters. The encoder structure has 4 residual blocks with max pooling after each block. We perform global average pooling after the final layer of the final residual block, resulting in a size-512 feature vector. We train all pose encoders with an initial learning rate 0.001 and with the Adam optimizer (Kingma and Ba 2015), which is reduced by a factor of 10 after each validation plateau. We truncate the image intensity values of all scans to the range of $[−160, 240]$ Hounsfield units (HU) to remove the irrelevant details. Table S2 lists other hyper parameters for training the deep implicit statistical shape model (DISSM) pose encoders.

Following marginal space learning (MSL) procedures, we first train a translation encoder to predict the translation needed for the mean shape to fit the volume. To estimate translation pose, the model needs to see the entire volume so that the necessary adjustments can be made to move closer to the ground truth. Due to limited computational resources, we resample the volume to $4 \times 4 \times 4$ mm resolution and symmetrically pad across the dimensions to create a unified volume size of physical dimension $512 \times 512 \times 808$ mm, which is large enough to cover all volumes in the medical segmentation decathlon (MSD) training set. To allow DISSM to learn to provide pose corrections from a variety of translation trajectories, we perturb the prior pose at every episodic step by random values within a range of $[−40, 40]$ mm using a uniform distribution.

Once the translation encoder is trained, we crop the resampled volume based on the translated mean shape. The cropped volume is then padded so that we cover enough area around the location to account for any changes in scale. Based on the dataset and a margin for safety, we crop the volume to a $512 \times 432 \times 352$ mm area. The cropped volume is then used to train the scale, rotation, and non-rigid encoders. Focusing on scale first and similar to the translation model, we perturb the trajectory at every episodic step. However, following MSL principles, the translation perturbations are smaller in magnitude than before to focus the search space on scale. Additionally, we also randomly pick between “standard” and “fine-scale” ranges of perturbations.

When the latter is picked, it helps sample more poses closer to the convergence point when the model is at the final steps of its episode. This allows the model to better learn when to stop producing corrections, ensuring it will converge to a pose in inference. We also clip the scale parameters to lie between $[0.5, 2]$ and impose a penalty for any scale predictions exceeding that range:

$$
\max(0, s(\_y) - 2) + \max(0, 0.5 - s(\_z)),
$$

where $s(\_y)$ is any of $s_x$, $s_y$, or $s_z$. This regularization threshold is determined based on the distribution of scales in the training set, which are computed during the rigid alignment coherent point drift step of Sec. . The perturbation hyper parameters for training the scale encoder can be seen in Table S3.

We follow the same procedure to train rotation and non-rigid encoders. The perturbation hyper parameters for training rotation and non-rigid encoders can be seen in Tables S4 and S5, respectively. Finally, we also perform image-based data augmentations such as random affine transformations, random Gaussian noise, random brightness changes, and random intensity shifts for all pose encoders.

Training Local Refinement Model

Once the non-rigid model is trained, we generate the non-rigid SDF for each volume and in the original volume resolution. Because the SDF output from the shape decoder is not...
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard Range</th>
<th>Fine Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation (mm)</td>
<td>[−8, 8]</td>
<td>[−4, 4]</td>
</tr>
<tr>
<td>Scale</td>
<td>[0.99, 1.01]</td>
<td>[0.995, 1.015]</td>
</tr>
<tr>
<td>Rotation (deg)</td>
<td>[−4.5, 4.5]</td>
<td>[−2.5, 2.5]</td>
</tr>
</tbody>
</table>

Table S5: Perturbation ranges for training the non-rigid encoder. The “standard” and “fine” ranges are randomly chosen at each step with 50% probability each. Values are then randomly sampled within the resulting ranges using a uniform distribution.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>2</td>
</tr>
<tr>
<td>Epochs</td>
<td>150</td>
</tr>
<tr>
<td>Model Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.0001</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table S6: Hyper-parameters for training local refinement model

necessarily a proper SDF, we reinitialize it (Sethian 1999) using the ITK software (McCormick et al. 2014). To make the model robust to variations in the SDF, prior to generating each SDF we randomly add Gaussian noise with mean 0 and standard deviation 0.01 to the predicted latent vector. We generate 10 non-rigid SDFs for each volume this way, randomly choosing one SDF to pair with the image when training.

We follow the standard 3D U-Net architecture from (Ciccek et al. 2016) and input both the original volume along with randomly picked non-rigid SDF from the pool of SDFs for each volume. We further augment the SDF channels through small rigid random affine transformations that are independent of the image channel. We weight the loss with $\lambda_1$ and $\lambda_2$ in (13) using the values mentioned in Table S6. As mentioned in the main body, we only produce refinements within a narrow band of the non-rigid SDFs iso-boundary, which allows the model to focus only on the surface of the liver. We define the narrow band as being within 25 mm of the iso-boundary. Finally, we set the $\rho$ in (13) to 12 mm. We apply standard data augmentations to both input channels, such as random affine transformations (both channels), and random shifting intensity, random Gaussian noise, and random brightness shifts (only image channel) while training the local refinement model.

**Competitor Methods**

We compare DISSM with the competitor methods 2D PHNN (Raju et al. 2020), H-DenseUNet(Li et al. 2018), nnU-Net (cascade)(Isensee et al. 2018) for the liver segmentation task. We follow the same procedure including the data preprocessing, data augmentation and post processing as mentioned in their respective papers. nnU-Net cascade preprocesses the image by resampling to the median voxel spacing. Third order spline interpolation is used for image data and nearest neighbour interpolation for label data. H-DenseUNet preprocesses the image by resampling to a fixed resolution $0.69 \times 0.69 \times 1 mm^3$ with the HU values clipped to $[−200, 250]$. For 2D PHNN, we use the original resolution and clip the HUs to $[−200, 250]$. For post processing, we select largest connected component for all the competitor methods coupled with 3D hole-filling.

**Larynx CT Dataset**

The larynx dataset contains 142 non-contrast CT images from head and neck cancer patients. The larynx annotations are from the target delineation process during patients’ radiotherapy treatment. We use the same one fold split as used in the ablation study of SOARS (Guo et al. 2020) to train and evaluate the performance of DISSM and nnU-Net, which includes 89, 17 and 36 training, validation and testing patients. Similar to MSD dataset, we resample the volume to $4 \times 4 \times 1.5$ mm resolution and pad across the dimensions to create an unified volume of size of physical dimension $512 \times 512 \times 612$ mm. We further crop the volume after translation model to a $432 \times 284 \times 156$ mm to train scale, rotation and non-rigid encoders.

**References**


DenseUNet preprocesses the image by resampling to a fixed resolution $0.69 \times 0.69 \times 1 mm^3$ with the HU values clipped to $[−200, 250]$. For 2D PHNN, we use the original resolution and clip the HUs to $[−200, 250]$. For post processing, we select largest connected component for all the competitor methods coupled with 3D hole-filling.
Figure S1: Qualitative comparison of DISSM (without or with refinement) versus four state-of-the-art competitor approaches (Harrison et al. 2017; Isensee et al. 2021; Li et al. 2018; Yang et al. 2017). Red and green contours represent the current estimate and the ground truth of liver delineations, respectively. The first and second-to-fourth rows are drawn from the MSD and clinical dataset, respectively. Fourth row is worst-case performance for DISSM.


